Nonlinear Gaussian Filtering

— Theory, Algorithms, and Applications —

HABILITATIONSSCHRIFT

zur Erlangung der Venia Legendi
für das Fach Informatik

vorgelegt der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

von

Dr.-Ing. Marco Huber

aus Kehl

Eingereicht am:	14.02.2014
Hauptgutachter:	Prof. Dr.-Ing. Jürgen Beyerer
Zweitgutachter:	Prof. Hugh Durrant-Whyte
	Prof. Peter K. Willett

Statement of Authorship

I hereby declare that the work submitted for review is entirely of my own hand, or, where produced in collaboration with others, has been clearly identified as such. I further declare that I have labeled all material that has been taken with or without modification from the work of others.

Weinheim, 14.02.2014 Dr.-Ing. Marco Huber

Acknowledgement

While preparing this thesis, I have worked for three different organizations. Accordingly, I met many people that contributed to this work the one way or the other. I would like to spent the following lines to thank for the time, work, diversion, and inspiration invested by the contributors.

The first contributions to this thesis were made at the Intelligent Sensor-Actuator-Systems Laboratory (ISAS) of the Karlsruhe Institute of Technology (KIT). I would like to thank *Uwe D. Hanebeck*, director of ISAS, for supporting this thesis from the beginning. Also a big thank you to *Frederik Beutler* for his contributions to the semi-analytic filtering stuff and for running the experiments with the "Holodeck" as well as to *Peter Krauthausen* for his contributions to Gaussian mixture reduction. But most importantly: for their friendship.

When I moved on to the Fraunhofer Institute of Optronics, System Technologies, and Image Exploitation (IOSB), I met *Jürgen Beyerer*, who is the director of this institute. He actually motivated me for starting the whole habilitation endeavor and took the responsibility of the official habilitation process at the KIT. I can imaging that this work together with preparing a review of this thesis was consuming a significant share of time. So, many thanks for this.

At IOSB and the associated Vision and Fusion Laboratory (IES) I worked with many talented people. For the many fruitful discussions, barbecues, and for other thrilling activities I would like to thank *Christian Frese*, *Ioana Gheţa*, *Martin Grafmüller*, *Robin Gruna*, *Michael Heizmann*, *Sebastian Höfer*, *Christian Kühnert*, *Achim Kuwertz*, *Alexey Pak*, *Jennifer Sander*, *Michael Teutsch*, *Stefan Werling*, and last but not least *Philipp Woock*. Special thanks go to *Masoud Roschani*. Besides being a strong author of one of the papers used for this thesis, he acted and still acts as a very reliable contributor, organizer, and backup for my lecture

“Probabilistische Planung". I would also like to thank *Gaby Gross*, who is by far more than a secretary for the IES. I enjoyed sharing work time and experiences about traveling and football with her.

The last three and a half years I worked for AGT International in Darmstadt, which was an interesting experience. I appreciate the time and space that was provided and blocked by both *Zachos Boufidis* and *Panayotis Kikiras*. They were great supporters of this thesis. I also have to thank *Florian Zeiger* for his intelligence, insights, and most importantly his friendship. Working at AGT would have been less fun without him. Finally, I would like to thank my students *Tobias Dencker* and *Jason Rambach* for their great work.

Outside these three organizations I would like to thank *Marc Deisenroth* for his ideas on Gaussian Processes and his bright mind in general. I'm also grateful for the review work done by *Hugh Durrant-Whyte* and *Peter Willett* and for the support that was given to this thesis by many computer science professors of the KIT. Here, I would like to mention *Rainer Stiefelhagen* and *Tamim Asfour* among many others.

Such a work not only needs to be backed from a professional side. Of at least the same importance is the support and understanding from beloved. I'm grateful for having *Karin* by my side. Life has become much more colorful with her and with our kids *Anna-Lena* and *Paul*.

Weinheim, January 2015 Marco Huber

To Anna-Lena and Paul

Contents

I Background & Summary 1

1 Introduction 3
- 1.1 Nonlinear Bayesian Filtering 4
  - 1.1.1 Dynamic Models and Measurement Models 4
  - 1.1.2 Recursive Filtering 6
  - 1.1.3 Closed-form Calculation 8
  - 1.1.4 Approximate Filtering: State of the Art 9
- 1.2 Research Topics 12
- 1.3 Main Contributions 14
  - 1.3.1 Gaussian Filtering 15
  - 1.3.2 Gaussian Mixture Filtering 18
  - 1.3.3 Gaussian Process Filtering 19
- 1.4 Thesis Outline 21

2 Gaussian Filtering 23
- 2.1 The Gaussian Distribution 24
  - 2.1.1 Importance of the Gaussian 24
  - 2.1.2 Dirac Delta Distribution 25
  - 2.1.3 The Exponential Family 26
- 2.2 Exact Gaussian Filtering and Approximations 27
  - 2.2.1 General Formulation 27
  - 2.2.2 Linear Filtering 29
  - 2.2.3 Linearized and Extended Kalman Filter 31
  - 2.2.4 Statistical Linearization 32
  - 2.2.5 Linear Regression Kalman Filters 33

2.3 Gaussian Smoothing . . . 39
2.3.1 General Formulation . . . 40
2.3.2 Linear Case . . . 41
2.3.3 Nonlinear Case . . . 42
2.4 Rao-Blackwellization . . . 42
2.5 Contributions . . . 43
2.5.1 Combining Rao-Blackwellization with Observed-Unobserved Decomposition . . . 43
2.5.2 Semi-Analytical Filtering . . . 47
2.5.3 Chebyshev Polynomial Kalman Filtering . . . 52
2.5.4 Efficient Moment Propagation for Polynomials . . . 55
2.5.5 Homotopic Moment Matching for Polynomial Measurement Models . . . 63
2.6 Summary . . . 70

3 Gaussian Mixture Filtering 73
3.1 Gaussian Mixtures . . . 74
3.2 Nonlinear Filtering . . . 75
3.2.1 Individual Approximation . . . 76
3.2.2 Generic Gaussian Mixture Filter . . . 78
3.3 Component Adaptation . . . 79
3.3.1 Weight Optimization . . . 80
3.3.2 Reduction . . . 80
3.3.3 Refinement . . . 84
3.4 Contributions . . . 86
3.4.1 Semi-Analytic Gaussian Mixture Filter . . . 86
3.4.2 Adaptive Gaussian Mixture Filter . . . 90
3.4.3 Curvature-based Gaussian Mixture Reduction . . . 96
3.5 Summary . . . 103

4 Gaussian Process Filtering 105
4.1 Gaussian Processes . . . 106
4.2 Covariance Functions . . . 109
4.2.1 Examples . . . 109
4.2.2 Hyperparameter Learning . . . 111
4.3 Large Data Sets . . . 112
4.3.1 Active Set Approaches . . . 113
4.3.2 Local Approaches . . . 114
4.3.3 Algebraic Tricks . . . 114

4.3.4 Open Issues . . . 115
4.4 Nonlinear Filtering . . . 115
4.5 Contributions . . . 116
4.5.1 Gaussian Process Filtering . . . 116
4.5.2 Gaussian Process Smoothing . . . 121
4.5.3 Recursive Gaussian Process Regression . . . 124
4.5.4 On-line Hyperparameter Learning . . . 129
4.6 Summary . . . 135

5 Applications **137**
5.1 Range-based Localization . . . 137
5.1.1 Position Estimation . . . 139
5.1.2 Position and Orientation Estimation . . . 144
5.2 Gas Dispersion Source Estimation . . . 149
5.2.1 Atmospheric Dispersion Models . . . 150
5.2.2 Parameter Estimation . . . 152
5.3 Active Object Recognition . . . 156
5.3.1 Object Classification . . . 157
5.3.2 Learning . . . 158
5.3.3 Estimation . . . 158
5.3.4 Planning . . . 160
5.4 Summary . . . 164

6 Concluding Remarks **165**
6.1 Conclusions . . . 165
6.2 Future Work . . . 166

A Particle Filtering **169**
A.1 Perfect Monte Carlo Sampling . . . 169
A.2 Importance Sampling . . . 170
A.2.1 Sequential Importance Sampling . . . 171
A.2.2 Choice of Importance Function . . . 172
A.2.3 Resampling . . . 172

B Performance Measures **175**
B.1 Root Mean Square Error . . . 175
B.2 Mean Absolute Error . . . 176
B.3 Normalized Estimation Error Square . . . 176
B.4 Negative Log-Likelihood . . . 177

C Quadratic Programming **179**

Bibliography **181**

II Publications **201**

A Gaussian Filtering using State Decomposition Methods **203**
1 Introduction . . . 205
2 Problem Formulation . . . 207
3 The Gaussian Filter (GF) . . . 208
4 Decomposition Methods . . . 210
4.1 Case I: directly observed, indirectly observed . . . 210
4.2 Case II: Linear, Nonlinear . . . 211
5 Estimation . . . 214
5.1 Prediction Step . . . 214
5.2 Filter Step . . . 215
6 Considered Example . . . 216
6.1 Prediction . . . 216
6.2 Filtering . . . 219
7 Results . . . 220
7.1 Simulation . . . 220
7.2 Experiment . . . 222
8 Conclusions . . . 224

B Semi-Analytic Gaussian Assumed Density Filter **227**
1 Introduction . . . 229
2 Problem Formulation . . . 231
2.1 Bayesian Estimation . . . 231
2.2 Gaussian Assumed Density Filter . . . 232
3 Analytic Gaussian Assumed Density Filter (AGF) . . . 233
4 Sample-Based Gaussian Assumed Density Filter (SGF) . . . 235
5 Semi-Analytic Gaussian Assumed Density Filter (SAGF) . . . 237
5.1 General Solution . . . 239
5.2 Estimation . . . 240
6 Simulation Examples . . . 241
6.1 System Equation . . . 241
6.2 Case I: Linear Measurement Equation . . . 242
6.3 Case II: Nonlinear Measurement Equation . . . 242

6.4 Simulation Results . . . 243
7 Discussion and Future Work . . . 246

C Chebyshev Polynomial Kalman Filter 249
1 Introduction . . . 251
2 Problem Formulation . . . 253
3 Chebyshev Polynomials . . . 255
3.1 Properties . . . 255
3.2 Chebyshev Series . . . 256
3.3 Approximate Series Expansion . . . 257
4 Closed-Form Moment Propagation . . . 258
4.1 Non-central Moments of a Gaussian . . . 259
4.2 Efficient Mean Propagation . . . 261
4.3 Efficient Variance Propagation . . . 262
5 The Gaussian Estimator . . . 263
5.1 Approximation Interval . . . 263
5.2 Prediction . . . 264
5.3 Measurement Update . . . 264
6 Results . . . 266
6.1 Example I: Higher-Order Moments . . . 266
6.2 Example II: Real-World Application . . . 267
6.3 Example III: Time Series . . . 270
7 Discussion . . . 274
7.1 Strengths . . . 274
7.2 Limitations and Potential Extensions . . . 275
8 Conclusions . . . 276
A State Decomposition . . . 276

D Gaussian Filtering for Polynomial Systems Based on Moment Homotopy 283
1 Introduction . . . 285
2 Problem Formulation . . . 287
3 Exponential Densities . . . 288
3.1 Definition . . . 288
3.2 Recursive Moment Calculation . . . 289
4 Gaussian Filtering . . . 290
4.1 Mean Propagation . . . 290
4.2 Variance Propagation . . . 291
4.3 Covariance Calculation . . . 292

4.4 Polynomial Kalman Filter 292
4.5 Discussion 294
5 Homotopic Bayesian Measurement Update 297
5.1 Parameterization 298
5.2 System of Ordinary Differential Equations 299
5.3 Initialization and Solution 300
5.4 Homotopic Polynomial Gaussian Filter 301
6 Results 302
6.1 Moment Homotopy Examples 302
6.2 Chaotic Synchronization 304
7 Conclusion and Future Work 306

E (Semi-)Analytic Gaussian Mixture Filter 309
1 Introduction 311
2 Problem Formulation 313
2.1 Prediction Step 313
2.2 Filter Step 314
3 Gaussian Mixture Filter 314
3.1 Prediction Step 314
3.2 Filter Step 316
3.3 Gaussian Mixture Noise 317
4 Types of Linearization 317
4.1 Classical Linearization 318
4.2 Analytic Stochastic Linearization 319
4.3 Semi-Analytic Stochastic Linearization 320
5 Simulation Results 323
5.1 System and Measurement Model 323
5.2 Estimator: A-GMF 324
5.3 Estimator: SA-GMF 325
5.4 Setup 325
5.5 Results 326
6 Conclusions 328
A Analytic Expressions for A-GMF 329

F Adaptive Gaussian Mixture Filter Based on Statistical Linearization 333
1 Introduction 335
2 Problem Formulation 337

3 Statistical Linearization . . . 338
3.1 Classical Linearization . . . 339
3.2 Statistical Linear Regression . . . 339
3.3 Calculating the Regression Points . . . 340
4 Splitting Scheme . . . 342
4.1 Component Selection . . . 342
4.2 Splitting a Gaussian . . . 343
4.3 Splitting Direction . . . 346
5 Adaptive Gaussian Mixture Filter . . . 347
5.1 Prediction Step . . . 347
5.2 Filtering Step . . . 350
6 Simulation Results . . . 351
6.1 Shape Approximation . . . 351
6.2 Object Tracking . . . 352
7 Conclusions . . . 356

G Superficial Gaussian Mixture Reduction 359
1 Introduction . . . 361
2 Problem Statement . . . 364
3 Superficial Gaussian Mixture Reduction . . . 365
3.1 Distance Measure . . . 366
3.2 Upper bound of Curvature . . . 366
3.3 Algorithm . . . 367
4 Limitations . . . 369
5 Experiments . . . 370
5.1 1D Experiment . . . 372
5.2 2D Experiment . . . 374
6 Conclusion . . . 375

H Analytic Moment-based Gaussian Process Filtering 379
1 Introduction . . . 381
2 Model and Problem Statement . . . 383
3 Related Work . . . 385
4 Gaussian Processes . . . 386
4.1 Predictions for Uncertain Inputs . . . 387
4.2 Multivariate Predictions . . . 388
5 GP-ADF: Assumed Density Filtering with Gaussian Processes . . . 389
5.1 Prediction Step . . . 390
5.2 Filter Step . . . 390

5.3 Assumptions and Computational Complexity 392
6 Results 392
6.1 1D Example 393
6.2 Recursive Filtering: Time-Series 395
7 Discussion 397
8 Summary and Future Work 398

I Robust Filtering and Smoothing with Gaussian Processes 401
1 Introduction 404
1.1 Problem Formulation and Notation 405
1.2 Gaussian RTS Smoothing 405
2 Gaussian Processes 406
2.1 Expressiveness of the Model 407
2.2 Training via Evidence Maximization 409
3 Robust Smoothing in Gaussian Process Dynamic Systems 409
3.1 Marginal Distribution 410
3.2 Cross-Covariance 413
4 Simulations 415
4.1 Filter Robustness 415
4.2 Smoother Robustness 419
5 Discussion and Conclusion 420

J Recursive Gaussian Process Regression 425
1 Introduction 427
2 Problem Statement 428
3 Recursive Processing 430
3.1 Inference 431
3.2 Update 433
3.3 Summary 434
4 Discussion 435
5 Simulation Examples 436
6 Conclusions 440
A Useful Lemmas 440

K Recursive Gaussian Process: On-line Regression and Learning 445
1 Introduction 447
2 Problem Formulation 448
3 On-line Regression 450
3.1 Inference 451

3.2 Update . . . 452
4 On-line Learning . . . 453
4.1 Inference . . . 454
4.2 Update . . . 456
5 Discussion . . . 458
6 Results . . . 459
6.1 Synthetic Data . . . 460
6.2 Particulate Matter Data . . . 463
7 Conclusion and Future Work . . . 464

L Optimal Stochastic Linearization for Range-Based Localization 467
1 Introduction . . . 469
2 Problem Formulation . . . 471
2.1 Dynamics Model . . . 471
2.2 Measurement Models . . . 471
3 Recursive State Estimation . . . 472
3.1 Prediction Step . . . 472
3.2 Filter Step . . . 473
4 Analytic Moment Calculation (AMC) . . . 474
4.1 Modified Measurement Equation . . . 474
4.2 Moment Calculation . . . 475
4.3 Summary . . . 478
4.4 Computational Complexity . . . 479
5 Experiments . . . 479
5.1 Simulation . . . 480
5.2 Experiment . . . 482
6 Conclusions . . . 483

M Semi-Analytic Stochastic Linearization for Range-Based Pose Tracking 487
1 Introduction . . . 489
2 Problem Formulation . . . 491
2.1 Measurement Equation . . . 491
2.2 System equation . . . 492
2.3 Recursive Gaussian State Estimation . . . 493
3 Semi-Analytic Linearization . . . 495
3.1 Modified Measurement Equation . . . 495
3.2 Decomposition . . . 496
3.3 Mean . . . 497

3.4 Covariance 498
3.5 Cross-Covariance 499
3.6 Summary 500
3.7 Computational Complexity 500
4 Simulation Results 501
5 Conclusions 502

N On-line Dispersion Source Estimation 505
1 Introduction 507
2 Problem Formulation 509
2.1 General Dispersion Model 509
2.2 Gaussian Plume 509
3 Recursive Estimation 511
3.1 Bayesian Estimation 511
3.2 Measurement Model 512
3.3 Approximate Estimation 513
4 Adaptive Gaussian Mixture Filter 513
4.1 Statistical Linearization 514
4.2 Splitting 515
4.3 Filtering 517
4.4 Reduction 517
5 Results 518
5.1 AGMF Parametrization 518
5.2 Indianapolis Field Study 519
5.3 Simulation 521
6 Conclusion 525

O Bayesian Active Object Recognition via Gaussian Process Regression 529
1 Introduction 531
2 Problem Formulation 533
3 Gaussian Process Regression 535
4 Active Object Recognition 536
4.1 Learning Object Models 536
4.2 Bayesian Estimation 537
4.3 Next-Best-View Planning 539
5 Simulation Results 542
5.1 General Simulation Setup 542
5.2 Example I: Cups 543

5.3 Example II: Toy Manikins 545
6 Conclusion and Future Work 547
A Gradient 547

Notation

Conventions

x	Scalar
$\underline{x}$	Column vector
$\boldsymbol{x}$, $\underline{\boldsymbol{x}}$	Random variable/vector
$\hat{x}$, $\underline{\hat{x}}$	Realization of a random variable/vector
$\bar{x}$, $\underline{\bar{x}}$	Nominal point/vector
n_x	Dimension of vector $\underline{x}$
L_x, N_x	Number of elements of type x
$(\cdot)^p$	Predicted quantity (after prediction step)
$(\cdot)^e$	A posteriori quantity (after measurement update)
$(\cdot)^s$	Smoothed quantity (after smoothing)
$(\cdot)^x$, $(\cdot)_x$	Quantity related to variable/vector x
$(\cdot)_l$, $(\cdot)_n$	(Conditionally) linear and nonlinear substate
$(\cdot)_o$, $(\cdot)_u$	Observed and unobserved substate
$(\cdot)_k$	Quantity at discrete time step k
$(\cdot)_{0:k}$	Time sequence of quantities $\big((\cdot)_0, (\cdot)_1, \dots, (\cdot)_k\big)$
$\tilde{(\cdot)}$	Approximate quantity
$(\cdot)^*$	Optimal solution of an optimization problem
$i : j$	Sequence of integers from i to j with $i < j$
$\cdot\vert\cdot$	Conditioning, i.e., the left-hand quantity is conditioned on the right-hand quantity
$\mathbf{A}$	Matrices are denoted by bold upper case letters
$\mathcal{A}$	Sets are denoted by calligraphic upper case letters

Conventions (Cont'd)

$g(\mathbf{X})$	Evaluation of function $g:\mathbb{R}^n \rightarrow \mathbb{R}$ for every column of matrix $\mathbf{X} = [\underline{x}_1\ \underline{x}_2\ \ldots \underline{x}_m]^{\mathrm{T}}$ with $\underline{x}_i \in \mathbb{R}^n$, i.e., the results corresponds to vector $[g(\underline{x}_1)\ g(\underline{x}_2)\ \ldots g(\underline{x}_m)]^{\mathrm{T}} \in \mathbb{R}^m$
$\mathbb{N}$	Natural numbers
$\mathbb{R}$	Real numbers
$\square$	End of theorem

Operators

$\mathrm{diag}(\underline{x})$	Diagonal matrix with main diagonal elements according to $\underline{x}$
$\mathrm{diag}(\mathbf{A})$	Diagonal matrix with main diagonal elements taken from the main diagonal of $\mathbf{A}$
$\mathrm{vec}(\mathbf{A})$	Matrix vectorization, i.e., the columns of $\mathbf{A}$ are stacked to form a vector
$\mathbf{A}^{\mathrm{T}}$	Matrix transpose
$\mathbf{A}^{-1}$	Matrix inverse
$\mathrm{Tr}(\mathbf{A})$	Matrix trace
$\lvert\mathbf{A}\rvert$	Matrix determinant
$\lVert.\rVert_p$	p-Norm, where $p \in \mathbb{N}$. For $p = 2$, it corresponds to the L_2 or Euclidean norm
$\triangleq$	Definition
$\equiv$	Equivalent
$\preceq, \succeq$	Element-wise comparison of two vectors, i.e., $\underline{x} \preceq \underline{y}$ iff $x_i \leq y_i$ for all elements x_i, y_i of $\underline{x}$, $\underline{y}$
$*$	Convolution
$\dot{x}$	Time derivative of quantity x
$\otimes$	Kronecker product
$\odot$	Hadamard (element-wise) product
$\sim$	Distribution symbol
$\propto$	Proportional
$\mathrm{E}\{\cdot\}$	Expected value
$\mathrm{var}\{\cdot\}$	Variance
$\mathrm{Cov}\{\cdot\}$	Covariance
$\mathcal{O}(\cdot)$	Complexity class (Big-O according to Landau notation)

Special Functions

$\mathcal{N}(\underline{x};\underline{\mu},\mathbf{C})$	Multivariate Gaussian density function with mean vector $\underline{\mu}$ and covariance matrix $\mathbf{C}$
$\mathcal{N}(\underline{\mu},\mathbf{C})$	Multivariate Gaussian distribution
$\mathcal{GP}(\mu,\kappa)$	Gaussian process with mean function $\mu(\cdot)$ and covariance function $\kappa(\cdot,\cdot)$
$\delta(\underline{x})$	Dirac delta distribution
$\delta_{i,j}$	Kronecker delta function
$D(.,.)$	Integrated squared difference (ISD)
$G(.\|.)$	Kullback-Leibler divergence (KLD)
$\mathrm{H}(.)$	Shannon/differential entropy
$\mathrm{I}(.,.)$	Mutual information

Reserved Symbols

$f(\cdot), p(\cdot)$	General symbols for probability density functions
$\underline{g}(\cdot)$	General symbol for a nonlinear function
$\underline{a}_k(\cdot)$	System function
$\underline{h}_k(\cdot)$	Measurement function
$\underline{\boldsymbol{x}}$	System state
$\underline{\boldsymbol{z}}$	Measurement/observation
$\underline{\boldsymbol{w}}$	System noise
$\underline{\boldsymbol{v}}$	Measurement noise
$\underline{\boldsymbol{\theta}}$	(Hyper)parameter vector
$\underline{\mu}$	Mean vector
$\mathbf{C}$	Covariance matrix
$\mathbf{A}$	System matrix
$\mathbf{H}$	Measurement matrix
$\mathbf{K}$	Kalman gain matrix or kernel matrix
$\mathbf{J}$	Smoothing gain matrix
$\mathbf{I}_n$	Identity matrix of dimension $n \times n$
$\mathbf{0}_n$	Zero matrix of dimension $n \times n$
$\mathcal{X}_i$	Sample or sigma point
ω_i	Weighting coefficient
E_i	Non-central moment of order i
$\mathcal{D}$	Data set

Abbreviations

ADF	Assumed density filter
ADM	Atmospheric dispersion model
AGMF	Adaptive Gaussian mixture filter
CPKF	Chebyshev polynomial Kalman filter
CKF	Cubature Kalman filter
EKF	Extended Kalman filter
GPS	Global positioning system
GHKF	Gauss-Hermite Kalman filter
GP	Gaussian process
HPGF	Homotopic polynomial Gaussian filter
ISD	Integrated squared difference
KLD	Kullback-Leibler divergence
LRKF	Linear Regression Kalman filter
mae	Mean absolute error
MC	Monte Carlo
MCMC	Markov chain Monte Carlo
nees	Normalized estimation error square
nll	Negative log-likelihood
NN	Neural network
ODE	Ordinary differential equation
PF	Particle filter
PKF	Polynomial Kalman filter
QP	Quadratic program
RGP	Recursive Gaussian process regression
rmse	Root mean square error
RTSS	Rauch-Tung-Striebel smoother
SAGF	Semi-analytic Gaussian filter
SAGMF	Semi-analytic Gaussian mixture filter
SE	Squared exponential
SGMR	Superficial Gaussian mixture reduction
SIR	Sequential importance resampling
SOGP	Sparse on-line Gaussian process
SPGP	Sparse pseudo-input Gaussian process
SVM	Support vector machine
UKF	Unscented Kalman filter
UT	Unscented Transformation

Part I

Background & Summary

1 Introduction

In the early 1960s, the researchers of the NASA Apollo program faced severe problems with the navigation of their spacecraft. By observing external bodies like earth, moon, and stars, the pilot had to estimate the position, orientation, and velocity of the vehicle. These observations in combination with the high-dimensional models describing the dynamics of the spacecraft should allow correct guidance and trajectory following. At that point in time, the estimation toolbox merely contained algorithms like weighted least-squares and the Wiener filter. While the first one was computationally too complex for the on-board computer of the spacecraft, the latter was mathematically very involved and discrete-time measurements were not supported.

It was a lucky coincidence that at the same time Rudolf Kalman invented his famous recursive filtering approach, which is nowadays known as the Kalman filter. As Rudolf Kalman gave a presentation of his novel approach to NASA researchers, many of the open problems suddenly seemed to be solvable, even though a direct application of the Kalman filter was not possible—it was designed for linear problems, while spacecraft navigation is a nonlinear one. In addition, this novel filtering technique raised new questions like “How to deal with numerical instabilities or systematic measurement errors?”, which are always present when it comes to a realization to practice. At the end the navigation problem was solved and the rest of the story is well-known history. For more details see [122].

The implementation of a modified version[1] of the Kalman filter for spacecraft navigation as part of the Apollo Guidance Computer was one of the first—if not the first—technical realization of *nonlinear Bayesian filtering* theory, which is the general topic of this thesis. Besides spacecraft navigation, nonlinear Bayesian filtering appears in many technical fields like robotics, control, telecommunications, signal processing, data fusion, or machine learning, where one wants to estimate the latent parameters or state[2] of a nonlinear dynamic system from noisy measurements and imperfect knowledge. It provides a general framework in which all appearing uncertainties are represented by means of probability distributions and the processing of these distributions occurs according to the calculus of probability theory[3].

1.1 Nonlinear Bayesian Filtering

The latent *system state* $\underline{x} \in \mathbb{R}^{n_x}$ comprises the smallest set of variables, which is necessary to completely describe the dynamic behavior of the considered system at any time instant. In order to estimate the latent state, three main components are required: First, a model of the system dynamics. Second, a model of the sensor. Both include statistical models of the noise processes affecting the system and the sensor. Third and finally, the actual Bayesian filtering algorithm. The interactions of these three components are depicted in Figure 1.1 and mathematical formulations of the components are introduced in the following.

1.1.1 Dynamic Models and Measurement Models

The temporal evolution of the latent system state is typically described by means of differential equations in continuous time according to

$$\dot{\underline{\boldsymbol{x}}}(t) = \underline{\phi}\big(\underline{\boldsymbol{x}}(t), \underline{u}(t), \underline{\boldsymbol{w}}(t)\big) , \tag{1.1}$$

1 Kalman-Schmidt filter, which is nowadays called the extended Kalman filter.

2 In this thesis, parameters and state are not distinguished if not stated explicitly. Thus, only the notion of a latent state is used from now on.

3 This is in contrast to *frequentist statistics*, where the probability of an event reflects the proportion of the event in an infinite number of trials. In the Bayesian viewpoint however, the probability describes the uncertainty of an event in a single trial [19, 159].

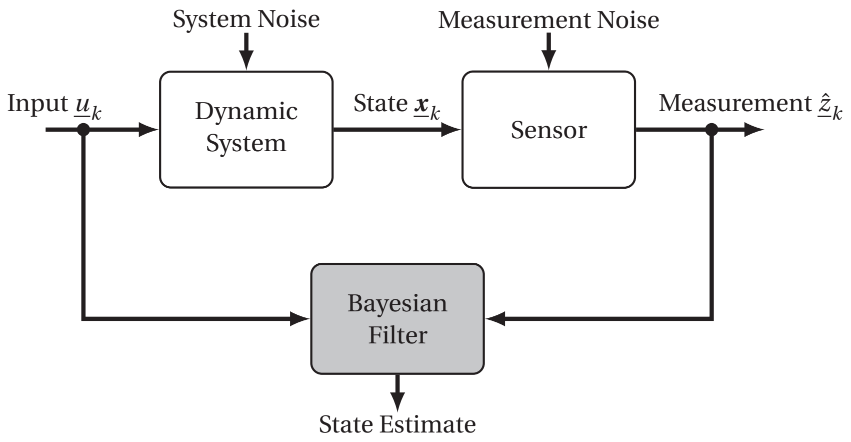


Figure 1.1: Block diagram of a dynamic system incorporating a Bayesian filter for estimating the latent state $\underline{\boldsymbol{x}}_k$.

which models the system dynamics and thus, is denoted the *dynamic model.* Here, $\dot{\underline{\boldsymbol{x}}}(t)$ is the derivative of the system state $\underline{\boldsymbol{x}}(t)$ with respect to time. For realization on a computer, a discrete-time version of the dynamic model (1.1) is required, which is given by

$$\underline{\boldsymbol{x}}_{k+1} = \underline{a}_k\big(\underline{\boldsymbol{x}}_k, \underline{u}_k, \underline{\boldsymbol{w}}_k\big) , \tag{1.2}$$

where the random vector $\underline{\boldsymbol{x}}_k$ is the system state at discrete time step $k = 0,1,\ldots$ The time steps are related via $t_{k+1} = t_k + T$, where T is the sampling time. The state itself is represented by means of the probability density function $f_k^x\big(\underline{x}_k\big)$ at time step k. Furthermore, $\underline{a}_k(.)$ is the nonlinear system function, $\underline{u}_k$ is the vector of deterministic system inputs, and $\underline{\boldsymbol{w}}_k$ is the system noise. In the reminder of this thesis, only discrete-time models are considered.

As the state is assumed to be latent, i.e., it cannot be directly observed, measurements of a sensor are required to gather information about the system state. The *measurement model* in discrete-time is given by

$$\underline{\boldsymbol{z}}_k = \underline{h}_k\big(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{v}}_k\big) , \tag{1.3}$$

with nonlinear measurement function $\underline{h}_k(.)$, measurement noise $\underline{\boldsymbol{v}}_k$, and measurement $\underline{\boldsymbol{z}}_k$. An actual measurement $\underline{\hat{z}}_k$ of the sensor is a realization of $\underline{\boldsymbol{z}}_k$.

An important special case of the general models in (1.2) and (1.3) is the additive noise case. Here, the dynamic model and the measurement model are given by

$$\begin{aligned} \underline{\boldsymbol{x}}_{k+1} &= \underline{a}_k\big(\underline{\boldsymbol{x}}_k, \underline{u}_k\big) + \underline{\boldsymbol{w}}_k \,, \\ \underline{\boldsymbol{z}}_k &= \underline{h}_k\big(\underline{\boldsymbol{x}}_k\big) + \underline{\boldsymbol{v}}_k \,, \end{aligned} \tag{1.4}$$

respectively. This special case simplifies Bayesian filtering significantly, as the transition density and likelihood can be expressed analytically (see next section), but still it is sufficient for modeling a large class of important estimation problems [73].

The noise processes $\underline{\boldsymbol{w}}_k$ and $\underline{\boldsymbol{v}}_k$ in (1.2) and (1.3), respectively, account for typical uncertainties affecting the dynamic and measurement model. Examples are modeling uncertainties or external disturbances [73]. It is assumed that both noise processes are *white*. That is, both noise terms at time k are independent of the system state $\underline{\boldsymbol{x}}_k$ and independent of the noises at any other time step $n \neq k$. Both noise processes are presented by means of the probability density functions $f_k^w\big(\underline{w}_k\big)$ and $f_k^v\big(\underline{v}_k\big)$, respectively.

1.1.2 Recursive Filtering

Given a sequence of measurements $\underline{\hat{z}}_{0:n} = \big(\underline{\hat{z}}_0, \underline{\hat{z}}_1, \ldots, \underline{\hat{z}}_K\big)$, the estimation problem consists of determining an estimate of the system state $\underline{\boldsymbol{x}}_k$ based on $\underline{\hat{z}}_{0:K}$. Depending on the relation between the time steps k and K, three particular estimation tasks exist: if $k > K$, the estimation problem is called *prediction*, for $k = K$ it is called *filtering* or *measurement update*, and if $k < K$, it is called *smoothing*. Predictions and measurement updates are typically performed on-line, while smoothing is an off-line estimation task, as it aims for improving of past state estimates given additional information.

According to Bayesian filtering theory, the result of each of the three estimation tasks is given in form of a conditional probability density that represents the state estimate. Assuming that the system state $\underline{\boldsymbol{x}}_k$ is a Markov process[4], these densities can be calculated in a recursive fashion commencing from a initial state density $f_0^x\big(\underline{x}_0\big)$ at time step $k = 0$.

4 The state $\underline{\boldsymbol{x}}_k$ merely depends on the previous state $\underline{\boldsymbol{x}}_{k-1}$, but not on older states $\underline{\boldsymbol{x}}_l$ with $l < k-1$. This assumption automatically holds if the noise processes are white [92].

Prediction

In practical applications it is common that prediction and measurement update are performed alternatingly. Thus, without loss of generality, only the one-step prediction is considered here. Predictions over multiple time steps can be achieved by recursively performing one-step predictions.

Given the conditional density $f_k^e(\underline{x}_k) \triangleq f_k^x(\underline{x}_k | \underline{\hat{z}}_{0:k}, \underline{u}_{0:k-1})$ of the previous measurement update, the density of the predicted state for time step $k+1$ is calculated according to the so-called *Chapman-Kolmogorov equation* [92, 166]

$$f_{k+1}^p(\underline{x}_{k+1}) \triangleq f_{k+1}^x(\underline{x}_{k+1} | \underline{\hat{z}}_{0:k}, \underline{u}_{0:k}) = \int f(\underline{x}_{k+1} | \underline{x}_k, \underline{u}_k) \cdot f_k^e(\underline{x}_k) \, \mathrm{d}\underline{x}_k \, . \tag{1.5}$$

Here, the conditional density $f(\underline{x}_{k+1} | \underline{x}_k, \underline{u}_k)$ is the *transition density*, which depends on the dynamic model (1.2) and the system noise $\underline{\boldsymbol{w}}_k$. For the additive noise case (1.4), the transition density is given explicitly by

$$f(\underline{x}_{k+1} | \underline{x}_k, \underline{u}_k) = f_k^w\left(\underline{x}_{k+1} - \underline{a}_k(\underline{x}_k, \underline{u}_k)\right) \, , \tag{1.6}$$

where $f_k^w(\underline{w}_k)$ is the density of the system noise $\underline{\boldsymbol{w}}_k$.

Measurement Update

The measurement update incorporates the current measurement vector $\underline{\hat{z}}_k$ into the predicted density $f_k^p(\underline{x}_k)$. By employing Bayes' theorem, the measurement update results in the *posterior density* of the state given by

$$f_k^e(\underline{x}_k) = c_k \cdot f(\underline{\hat{z}}_k | \underline{x}_k) \cdot f_k^p(\underline{x}_k) \tag{1.7}$$

with normalization constant $c_k \triangleq 1/\int f(\underline{\hat{z}}_k | \underline{x}_k) \cdot f_k^p(\underline{x}_k) \, \mathrm{d}\underline{x}_k$. The term $f(\underline{\hat{z}}_k | \underline{x}_k)$ is known as the *likelihood* of the measurement $\underline{\hat{z}}_k$ and depends on the measurement model (1.3) and the measurement noise $\underline{\boldsymbol{v}}_k$. For the additive noise case (1.4), the likelihood can be expressed explicitly via

$$f(\underline{\hat{z}}_k | \underline{x}_k) = f_k^v\left(\underline{\hat{z}}_k - \underline{h}_k(\underline{x}_k)\right) \tag{1.8}$$

with $f_k^v(\underline{v}_k)$ being the density of the measurement noise $\underline{\boldsymbol{v}}_k$.

In [215] it is shown that the measurement update (1.7) is optimal from an information processing perspective, i.e., there is no loss or waste of information.

Smoothing

While the measurement update utilizes the current measurement for calculating the estimate, smoothing additionally incorporates future measurements. This restricts the application of smoothing for non-real time tasks, but the resulting estimate is more accurate as more information is used. The *smoothed* density $f_k^s(\underline{x}_k) \triangleq f_k^x(\underline{x}_k | \hat{\underline{z}}_{0:K}, \underline{u}_{0:K-1})$ of the state for any time step $k < K$ is calculated according the backward recursion[5]

$$f_k^s(\underline{x}_k) = f_k^e(\underline{x}_k) \cdot \int \frac{f(\underline{x}_{k+1} | \underline{x}_k, \underline{u}_k) \cdot f_{k+1}^s(\underline{x}_{k+1})}{f_{k+1}^p(\underline{x}_{k+1})} \, \mathrm{d}\underline{x}_{k+1} \tag{1.9}$$

commencing from the posterior density $f_K^s(\underline{x}_K) \equiv f_K^e(\underline{x}_K)$. All ingredients in (1.9) are already provided from the prediction and measurement update.

1.1.3 Closed-form Calculation

Estimating the latent state from a sequence of noisy measurements can be considered a statistical inverse problem (see for instance [41]), to which Bayesian filtering provides an optimal solution by means of (1.5), (1.7), and (1.9). The resulting conditional density functions of the system state form a basis for calculating further statistics like the mean vector or the covariance matrix, which are of practical use. The optimal solution, however, is of conceptional value only, mainly for two reasons. Although the recursive nature of the Bayesian filtering equations avoids calculating a joint distribution comprising the system states of all time steps, the resulting conditional densities cannot be represented by means of a finite number of parameters in general [26]. Furthermore, a closed-form solution of the filtering equations is not possible in general due to the involved integrals and multiplications of density functions.

Only for a few special cases, closed-form solutions can be found. An exception for instance exists for linear models affected by Gaussian noise, where the famous *Kalman filter* ([96] and Section 2.2.2) is optimal with closed-form expressions. In case of a finite state space, the *Wonham filter* [211] provides a closed-form solution. For general nonlinear models (1.2) and (1.3), however, an approximation of the optimal Bayesian solution is inevitable to obtain a feasible filter for practical

5 This recursion actually holds for *fixed-interval* smoothing. Fixed-lag and fixed-point smoothing can be directly derived from it.

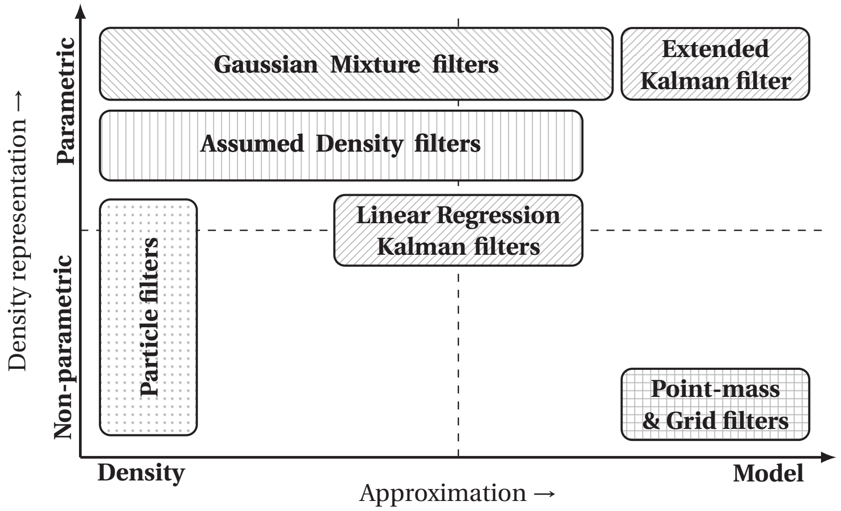


Figure 1.2: Popular approximate nonlinear filtering approaches.

applications. The following section gives a brief overview of major streams in approximate Bayesian filtering.

1.1.4 Approximate Filtering: State of the Art

In Figure 1.2, different groups of popular approximate filtering approaches are depicted. To show the differences between these groups, they are arranged regarding two approximation aspects:

1. The filtering approaches approximate the given nonlinear dynamic and measurement *models* or they approximate directly the *conditional densities* resulting from the optimal Bayesian filter.
2. For approximation purposes, a *parametric* (i.e., a fixed functional type) or *non-parametric* density representation is used.

Approximation of Models

One of mostly employed approximate nonlinear filtering approaches is the extended Kalman filter (EKF), which relies on a first-order Taylor-series expansion of the dynamic and measurement models [92, 173]. The Taylor-series expansion results in linear models, for which the Kalman filter equations can be employed. The major strengths of this approach are its simplicity and computational efficiency. Due to the linearization, the EKF is applicable only for mild nonlinearities. The EKF is described in more detail in Section 2.2.3.

Instead of a linearization, point-mass and grid filters [14, 33, 108, 172] utilize a discretization of the models, which leads to discrete version of Bayesian filtering problem. In doing so, all integrals in (1.5), (1.7), and (1.9) become summations, which are straightforward to evaluate. Point-mass and grid filters suffer from the so-called *curse of dimensionality* [16], as the number of discrete states increases exponentially with the dimension of the state space. Thus, these filtering methods are only feasible for low-dimensional problems.

Non-parametric Density Representation

Figure 1.2 indicates that the majority of the filters rely on an approximation of the density instead of the models. One explanation for this imbalance can be found in [95], where the authors state:

> *"It is easier to approximate a probability distribution than it is to approximate an arbitrary nonlinear function or transformation."*

Particle filters (PFs, see Appendix A) for instance utilize a weighted sample representation of the conditional densities—the so-called *particles*. In contrast to the point-mass and grid filters, where the discretization is performed deterministically, the particles are drawn randomly. This discrete density representation simplifies Bayesian filtering significantly, as prediction is simply performed by propagating the samples through the dynamic model (1.2), while measurement update and smoothing essentially boil down to adapting the particle weights. This simplicity is one of the key factors, why particle filters a very popular. Furthermore, PFs make no strong assumptions on the conditional densities. PFs form a Monte Carlo approximation, for which convergence towards the optimal Bayesian filtering solution with an increasing number of particles can be proven [58].

Although the convergence analysis is independent of the state dimension, practice shows that also PFs suffer from the curse of dimensionality [48]. This can be explained by the fact that with an growing state dimension also the volume to be filled with particles growth exponentially. An additional problem with PFs is sample depletion, i.e., over time most of the particles have zero-weight, which limits the representation of multi-model densities. As countermeasure resampling has to be employed. Also the choice of an appropriate proposal density from which the samples are drawn is critical.

In the recent years, modified PF algorithms have been proposed, which are not relying on a pure sample representation. To attenuate the aforementioned problems of PFs, these algorithms temporarily or even completely employ continuous densities like Gaussian densities [105], hybrid sample-Gaussian representations [200], or various mixture densities [1, 106, 128].

Parametric Density Representation

Thanks to their universal approximator property [121], *Gaussian mixture* densities are a welcome choice for approximating the conditional densities of the Bayesian filter. They provide an analytical and continuous representation that theoretically can approach the true density arbitrarily well, depending on the number of mixture components (see Section 3.1 for detailed introduction to Gaussian mixtures). Relevant statistics like mean or covariance can be derived in closed-form. In contrast to a single Gaussian density, determining the optimal parameters of a Gaussian mixture typically requires solving very demanding optimization problems. These optimization problems can be performed on-line in order to directly approximate the true conditional densities [75] or off-line, by replacing the transition density and likelihood by Gaussian mixtures [83, 86]. The latter case corresponds to the class of model approximating filters.

A more light-weight class of Gaussian mixture filters utilize multiple Gaussian filters like the EKF or linear regression Kalman filters (see next section) simultaneously, i.e., for each component of the mixture, a Gaussian filter is employed (see for example [7, 170] and Section 3). These filters combine the benefits of both worlds: the simplicity of Gaussian filters and the approximation power of Gaussian mixtures, but without sophisticated parameter optimization.

Alternatively to fitting the approximate density directly to the true density, *assumed density filters* (ADFs) calculate the approximate density in such a way that the moments of the true density are preserved—which is known as *moment matching*. This approach at least guarantees the correctness of some important statistics like mean and covariance, which might not be the case when directly approximating the density. Furthermore, for some nonlinear filtering problems the true conditional densities cannot be expressed analytically, but the moments are available in closed form (see for instance Section 2.5.2). For most filtering problems, however, calculating the desired moments requires numerical integration, which is computationally demanding especially for high-dimensional states. Typical density representations employed in ADFs are Gaussians [66, 120], Edgeworth/Gram-Charlier series [38, 181, 182], or exponential densities [27, 29].

Linear Regression Kalman Filters

Basically, linear regression Kalman filters (LRKFs) like the famous unscented Kalman filter [95, 205] calculate a Gaussian approximation of the true conditional densities, whereas the parameters of the Gaussian density are obtained by propagating samples through the nonlinear models (1.2) and (1.3). In contrast to PFs, these samples are chosen in a deterministic fashion and capture the mean and covariance of the prior density exactly. Although this deterministic sampling clearly refers to a density approximation approach, there also exists an alternative interpretation [112, 113]: the same Gaussian density can be obtained by means of stochastic linearization of the models through the use of statistical linear regression (a theoretical treatment can be found in Section 2.2.5).

In contrast to the EKF, LRKFs provide more accurate estimates, while the computational complexity is almost the same. Furthermore, LRKFs are applicable to a larger class of filtering problems, as no differentiability of the system and measurement functions is required.

1.2 Research Topics

Especially with the advent of the LRKFs—besides the particle filters—significant improvements in approximating the optimal Bayesian filter have been achieved in the recent years. Motivated by the benefits of these Gaussian filters, in this thesis three major research topics grouped around *nonlinear Gaussian filtering* are covered. At first, further improvements of Gaussian filtering in general and LRKFs in particular are proposed. These improvements are then used for the two other research topics: filtering via Gaussian mixtures and Gaussian process models. In the following, for each of the three topics, particular reasons are identified why dealing with Gaussians, Gaussian mixtures, and Gaussian processes in the context of Bayesian filtering is reasonable and thus, worth for further investigation. Also limitations of state-of-the-art approaches are exposed, which are resolved in this thesis.

Why Gaussian Filtering?

Besides its optimality, a major reason of the wide application of the Kalman filter is its simplicity; all calculations are performed on the basis of matrix calculus (see Section 2.2.2). The EKF and LRKFs leverage the elegant Kalman filter equations for nonlinear filtering problems by assuming that the optimal conditional densities can be sufficiently well approximated by Gaussians. Additionally, these filters do not suffer from the curse of dimensionality as the number of parameters of a Gaussian density, i.e., the number elements of the mean vector and covariance matrix, merely grow quadratically with the state dimension. This is different for many other approaches like PFs. Gaussian filters are especially useful in applications where limited computational demand and memory usage is key, but an essential consideration of uncertainty is necessary.

Current Gaussian filters are typically applied in a black-box fashion, i.e., without a detailed analysis of the properties of the given dynamic and measurement models. Hence, approximations are applied even in cases, where at least some parts of the models do not require an approximate treatment. Furthermore, the Gaussian assumption is not only applied for the filtering results, it is also used for representing the joint distribution of state and measurement. This assumption is necessary for exploiting the Kalman filtering equations, but for many applications it is too strong and limits the quality of the approximation.

Why Gaussian Mixture Filtering?

Obviously, employing Gaussian densities to approximate the optimal Bayesian results may not be sufficient for every filtering problem[6]. Especially in cases, where the resulting densities are multimodal, heavily skewed, or have heavy tails, extending Gaussian filters towards using Gaussian mixture densities is desirable. Here, Gaussian filters benefit from the fact that this extension is straightforward to obtain as they can be applied on each mixture component independently. With an increasing number of components, it can be shown that Gaussian mixture filters utilizing EKF or LRKFs converge towards the optimal result [4].

One critical part of Gaussian mixture filters is the increase of the number of components. New components should only be introduced where the current ap-

6 In [190], it is illustrated that many natural and technical phenomena generate Gaussian distributions, but especially sociological systems—these also include financial systems—cannot be described with Gaussian statistics.

proximation is not accurate enough. To limit the computation time and memory consumption, it is also necessary the reduce the number of components from time to time, especially when the number of components is disproportionate to the complexity of the density's shape.

Why Gaussian Process Filtering?

So far it was assumed that the dynamic and measurement models are known. This assumption does not hold in applications, where it is too complicated or even impossible to derive these models. These issues for instance may arise, when the underlying real system consists of many interacting elements like in robotics [130, 187], when the models cannot be calibrated sufficiently well like in WiFi-based localization [64] or in calibrating metal oxide sensors [124], or when the mapping between state and measurement is artificial like in classification problems [87]. Here, the mathematical models can be substituted by means of so-called *Gaussian process* (GP) models, which are non-parametric probabilistic models learned from data and which are a popular tool in machine learning (see Section 4.1 for a brief introduction to GPs).

Bayesian filtering with GP models so far has only been performed approximately. Furthermore, due to the non-parametric nature of GPs, the model accuracy but unfortunately also the model complexity increases with the size of data set used for learning. Performing Bayesian filtering can become infeasible if the data set grows over time.

1.3 Main Contributions

The research areas and questions posed in the previous section are covered by edited versions of the papers forming the second part of this thesis. The contributions of these 15 papers are summarized in this section. The papers for each research area can be divided into two groups: one group proposes *theoretical findings* and *algorithms* derived on the basis of these findings, while the other group investigates a dedicated practical *application*. In Figure 1.3, the papers contained in each of the three research areas as well as the dependencies between the research areas are depicted.

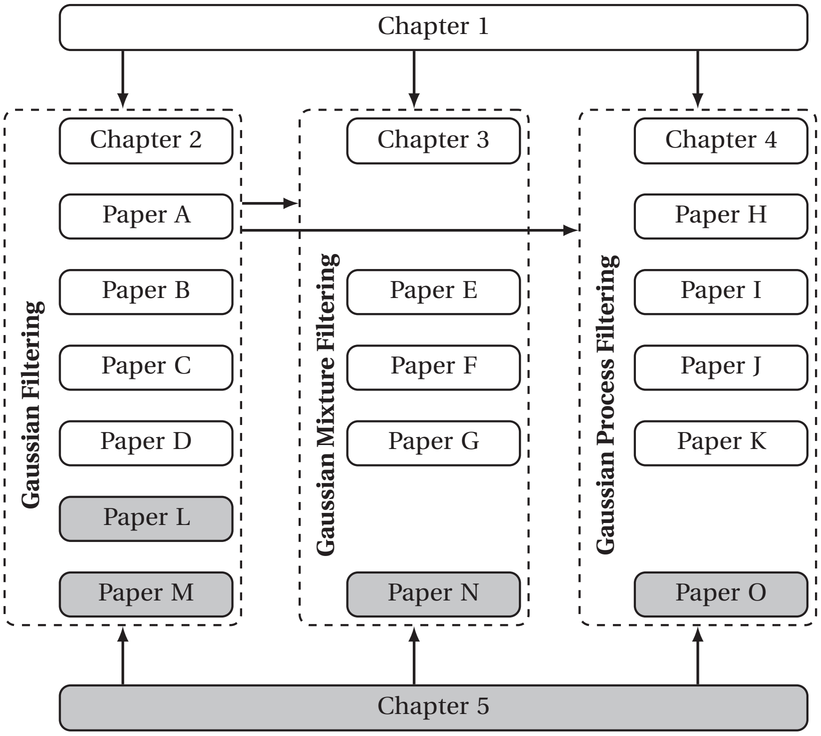


Figure 1.3: Structure of the thesis. Gray boxes indicate applications.

1.3.1 Gaussian Filtering

Gaussian Filtering using State Decomposition Methods

In Paper A,

> F. Beutler, M. F. Huber, and U. D. Hanebeck. Gaussian Filtering using State Decomposition Methods. In *Proceedings of the 12th International Conference on Information Fusion (FUSION)*, pages 579–586, Seattle, WA, USA, July 2009,

LRKFs are extended in such a way that the number of samples used can be reduced significantly by exploiting special structures in both the dynamic model and measurement model. For this purpose, the state vector is decomposed in two ways: First, it is exploited that only some parts of the state vector are observable by measurements. Second, the models are decomposed in linear and nonlinear parts, where merely the nonlinear part is treated approximately by means of LRKFs. It shown by means of simulations and experiments that the estimation performance of the decomposed filters is comparable to the full-state ones, but the computation time is significantly reduced.

Semi-Analytic Gaussian Assumed Density Filter

The findings of Paper A are extended in Paper B,

> Marco F. Huber, Frederik Beutler, and Uwe D. Hanebeck. Semi-Analytic Gaussian Assumed Density Filter. In *Proceedings of the 2011 American Control Conference (ACC)*, pages 3006–3011, San Francisco, CA, USA, June 2011.

Instead of merely decomposing the models into linear and nonlinear substructures, a nonlinear-nonlinear decomposition is proposed, where one nonlinear part is conditionally integrable in closed form. This property holds only for special nonlinearities like polynomials, trigonometric functions, or squared exponential functions. For the conditionally integrable nonlinear part, mean vector and covariance matrix can be calculated analytically, if the filtering result is assumed to be Gaussian distribution. The other nonlinear part is still approximated via LRKFs. Simulations show an improved filtering accuracy and reduced computational demand.

Chebyshev Polynomial Kalman Filter

For polynomial nonlinearities closed-form moment propagation is possible, a property that was also exploited in Paper B. Paper C,

> M. F. Huber. Chebyshev Polynomial Kalman Filter. In *Digital Signal Processing*, vol. 23, no. 5, pages 1620–1629, September 2013,

leverages this property for arbitrary nonlinear systems. Here, these systems first are approximated by means of a Chebyshev polynomial series. The special structure of these orthogonal polynomials is then exploited for deriving very efficient closed-form vector-matrix expressions for mean and variance propagation. The superior performance of the resulting filter over the state-of-the-art is demonstrated via simulations and a real-world application.

Gaussian Filtering for Polynomial Systems Based on Moment Homotopy

All the algorithms proposed in Paper A–Paper C still rely on the assumption that state and measurement are joint Gaussian distributed. This assumption is relaxed in Paper D,

> M. F. Huber and U. D. Hanebeck. Gaussian Filtering for Polynomial Systems Based on Moment Homotopy. In *Proceedings to the 16th International Conference on Information Fusion (FUSION)*, pages 1080–1087, Istanbul, Turkey, July 2013,

for polynomial nonlinearities. This relaxation offers the opportunity of exactly determining the posterior mean and variance. However, a closed-form calculation is not possible and thus, a novel homotopy continuation method is proposed, which yields almost exact posterior mean and variance.

Application: Range-Based Localization

As application scenario for Gaussian filtering *range-based localization* is considered in Paper L,

> F. Beutler, M. F. Huber, and U. D. Hanebeck. Optimal Stochastic Linearization for Range-Based Localization. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5731–5736, Taipei, Taiwan, October 2010,

and in Paper M,

> F. Beutler, M. F. Huber, and U. D. Hanebeck. Semi-Analytic Stochastic Linearization for Range-Based Pose Tracking. In *Proceedings of the 2010 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 44–49, Salt Lake City, UT, USA, September 2010.

In Paper L merely position and velocity of an object are estimated based on range measurements, which allows Gaussian assumed density filtering with closed-form calculation of mean and covariance. The orientation of an object is additionally considered in Paper M, which no longer allows an analytical solution. Instead, the nonlinear-nonlinear decomposition proposed in Paper B has to be employed.

1.3.2 Gaussian Mixture Filtering

(Semi-)Analytic Gaussian Mixture Filter

Closed-form mean and covariance calculation for special nonlinearities as well as the nonlinear-nonlinear model decomposition are extended in Paper E,

> M. F. Huber, F. Beutler, and U. D. Hanebeck. (Semi-)Analytic Gaussian Mixture Filter. In *Proceedings of the 18th IFAC World Congress*, pages 10014–10020, Milano, Italy, August 2011,

for Gaussian mixture filters. Both techniques can be employed component-wise and the superiority over state-of-the-art Gaussian mixture filters is demonstrated by means of simulations.

Adaptive Gaussian Mixture Filter Based on Statistical Linearization

Typically, Gaussian mixture filters based on the EKF or LRKFs assume a fixed number of mixture components, but determining the appropriate number requires to trade filtering performance off against computational load. In Paper F,

> M. F. Huber. Adaptive Gaussian Mixture Filter Based on Statistical Linearization. In *Proceedings of the 14th International Conference on Information Fusion (Fusion)*, Chicago, Illinois, July 2011,

an adaptive algorithm is proposed, which introduces new mixture components via splitting existing components. Splitting is performed whenever the nonlinearity of the dynamic or measurement model is high, but the current number of components is too low and thus, large (statistical) linearization errors are caused. Simulations show that new components are actually introduced where needed, while splitting at mild nonlinear or even linear parts of the models are avoided.

Superficial Gaussian Mixture Reduction

Due to adaptive splitting as in Paper F or due to the multiplication of Gaussian mixtures—which occurs if the transition density (1.6) or the likelihood (1.8) are also Gaussian mixtures—the number of mixture components grows unbounded. To limit this growth, in Paper G,

M. F. Huber, P. Krauthausen, and U. D. Hanebeck. Superficial Gaussian Mixture Reduction. In *INFORMATIK 2011 - the 41th Annual Conference of the Gesellschaft für Informatik e.V. (GI), 6th Workshop Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, Berlin, Germany, October 2011,

a reduction algorithm is proposed. It minimizes an upper bound of the curvature of the Gaussian mixture. This minimization can be formulated as a quadratic program that optimizes the component weights. The mixture is then reduced by removing component with zero weight. The advantages are an automated determination of the necessary number of components and a computational efficient implementation thanks to the plethora of solvers for quadratic programs.

Application: Gas Dispersion Source Estimation

The accurate and timely estimation of the location and strength of a gas release into atmosphere is imperative in order to increase the effectiveness of counter measures for protecting the public. In Paper N,

M. F. Huber. On-line Dispersion Source Estimation using Adaptive Gaussian Mixture Filter. In Proceedings of the *19th IFAC World Congress*, pages 1059–1066, Cape Town, South Africa, August 2014,

the adaptive Gaussian mixture filter proposed in Paper F is applied to this parameter estimation problem. In contrast to the Monte Carlo methods commonly used in this field, the proposed filter allows on-line estimation of the source parameters, while at the same time the estimation error is comparable or even lower than the state-of-the-art.

1.3.3 Gaussian Process Filtering

Analytic Moment-based Gaussian Process Filtering

Assuming that both the dynamic model and the measurement model are represented by means of Gaussian processes, Paper H,

M. P. Deisenroth, M. F. Huber, and U. D. Hanebeck. Analytic Moment-based Gaussian Process Filtering. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, Montreal, Canada, June 2009, derives closed-form expressions for prediction and measurement update. Besides the joint Gaussian assumption of state and measurement, no additional approximations are employed.

Robust Filtering and Smoothing with Gaussian Processes

The results of Paper H are extended in Paper I,

> M. P. Deisenroth, R. D. Turner, M. F. Huber, U. D. Hanebeck, and C. E. Rasmussen. Robust Filtering and Smoothing with Gaussian Processes. In *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pages 1865–1871, July 2012,

for smoothing. Assuming GPs with squared exponential covariance function and zero mean function, mean and covariance are derived analytically exactly and in closed form. It is further shown that restricting to this particular type of GP is not too restrictive as it corresponds to a universal function approximator.

Recursive Gaussian Process Regression

GPs are not well suited for applications, where data arrives during runtime. In Paper J,

> M. F. Huber. Recursive Gaussian Process Regression. In *Proceedings of the 38th International Conference on Acoustics, Sound, and Signal Processing (ICASSP)*, pages 3362–3366, Vancouver, BC, Canada, May 2013,

an on-line GP regression algorithm is proposed, which determines GP models recursively with a constant computation time. The key idea is to use a fixed-size set of so-called basis vectors that store all information necessary for regression. It is shown via synthetic and real-world data that this novel GP regression performs better than other on-line approaches. The resulting GP models can be utilized in the GP filtering algorithms proposed in Paper H and Paper I.

Recursive Gaussian Process: On-line Regression and Learning

In Paper J it is assumed that the hyperparameters of the GPs are known. This assumption is not valid in many applications. Alternatively, one has to learn the hyperparameters from data. Paper K,

M. F. Huber. Recursive Gaussian Process: On-line Regression and Learning. *Pattern Recognition Letters*, vol. 45, pages 85–91, August 2014,

extends the recursive GP regression of Paper J with an on-line hyperparameter learning capability, i.e., regression and learning are performed simultaneously and during runtime. For this purpose, regression and learning are formulated as a Bayesian filtering problem, where the hyperparameters introduce nonlinearities. The nonlinear-nonlinear decomposition technique proposed in Paper B is employed to allow closed-form estimation of the hyperparameters.

Application: Bayesian Active Object Recognition via Gaussian Process Regression

In active object recognition, camera parameters like zoom or orientation are adapted to improve object classification performance. Due to the abstract nature of the mapping from object class—which corresponds to the system state—and image features, in Paper O,

M. F. Huber, T. Dencker, M. Roschani, and J. Beyerer. Bayesian Active Object Recognition via Gaussian Process Regression. In *Proceedings of the 15th International Conference on Information Fusion (Fusion)*, pages 1718–1725, Singapore, July 2012,

this mapping is learned from data and represented by means of a GP. The classification task itself is formulated as a Bayesian filtering problem, for which closed-form expressions are derived. The appropriate camera parameters result from solving a sequential optimization problem that maximizes the mutual information between object class and image features.

1.4 Thesis Outline

This thesis consists of two parts: The first provides the background that is necessary for understanding the second part, which comprises the edited versions of the aforementioned papers. The first part also provides summaries of the results of these papers. Please note that the background material and the summaries are kept to a minimum in order to avoid unnecessary redundancy.

The structure of the thesis is depicted in Figure 1.3 on page 15. Chapter 2 briefly introduces the Gaussian density function and its properties. Also state-of-the-art

Gaussian filters like the Kalman filter, LRKFs, or moment matching are described and summaries of the papers A–D are provided.

Gaussian mixture filtering is the content of Chapter 3. Besides the extension of Gaussian filters to Gaussian mixtures, this chapter also introduces the problem of increasing and reducing the number of mixture components. The chapter closes with summaries of the papers E–G.

In Chapter 4, Gaussian process regression and the state-of-the-art of Bayesian filtering with GP models is introduced. Furthermore, the complexity issues with GPs are discussed. This chapter comprises the summaries of the papers H–K.

Chapter 5 provides an introduction to each application treated with the algorithms developed in the papers A–K. It further summarizes the results of the papers L–O.

The first part of the thesis closes with Chapter 6, which gives a conclusion and an outlook to future work.

2 Gaussian Filtering

This chapter lays the foundation of the thesis. It first introduces the Gaussian distribution and some of its important properties in Section 2.1. Based on this, a general Gaussian filtering problem with focus on predictions and measurement updates is formulated in Section 2.2.1. This general problem can only be solved for some special cases, where the linear is the most prominent one (see Section 2.2.2). Otherwise, approximations have to be applied, where Sections 2.2.3–2.2.5 discuss the mostly utilized approximation techniques. In Section 2.3, the smoothing problem is formulated. The same approximation techniques introduced for prediction and measurement update can be applied here as well.

Many filtering problems, even though being nonlinear, comprise linear substructures. This special situation has been exploited in the past by means of the so-called Rao-Blackwellisation theorem. Section 2.4 provides a brief summary of this theorem and its application to Gaussian filtering. This theorem is also the starting point for the contribution made in this thesis for Gaussian filtering. The key findings of the Papers A–D are summarized in Section 2.5.

2.1 The Gaussian Distribution

The central distribution employed in this thesis is the Gaussian or normal distribution. It is also the most widely used distribution in statistics, filtering, and machine learning. Its probability density function is defined by

$$\mathcal{N}\left(\underline{x};\underline{\mu},\mathbf{C}\right) \triangleq \frac{1}{\sqrt{|2\pi\mathbf{C}|}} \mathrm{e}^{-\frac{1}{2}\left(\underline{x}-\underline{\mu}\right)^{\mathrm{T}}\mathbf{C}^{-1}\left(\underline{x}-\underline{\mu}\right)} \tag{2.1}$$

with the parameters mean vector and covariance matrix

$$\begin{aligned}\underline{\mu} &= \mathrm{E}\left\{\underline{\boldsymbol{x}}\right\} = \int_{\mathbb{R}^{n_x}} \underline{x} \cdot \mathcal{N}\left(\underline{x};\underline{\mu},\mathbf{C}\right) \mathrm{d}\underline{x} \ ,\\ \mathbf{C} &= \mathrm{Cov}\left\{\underline{\boldsymbol{x}}\right\} = \mathrm{E}\left\{\left(\underline{x}-\underline{\mu}\right)\left(\underline{x}-\underline{\mu}\right)^{\mathrm{T}}\right\} \ ,\end{aligned}$$

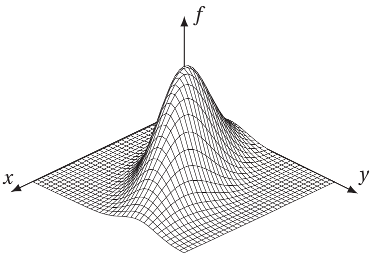
respectively. The dimension of the mean vector corresponds to the dimension of $\underline{\boldsymbol{x}}$, which is n_x. The covariance matrix is symmetric and positive semi-definite, where the number of elements is the dimension of $\underline{\boldsymbol{x}}$ squared. The term $\sqrt{|2\pi\mathbf{C}|}$ in (2.1) is a normalization factor ensuring that the integral of the density is equal to one. If a random vector $\underline{\boldsymbol{x}}$ is Gaussian distributed, the term $\underline{\boldsymbol{x}} \sim \mathcal{N}\left(\underline{\mu},\mathbf{C}\right)$ expresses that $\underline{\boldsymbol{x}}$ is Gaussian distributed with parameters $\underline{\mu}$ and $\mathbf{C}$, where $\mathcal{N}\left(\underline{\mu},\mathbf{C}\right)$ is the (cumulative) Gaussian distribution function. If $\boldsymbol{x} \sim \mathcal{N}(0,1)$, $\boldsymbol{x}$ follows the *standard Gaussian* distribution.

In Figure 2.1, the density functions of various Gaussian random variables are depicted. It can be seen that the Gaussian density has only one single mode that is located at the mean $\underline{\mu}$.

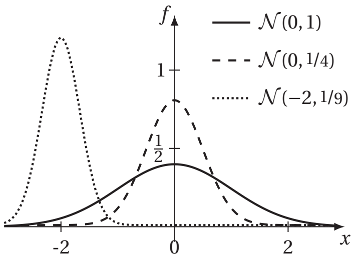
2.1.1 Importance of the Gaussian

The Gaussian distribution is a universal tool in many fields. Besides its application in statistics and filtering as considered in this thesis, it is for example important in statistical mechanics to describe energy fluctuations while in biology it is used to describe population dynamics ([91], Ch. 7). Also in economics it is the fundamental distribution, even though here the justification of applicability is sometimes questionable [190].

The success of the Gaussian distribution has many reasons, where the most important ones are the following [91, 100, 127]: First, it has merely two parameters



(a) Density function of a bivariate Gaussian.



(b) Density functions of different univariate Gaussians. The solid curve corresponds to the standard Gaussian.

Figure 2.1: Various univariate and bivariate Gaussian densities. Characteristic of Gaussian density functions is its *bell shape*.

and these are easy to interpret. These parameters at the same time correspond to the first two moments, which are the most basic properties of a distribution. Second, the number of parameters scales merely quadratically with the dimension of the state space. Third, if merely the first two moments are given, the Gaussian distribution makes the least assumptions about the true distribution according to the *maximum entropy principle* [19] and it is closest to the true distribution if the *Kullback-Leibler divergence* is considered as deviation measure [45, 73]. In parameter estimation it minimizes the *Fisher information.* Forth, it has a simple mathematical form, which allows closed-form calculations in many filtering problems and results in straightforward implementations. Fifth, according to the *central limit theorem* the sum of independent random variables approaches a Gaussian distribution, which makes the Gaussian a preferred choice for modeling residual errors and noise. Sixth, Gaussians are unimodal and thus, they possess a single maximum. Such distributions are typical in many filtering problems like single-target tracking [196].

2.1.2 Dirac Delta Distribution

The Gaussian distribution possesses an important limiting case. If the determinant of the covariance matrix approaches zero, i.e., if $|\mathbf{C}| \to 0$, the Gaussian becomes a “peak" centered around the mean. This distribution is known as the *Dirac delta distribution* given by

$$\delta\big(\underline{x}-\underline{\mu}\big) = \begin{cases} \text{undefined} & \text{if } \underline{x} = \underline{\mu} \\ 0 & \text{if } \underline{x} \neq \underline{\mu} \end{cases}$$

such that

$$\int_{\mathbb{R}^{n_x}} \delta\big(\underline{x}\big)\,\mathrm{d}\underline{x} = 1\ .$$

The value of the Dirac delta is zero everywhere except of $\underline{x} = \underline{\mu}$. A useful property that follows from this fact is the so-called *sifting property*

$$\int_{\mathbb{R}^{n_x}} g\big(\underline{x}\big)\cdot\delta\big(\underline{x}-\underline{\mu}\big)\,\mathrm{d}\underline{x} = g\big(\underline{\mu}\big) \tag{2.2}$$

that selects only a single value from an integration.

2.1.3 The Exponential Family

The Gaussian density itself is a special case of two families of very general density functions: the Gaussian mixture densities and the exponential family. While the first family is treated in more detail in Chapter 3, a brief introduction to the second family is provided here.

A density function belongs to the *exponential family* if it is of the form

$$f\big(\underline{x}\big) = c\big(\underline{\eta}\big)\cdot h\big(\underline{x}\big)\cdot \exp\Big(\underline{\eta}^{\mathrm{T}}\cdot\underline{\phi}\big(\underline{x}\big)\Big)\ , \tag{2.3}$$

where $\underline{\eta} \triangleq \left[\eta_0\ \eta_1\ \ldots\ \eta_n\right]^{\mathrm{T}}$ is a parameter vector, $\underline{\phi}\big(\underline{x}\big)$ is a vector of *sufficient statistics*[1] comprising a set of n functions

$$\underline{\phi}\big(\underline{x}\big) \triangleq \left[\phi_0\big(\underline{x}\big)\quad \phi_1\big(\underline{x}\big)\quad \cdots\quad \phi_n\big(\underline{x}\big)\right]^{\mathrm{T}}\ . \tag{2.4}$$

The term $c\big(\underline{\eta}\big)$ is a normalization constant ensuring the probability mass being one and $h\big(\underline{x}\big)$ is a scaling constant, often being one.

1 A statistic $\phi(\mathcal{D})$ is said to be sufficient for data $\mathcal{D}$ if $f(x|\mathcal{D}) = f(x|\phi(\mathcal{D}))$, i.e., the statistic contains all information about the data such that the data can be discarded without loss of information.

The exponential family comprises many well-known distributions as special cases. Examples are the Bernoulli distribution, Poisson distribution, multinomial distribution and of course the Gaussian distribution, which can be obtained from (2.3) by choosing

$$\underline{\eta} = \begin{bmatrix} \mathbf{C}^{-1} \cdot \underline{\mu} \\ -\frac{1}{2}\mathbf{C}^{-1} \end{bmatrix} , \quad \underline{\phi}(\underline{x}) = \begin{bmatrix} \underline{x} \\ \underline{x} \cdot \underline{x}^{\mathrm{T}} \end{bmatrix} , \quad c(\underline{\eta}) = \frac{1}{\sqrt{|2\pi\mathbf{C}|}} \mathrm{e}^{-\frac{1}{2}\underline{\mu}^{\mathrm{T}}\mathbf{C}^{-1}\underline{\mu}} , \quad h(\underline{x}) = 1 \, . \tag{2.5}$$

Furthermore, the exponential family is the only family of distributions with finite dimensional sufficient statistics and for which *conjugate priors* exist, i.e., the prior density has the same form as the likelihood, which simplifies Bayesian filtering significantly. However, except of some special cases like the Gaussian density, determining the moments of an exponential density in closed form is not possible in general [73].

2.2 Exact Gaussian Filtering and Approximations

In order to provide a solution of the general Bayesian filtering problem stated in Section 1.1, it is assumed in the following that the state vector $\underline{\boldsymbol{x}}$ is Gaussian distributed. Thus, the filtering task boils down to calculating the two parameters mean vector and covariance matrix.

2.2.1 General Formulation

Various approaches have been proposed in the past for calculating the mean and covariance in case of arbitrary nonlinear dynamic and measurement models. To provide a unified overview of these approaches, the nonlinear transformation[2]

$$\underline{\boldsymbol{y}} = \underline{g}(\underline{\boldsymbol{x}}) + \underline{\boldsymbol{w}} \, , \quad \underline{\boldsymbol{w}} \sim \mathcal{N}(\underline{0}, \mathbf{C}_w) \, , \tag{2.6}$$

is considered in the following, where the Gaussian state $\underline{\boldsymbol{x}} \sim \mathcal{N}(\underline{\mu}_x, \mathbf{C}_x)$ is mapped to a Gaussian random vector $\underline{\boldsymbol{y}}$ via an arbitrary nonlinear function $\underline{g}(.)$. The noise $\underline{\boldsymbol{w}}$ is independent of $\underline{\boldsymbol{x}}$. For predictions, the transformation $\underline{g}(.)$ corresponds to

2 Only the additive noise case is discussed here. For non-additive noise, the solutions derived next can be directly applied if the state $\underline{\boldsymbol{x}}$ is augmented with the noise $\underline{\boldsymbol{w}}$, i.e., $g(.)$ then becomes a function of both $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{w}}$.

the dynamic model $\underline{a}_k(.,.)$ in (1.4) for a given input $\underline{u}_k$ and $\underline{\boldsymbol{y}}$, $\underline{\boldsymbol{w}}$ are replaced by the predicted state $\underline{\boldsymbol{x}}_{k+1}$ and the noise $\underline{\boldsymbol{w}}_k$, respectively. In case of measurement updates, $\underline{g}(.)$ becomes the measurement model $\underline{h}_k(.)$, while $\underline{\boldsymbol{y}}$ and $\underline{\boldsymbol{w}}$ are replaced by the measurement vector $\underline{\boldsymbol{z}}_k$ and noise $\underline{\boldsymbol{v}}_k$, respectively.

For solving prediction and measurement update, the goal is to determine the joint Gaussian distribution of $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$, which is given by

$$\begin{bmatrix}\underline{\boldsymbol{x}}\\ \underline{\boldsymbol{y}}\end{bmatrix} \sim \mathcal{N}(\underline{\mu}, \mathbf{C}) \quad \text{with } \underline{\mu} \triangleq \begin{bmatrix}\underline{\mu}_x\\ \underline{\mu}_y\end{bmatrix}, \mathbf{C} \triangleq \begin{bmatrix}\mathbf{C}_x & \mathbf{C}_{xy}\\ \mathbf{C}_{xy}^{\mathrm{T}} & \mathbf{C}_y\end{bmatrix}, \tag{2.7}$$

where

$$\begin{aligned}\underline{\mu}_y &= \mathrm{E}\left\{\underline{g}(\underline{\boldsymbol{x}})\right\} = \int \underline{g}(\underline{x}) \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_x, \mathbf{C}_x\right) \mathrm{d}\underline{x}\,, \\ \mathbf{C}_y &= \mathrm{Cov}\left\{\underline{g}(\underline{\boldsymbol{x}})\right\} = \int \left(\underline{g}(\underline{x}) - \underline{\mu}_y\right)\left(\underline{g}(\underline{x}) - \underline{\mu}_y\right)^{\mathrm{T}} \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_x, \mathbf{C}_x\right) \mathrm{d}\underline{x} + \mathbf{C}_w\,, \\ \mathbf{C}_{xy} &= \mathrm{Cov}\left\{\underline{\boldsymbol{x}}, \underline{g}(\underline{\boldsymbol{x}})\right\} = \int \left(\underline{x} - \underline{\mu}_x\right)\left(\underline{g}(\underline{x}) - \underline{\mu}_y\right)^{\mathrm{T}} \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_x, \mathbf{C}_x\right) \mathrm{d}\underline{x}\,,\end{aligned} \tag{2.8}$$

are the unknown mean vector and covariance matrix of $\underline{\boldsymbol{y}}$ as well as the unknown cross-covariance matrix between $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$, respectively.

To perform a prediction, the respective parameters of the predicted Gaussian density can be retrieved from the first two lines of (2.8), namely the predicted mean and covariance. In case of a measurement update, an additional calculation is required as the posterior density is conditioned on the current measurement (see (1.7)). Thus, by conditioning $\underline{\boldsymbol{x}}$ on $\underline{\boldsymbol{y}}$ the resulting density is given by (see e.g. [143], Appendix A)

$$\underline{\boldsymbol{x}} | \underline{\boldsymbol{y}} \sim \mathcal{N}\left(\underline{\mu}_x + \mathbf{C}_{xy}\mathbf{C}_y^{-1} \cdot \left(\underline{y} - \underline{\mu}_y\right), \mathbf{C}_x - \mathbf{C}_{xy}\mathbf{C}_y^{-1}\mathbf{C}_{xy}^{\mathrm{T}}\right), \tag{2.9}$$

which corresponds to desired *posterior density*.

Due to the restriction that both $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$ are assumed to be Gaussian, a Bayesian filter calculating the *exact* joint Gaussian (2.7) is named a *Gaussian assumed density filter*. That is, although the true density is not Gaussian due to the nonlinear transformation (2.6), at least the first two moments—mean vector and covariance matrix—coincide with the true respective moments, why this approach often is also named *moment matching* [120]. Unfortunately, the integrals in (2.8) possess no analytic solution for arbitrary nonlinear functions $\underline{g}(.)$ in general and thus, approximations are inevitable. The various approximations proposed in the past for Gaussian filtering essentially differ in the way the integrals in (2.8) are solved. Just for some special cases, closed-form expressions for (2.8) can be found. The most prominent one is discussed next.

2.2.2 Linear Filtering

Assuming that the transformation in (2.6) is a linear one according to

$$\underline{\boldsymbol{y}} = \mathbf{G} \cdot \underline{\boldsymbol{x}} + \underline{\boldsymbol{w}} \,,$$

where the matrix $\mathbf{G}$ corresponds to the nonlinear function $g(.)$, the moment integrals in (2.8) can be calculated in closed form. The mean vector and covariance matrix of the joint Gaussian are then given by

$$\underline{\mu} = \begin{bmatrix} \underline{\mu}_x \\ \mathbf{G} \cdot \underline{\mu}_x \end{bmatrix} , \quad \mathbf{C} = \begin{bmatrix} \mathbf{C}_x & \mathbf{C}_x \mathbf{G}^{\mathrm{T}} \\ \mathbf{G}\mathbf{C}_x & \mathbf{G}\mathbf{C}_x\mathbf{G}^{\mathrm{T}} + \mathbf{C}_w \end{bmatrix} . \tag{2.10}$$

This follows directly from exploiting the linearity of the expectation operator E{.}. In doing so, the integrals in (2.8) are reduced to calculating the mean and covariance of $\underline{\boldsymbol{x}}$.[3]

The Kalman Filter

Based on (2.10), the famous *Kalman filter* can be derived. For this purpose, linear dynamic and measurement models according to

$$\begin{aligned} \underline{\boldsymbol{x}}_{k+1} &= \mathbf{A}_k \cdot \underline{\boldsymbol{x}}_k + \underline{\boldsymbol{w}}_k \,, \quad & \underline{\boldsymbol{w}}_k &\sim \mathcal{N}\left(\underline{0}, \mathbf{C}_k^w\right) \\ \underline{\boldsymbol{z}}_k &= \mathbf{H}_k \cdot \underline{\boldsymbol{x}}_k + \underline{\boldsymbol{v}}_k \,, \quad & \underline{\boldsymbol{v}}_k &\sim \mathcal{N}\left(\underline{0}, \mathbf{C}_k^v\right) \end{aligned} \tag{2.11}$$

are assumed with system matrix $\mathbf{A}_k$ and measurement matrix $\mathbf{H}_k$. Given the posterior density $f_k^e(\underline{x}_k) = \mathcal{N}\left(\underline{x}_k; \underline{\mu}_k^e, \mathbf{C}_k^e\right)$ of the system state, the *prediction step* yields the predicted density $f_{k+1}^p(\underline{x}_{k+1}) = \mathcal{N}\left(\underline{x}_{k+1}; \underline{\mu}_{k+1}^p, \mathbf{C}_{k+1}^p\right)$ with parameters

$$\begin{aligned} \underline{\mu}_{k+1}^p &= \mathbf{A}_k \cdot \underline{\mu}_k^e \,, \\ \mathbf{C}_{k+1}^p &= \mathbf{A}_k \mathbf{C}_k^e \mathbf{A}_k^{\mathrm{T}} + \mathbf{C}_k^w \,, \end{aligned} \tag{2.12}$$

3 For calculating $\mathbf{C}_{xy}$ the identity $\mathrm{E}\{\underline{\boldsymbol{x}} \cdot \underline{\boldsymbol{x}}^{\mathrm{T}}\} = \mathbf{C}_x + \underline{\mu}_x \underline{\mu}_x^{\mathrm{T}}$ is required in addition.

where both the mean vector $\underline{\mu}_{k+1}^p$ and the covariance matrix $\mathbf{C}_{k+1}^p$ can be extracted from the second row in (2.10) by replacing $\underline{\mu}_x$, $\mathbf{C}_x$, $\mathbf{G}$, $\mathbf{C}_w$ with $\underline{\mu}_k^e$, $\mathbf{C}_k^e$, $\mathbf{A}_k$, $\mathbf{C}_k^w$, respectively.

For determining the *measurement update* of the Kalman filter, it is necessary to exploit the conditioning (2.9) by replacing $\underline{\mu}_x$, $\mathbf{C}_x$, $\mathbf{G}$, and $\mathbf{C}_w$ with $\underline{\mu}_k^p$, $\mathbf{C}_k^p$, $\mathbf{H}_k$, and $\mathbf{C}_v$, respectively. For a given measurement vector $\hat{\underline{z}}_k$, the resulting measurement update of the Kalman filter calculates the Gaussian posterior density $\mathcal{N}\left(\underline{x}_k; \underline{\mu}_k^e, \mathbf{C}_k^e\right)$ of the system state $\underline{\boldsymbol{x}}_k$ with mean vector and covariance matrix according to

$$\begin{aligned}
\underline{\mu}_k^e &= \underline{\mu}_k^p + \mathbf{K}_k \cdot \left(\hat{\underline{z}}_k - \mathbf{H}_k \cdot \underline{\mu}_k^p\right) , \\
\mathbf{C}_k^e &= \mathbf{C}_k^p - \mathbf{K}_k \mathbf{H}_k \mathbf{C}_k^p ,
\end{aligned} \tag{2.13}$$

where $\mathbf{K}_k = \mathbf{C}_k^p \mathbf{H}_k^{\mathrm{T}} \left(\mathbf{H}_k \mathbf{C}_k^p \mathbf{H}_k^{\mathrm{T}} + \mathbf{C}_k^v\right)^{-1}$ is the so-called *Kalman gain*.

It is worth mentioning that R. Kalman used a different derivation in his paper [96]. Instead of the above approach, which is driven from a Bayesian perspective, he exploited orthogonal projections on the vector space spanned by the measurements. Both derivations are equivalent, where the Bayesian one makes the link to nonlinear Gaussian filters more obvious [159].

Route to Nonlinear Approaches

The additional assumption that not only $\underline{\boldsymbol{y}}$ but also the joint density of $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$ is Gaussian as in (2.7) is only satisfied for linear models. However, all nonlinear Gaussian filtering approaches introduced next follow either implicitly or explicitly the same path that was utilized to derive the Kalman filter, i.e., constructing the joint Gaussian of $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$ and conditioning on $\underline{\boldsymbol{y}}$. Thus, they all form a Kalman filter like approximation for nonlinear filtering problems. This approach is justified by the fact that by assuming both $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$ being Gaussian, there *must* exist a linear transformation from $\underline{\boldsymbol{x}}$ to $\underline{\boldsymbol{y}}$ (see e.g. [197]). Hence, by providing an (approximate) solution to the integrals in (2.8), a linear transformation is constructed simultaneously that (implicitly) approximates the original nonlinear transformation (2.6). By explicitly calculating this linear transformation, the above Kalman filter equations can be applied directly. Hence, the focus in the next sections is mainly on how the approximation is achieved, while the actual filtering equations can be directly extracted from the above Kalman filter equations with appropriate substitution of the linear models (2.11).

Both approaches introduced next explicitly provide linear transformations for approximating the nonlinear filtering problem. They belong to the group of model approximating approaches depicted in Figure 1.2 on page 9. The filters in Section 2.2.5 instead can be considered as density approximating as they aim for the integrals in (2.8) in order to determine the joint Gaussian of $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$. A linear model is merely calculated implicitly.

2.2.3 Linearized and Extended Kalman Filter

The motivation behind the linearized Kalman filter is that in case of mild nonlinearities it might be sufficient to linearize (2.6) about a nominal point through a Taylor-series expansion. Let $\underline{\bar{x}}$ be this nominal point. The Taylor-series expansion of the function $g(.)$ is then given by

$$\underline{\boldsymbol{y}} = \underline{g}(\underline{\boldsymbol{x}}) + \underline{\boldsymbol{w}} = \underline{g}(\underline{\bar{x}}) + \mathbf{G}_x(\underline{\bar{x}}) \cdot \Delta\underline{\boldsymbol{x}} + \mathcal{O}_x + \underline{\boldsymbol{w}} \,,$$

where $\Delta\underline{\boldsymbol{x}} \triangleq \underline{\boldsymbol{x}} - \underline{\bar{x}} \sim \mathcal{N}\left(\underline{\mu}_x - \underline{\bar{x}}, \mathbf{C}_x\right)$, $\mathbf{G}_x(\underline{\bar{x}})$ is the Jacobian matrix of $\underline{g}(.)$ according to

$$\mathbf{G}_x(\underline{\bar{x}}) \triangleq \left.\frac{\partial \underline{g}(\underline{x})}{\partial \underline{x}^{\mathrm{T}}}\right|_{\underline{x}=\underline{\bar{x}}} , \tag{2.14}$$

and $\mathcal{O}_x$ is the remainder comprising all higher-order terms of the expansion. A linear approximation of $\underline{g}(.)$ is obtained by neglecting the remainder term, which yields

$$\underline{g}(\underline{\boldsymbol{x}}) \approx \underline{g}(\underline{\bar{x}}) + \mathbf{G}_x(\underline{\bar{x}}) \cdot \Delta\underline{\boldsymbol{x}} \,. \tag{2.15}$$

Depending on the choice of the nominal point, different realizations of a linearized Kalman filter are obtained. In case of the basic linearized Kalman filter as described in [120] the nominal points are chosen in advance. In doing so, the linearized model (2.15) and thus, the Kalman gains and covariance matrices of the Kalman filter can be calculated off-line [209], which is desirable in applications with low computation resources.

A famous variation known as the *extended Kalman filter* (EKF)—the nonlinear filter used in the NASA Apollo program—is obtained when choosing the nominal points as being equal with the current state mean vector. This filter is no longer an off-line filter, but it is more adaptive to the current situation. This choice of nominal points also has implications on the moments of the joint Gaussian (2.7) as $\Delta\underline{\boldsymbol{x}} = \underline{\boldsymbol{x}} - \underline{\mu}_x \sim \mathcal{N}(\underline{0}, \mathbf{C}_x)$ now has zero mean. The mean vector $\underline{\mu}_y$ for instance is independent of the Jacobian $\mathbf{G}_x$ according to

$$\underline{\mu}_y = \mathrm{E}\left\{\underline{g}(\underline{\boldsymbol{x}}) + \underline{\boldsymbol{w}}\right\} \approx \mathrm{E}\left\{\underline{g}\left(\underline{\mu}_x\right) + \mathbf{G}_x\left(\underline{\mu}_x\right)\Delta\underline{\boldsymbol{x}} + \underline{\boldsymbol{w}}\right\} = \underline{g}\left(\underline{\mu}_x\right) ,$$

i.e., for calculating the mean it is sufficient to merely evaluate the nonlinear function $\underline{g}(.)$ at the nominal point $\underline{\mu}_x$.

A major advantage of linearized Kalman filters compared to Gaussian filters introduced below is their simplicity. Given the Jacobian matrix, they directly boil down to a Kalman filter. As a consequence, they are one of the computationally cheapest if not the cheapest Gaussian filters. However, differentiability of the nonlinear function $\underline{g}(.)$ is required, which is not given in every application.

Many improvements of linearized Kalman filters, but especially for the EKF have been suggested. The second-order EKF for instance utilizes a second-order Taylor-series expansion [66]. The iterated EKF performs multiple iterations of the measurement update for the same measurement value in order to linearize not only about the predicted state, but about the most recent estimate. This procedure converges faster to the exact solution than the EKF [92]. In [214] this concept has been generalized from discrete iterations to a differential update.

2.2.4 Statistical Linearization

An alternative to a Taylor-series expansion for explicitly determining a linear model is *statistical linearization* [66], where the nonlinear function $\underline{g}(.)$ is approximated via

$$\underline{g}(\underline{\boldsymbol{x}}) \approx \mathbf{G} \cdot \Delta\underline{\boldsymbol{x}} + \underline{b}$$

with $\Delta\underline{\boldsymbol{x}} \triangleq \underline{\boldsymbol{x}} - \underline{\mu}_x$. Here, the terms $\mathbf{G}$ and $\underline{b}$ are chosen in such a way that the mean squared error

$$\mathrm{E}\left\{\left(\underline{g}(\underline{\boldsymbol{x}}) - \mathbf{G} \cdot \Delta\underline{\boldsymbol{x}} - \underline{b}\right)^{\mathrm{T}}\left(\underline{g}(\underline{\boldsymbol{x}}) - \mathbf{G} \cdot \Delta\underline{\boldsymbol{x}} - \underline{b}\right)\right\} \tag{2.16}$$

is minimized with respect to $\mathbf{G}$ and $\underline{b}$. The solution to (2.16) yields

$$\underline{b} = \mathrm{E}\left\{\underline{g}\left(\underline{\boldsymbol{x}}\right)\right\} , \tag{2.17}$$

$$\mathbf{G} = \mathrm{E}\left\{\underline{g}\left(\underline{\boldsymbol{x}}\right) \cdot \Delta\underline{\boldsymbol{x}}\right\} \mathbf{C}_x^{-1} . \tag{2.18}$$

There is an interesting relation between the linearized Kalman filter and statistical linearization. Assuming that $\underline{g}(.)$ is differentiable, the expectation in (2.18) can be reformulated to (see [159])

$$\mathrm{E}\left\{\underline{g}\left(\underline{\boldsymbol{x}}\right) \cdot \Delta\underline{\boldsymbol{x}}\right\} = \mathrm{E}\left\{\mathbf{G}_x\left(\underline{\boldsymbol{x}}\right)\right\} \mathbf{C}_x ,$$

where $\mathbf{G}_x$ is the Jacobian matrix (2.14). In this case, the statistical linearization becomes a Taylor-series based linearization, except that instead of directly utilizing $\underline{g}(.)$ and the Jacobian $\mathbf{G}_x$, their expected values are employed. This is beneficial in the sense that statistical linearization exploits additional information about the state $\underline{\boldsymbol{x}}$ thanks to the expectation calculation, while for the Taylor-series expansion merely the mean vector $\underline{\mu}_x$ is utilized. This comes at the expense that the expected values (2.17) and (2.18) often cannot be calculated analytically, albeit the Jacobian matrix may exist.

2.2.5 Linear Regression Kalman Filters

To overcome the flaws of statistical and Taylor-series based linearization, the group of so-called *linear regression Kalman filters* (LRKFs) are based on a completely different approach. Instead of directly approximating the nonlinear model (2.6), the Gaussian representing the state $\underline{\boldsymbol{x}}$ is approximated by means of a set of weighted samples $\mathcal{L}_x = \{\omega_i, \mathcal{X}_i\}, i = 1 \ldots L$, which are sometimes also called *sigma points*. Given a sample representation of $\mathcal{N}\left(\underline{x}; \underline{\mu}_x, \mathbf{C}_x\right)$, the efficient evaluation of the integrals in (2.8) is straightforward.

In order to see this, the focus is restricted in the following on the integral

$$\mathrm{E}\left\{\underline{g}\left(\underline{\boldsymbol{x}}\right)\right\} = \int \underline{g}\left(\underline{x}\right) \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_x, \mathbf{C}_x\right) \mathrm{d}\underline{x} . \tag{2.19}$$

The solution method for this integral can be directly applied to all integrals in (2.8). In order to obtain the sample representation independent of the current mean vector and covariance matrix of the Gaussian density, a change of the integration variable is employed for (2.19), which results in

$$\mathrm{E}\left\{\underline{g}(\underline{\boldsymbol{x}})\right\} = \int \underline{g}(\underline{x}) \cdot \mathcal{N}\left(\underline{x};\underline{\mu}_x,\mathbf{C}_x\right) \mathrm{d}\underline{x} = \int \underline{g}\left(\underline{\mu}_x + \sqrt{\mathbf{C}_x} \cdot \underline{\theta}\right) \cdot \mathcal{N}\left(\underline{\theta};\underline{0},\mathbf{I}\right) \mathrm{d}\underline{\theta} \qquad (2.20)$$

with $\sqrt{\mathbf{C}}$ being the matrix square root such that $\mathbf{C} = \sqrt{\mathbf{C}}\left(\sqrt{\mathbf{C}}\right)^{\mathrm{T}}$. This transformation allows determining the sample representation in advance for the multivariate standard Gaussian $\mathcal{N}\left(\underline{\theta};\underline{0},\mathbf{I}\right)$, while the adaptation to the actual Gaussian is performed on-line via the transformation $\underline{\mu}_x + \sqrt{\mathbf{C}_x} \cdot \underline{\theta}$.

With a given sample set $\mathcal{L}_\theta = \{\omega_i, \underline{\theta}_i\}, i = 1 \ldots L$, that approximates the standard Gaussian $\mathcal{N}\left(\underline{\theta};\underline{0},\mathbf{I}\right) \approx \sum_{i=1}^{L} \omega_i \cdot \delta\left(\underline{\theta} - \underline{\theta}_i\right)$ as a sum of weighted Dirac delta distributions, the integral (2.20) can be solved according to

$$\begin{aligned}
\mathrm{E}\left\{\underline{g}(\underline{\boldsymbol{x}})\right\} &= \int \underline{g}\left(\underline{\mu}_x + \sqrt{\mathbf{C}_x} \cdot \underline{\theta}\right) \cdot \mathcal{N}\left(\underline{\theta};\underline{0},\mathbf{I}\right) \mathrm{d}\underline{\theta} \\
&\approx \int \underline{g}\left(\underline{\mu}_x + \sqrt{\mathbf{C}_x} \cdot \underline{\theta}\right) \cdot \left(\sum_{i=1}^{L} \omega_i \cdot \delta\left(\underline{\theta} - \underline{\theta}_i\right)\right) \mathrm{d}\underline{\theta} \\
&= \sum_{i=1}^{L} \omega_i \cdot \underline{g}\Big(\underbrace{\underline{\mu}_x + \sqrt{\mathbf{C}_x} \cdot \underline{\theta}_i}_{\triangleq \mathcal{X}_i}\Big) , \qquad (2.21)
\end{aligned}$$

where the second line follows from substituting the standard Gaussian with its approximate sample representation. The third line follows from exploiting the sifting property (2.2).

The quantities $\mathcal{X}_i$ in (2.21) correspond to the *transformed sample points* with corresponding weights ω_i. The actual value of $\mathcal{X}_i$ depends on $\underline{\theta}_i$ and the way the matrix square root of $\mathbf{C}_x$ is determined. Also the weights ω_i offer an additional degree of freedom to the sample set. Various methods have been proposed in the past for calculating ω_i and $\underline{\theta}_i$. Main drivers for the calculation typically are numerical quadrature techniques for solving Gaussian integrals or capturing information of the standard Gaussian $\mathcal{N}\left(\underline{\theta};\underline{0},\mathbf{I}\right)$ like its mean and covariance.

Before the most popular of them are briefly introduced, it is worth mentioning that even though LRKFs resemble Monte Carlo integration in (2.21), the sample points are determined in a deterministic fashion. Monte Carlo methods like particle filters instead employ random sampling.

Cubature Kalman Filter (CKF)

The *cubature Kalman filter* [11, 213] exploits the third-order spherical cubature integration rule. According to this rule, the sample set consists of $L = 2 \cdot n_x$ points

$$\underline{\theta}_i = \begin{cases} \sqrt{n_x} \cdot \underline{e}_i & \text{if } i = 1 \ldots n_x \\ -\sqrt{n_x} \cdot \underline{e}_{i-n_x} \,, & \text{if } i = n_x + 1 \ldots 2n_x \end{cases} \tag{2.22}$$

with weights $\omega_i = 1/2n_x$ for all i. In (2.22), $\underline{e}_i = [0\;0\;1\;0\;0 \cdots 0]^\mathrm{T}$ is the canonical unit vector, where only element i is one. The CKF calculates (2.19) exactly if $\underline{g}(.)$ is a linear combination of monomials of order up to three [10], while the covariance $\mathbf{C}_x$ in (2.8) is determined exactly if $\underline{g}(.)$ is linear.

Unscented Kalman Filter (UKF)

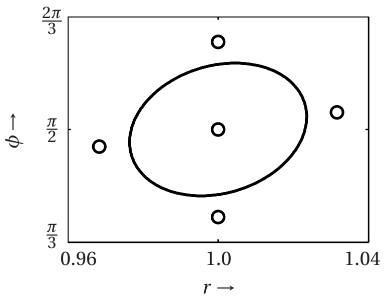
As shown in [159], the CKF approach can be generalized to the so-called *unscented transform* proposed by [94], which forms the basis for the unscented Kalman filter [205]. Therefore, instead of $2n_x$ sample points, $2n_x + 1$ points are considered. The additional point is specially dedicated to the mean of $\underline{\boldsymbol{x}}$. The sample points and the corresponding weights are given by

$$\underline{\theta}_i = \begin{cases} \sqrt{n_x + \kappa} \cdot \underline{e}_i & \text{if } i = 1 \ldots n_x \\ -\sqrt{n_x + \kappa} \cdot \underline{e}_{i-n_x} & \text{if } i = n_x + 1 \ldots 2n_x \\ \underline{0} & \text{if } i = 2n_x + 1 \end{cases} ,$$

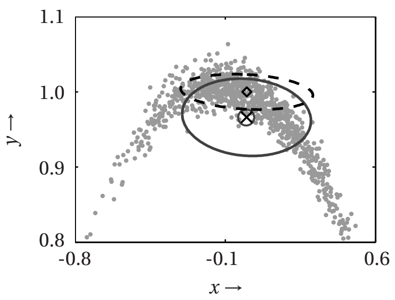
$$\omega_i = \begin{cases} \frac{1}{2(n_x+\kappa)} & \text{if } i = 1 \ldots n_x \\ \frac{1}{2(n_x+\kappa)} & \text{if } i = n_x + 1 \ldots 2n_x \\ \frac{\kappa}{n_x+\kappa} & \text{if } i = 2n_x + 1 \end{cases} ,$$

with κ being a free parameter. The placement of these sample points is depicted in Figure 2.2a on the next page. For $\kappa = 0$ the sample set is identical with the one provided by the CKF.

Like for the CKF, it can be shown that the above sample set exactly captures the mean and covariance of the multivariate standard Gaussian. Furthermore, (2.19) is evaluated exactly if $\underline{g}(.)$ is a polynomial of order up to three.



(a) Sample points of the UKF and covariance ellipse corresponding to the polar coordinates.



(b) True mean (black cross) as well as estimated mean and covariance of EKF (diamond for mean, dashed ellipse for covariance) and UKF (circle and solid gray ellipse).

Figure 2.2: Comparison of estimates of EKF and UKF. The gray dots forming a "banana" shape in (b) correspond to a Monte Carlo estimate of the true density.

Example 1: UKF vs. EKF

To demonstrate the difference in estimation between the UKF and the EKF, the nonlinear function

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \boldsymbol{r} \cdot \begin{bmatrix} \cos(\boldsymbol{\phi}) \\ \sin(\boldsymbol{\phi}) \end{bmatrix}, \quad \text{with } \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{\phi} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 1 \\ \frac{\pi}{2} \end{bmatrix}, \begin{bmatrix} 0.0004 & 0.001 \\ 0.001 & 0.0685 \end{bmatrix} \right)$$

is considered (see [95]), which transforms the polar coordinates (r,ϕ) to Cartesian coordinates (x,y). The true density of the Cartesian coordinates is clearly not Gaussian as can be seen in Figure 2.2b. However, the UKF almost exactly estimates the correct mean, while the estimate of the EKF is biased and inconsistent, especially in y direction. Furthermore, the EKF strongly underestimates the variance of the y coordinate.

Gaussian Estimator

The number of sample points of the CKF or UKF are restricted by the dimension of the state, although there are approaches to extend the sample set (see e.g.

[192]). To allow an arbitrary number of samples, [88] proposed a sampling scheme that incorporates additional information about the shape of the Gaussian density function, besides merely capturing mean and covariance. The sample points are therefore given by

$$\underline{\theta}_i = \begin{cases} \mu_i \cdot \underline{e}_i & \text{if } i = 1 \dots m \\ \mu_{i-m} \cdot \underline{e}_{i-m} & \text{if } i = m+1 \dots 2m \\ \vdots & \\ \mu_{i-n_x \cdot m} \cdot \underline{e}_{i-(n_x-1)\cdot m} & \text{if } i = (n_x - 1)\cdot m + 1 \dots n_x \cdot m \end{cases} \tag{2.23}$$

with weights $\omega_i = {}^1\!/\!(n_x \cdot m)$. Hence, the set of samples consists of $L = n_x \cdot m$ points, where m is a free parameter allowing to increase or decrease the required number of sample points. For each state dimension, the same parameters μ_i, $i = 1 \dots m$, in (2.23) are required, which are the roots of the set of nonlinear equations

$$\tfrac{1}{2}\left(1 + \operatorname{erf}\left(\tfrac{\mu_i}{\sqrt{2}}\right)\right) - \tfrac{2i-1}{2m} + \lambda \mu_i = 0\,,$$
$$\sum_{j=1}^{m} \mu_j^2 - m = 0\,,$$

where erf(.) is the Gaussian error function and λ is a Lagrangian multiplier. Finding the roots is not possible in closed form and thus, requires a numerical solution. The necessary computational overhead is uncritical, as the sample set is determined for the multivariate standard Gaussian and can be transformed on-line according to (2.21).

Gauss-Hermite Kalman Filter (GHKF)

As can be seen from (2.23), all sample points are placed along the coordinate axes, while no points are placed in the quadrants. The same observation holds for CKF and UKF. A more dense placement of sample points as a irregular grid results when applying the *Gauss-Hermite quadrature* rule. Accordingly, the sample points and corresponding weights are

$$\underline{\theta}_{\underline{i}} = \left[\theta_{i_1}\ \theta_{i_2}\ \cdots\ \theta_{i_{n_x}}\right]^{\mathrm{T}}, \quad \omega_{\underline{i}} = \prod_{j=1}^{n_x} \frac{m!}{\left(m \cdot H_{m-1}(\theta_{i_j})\right)^2}\,, \tag{2.24}$$

with $\underline{i} \triangleq (i_1 i_2 \dots i_{n_x})$ being an index vector where each index i_j takes values $1 \dots m$. The jth element θ_{i_j}, $j = 1 \dots n_x$, of the sample point $\underline{\theta}_{\underline{i}}$ is one of the m roots of m-order Hermite polynomial

$$H_m(x) = x \cdot H_{m-1}(x) - (m-1) \cdot H_{m-2}(x) \,, \quad m = 2, 3, \dots \tag{2.25}$$

whereas this recursion commences from $H_0(x) = 1$ and $H_1(x) = x$.

The LRKF employing the above set of sigma points is named *Gauss-Hermite Kalman Filter* and was proposed in [13, 90]. Like the Gaussian estimator, the number of sample points can be varied. However, while for the Gaussian estimator the number of samples still scales linearly with the dimension of the state, the number of samples of the GHKF scales exponentially due the grid placement of the samples. This comes with the advantage that a GHKF using an m-order Hermite polynomial is exact is for polynomials up to order $2m-1$.

Implicit Linearization

Besides the discussed LRKFs, there exist further approaches like the central difference filter [161] or the divided difference filter [133], which are not discussed here in order to constrain the focus on the mostly used approaches. However, all LRKFs share the property that they directly target the integrations (2.8). In doing so, also a linear transformation approximating the nonlinear one in (2.6) is constructed, although implicitly. As has been shown in [112, 203], LRKFs actually perform *weighted statistical linear regression* by approximating $\underline{g}(.)$ according to

$$\underline{g}(\underline{\boldsymbol{x}}) \approx \mathbf{G} \cdot \underline{\boldsymbol{x}} + \underline{b} \,, \tag{2.26}$$

where $\mathbf{G}$ and $\underline{b}$ result from minimizing

$$\sum_{i=1}^{L} \omega_i \cdot \left(\underline{g}(\mathcal{X}_i) - \mathbf{G} \cdot \mathcal{X}_i - \underline{b}\right)^{\mathrm{T}} \left(\underline{g}(\mathcal{X}_i) - \mathbf{G} \cdot \mathcal{X}_i - \underline{b}\right) .$$

The desired quantities are then given by

$$\mathbf{G} = \mathbf{C}_{xy}^{\mathrm{T}} \mathbf{C}_x^{-1} \,, \quad \underline{b} = \underline{\mu}_y - \mathbf{G} \cdot \underline{\mu}_x \,,$$

where

$$\underline{\mu}_x = \sum_i \omega_i \cdot \mathcal{X}_i \,, \qquad \mathbf{C}_x = \sum_i \omega_i \cdot \left(\mathcal{X}_i - \underline{\mu}_x\right)\left(\mathcal{X}_i - \underline{\mu}_x\right)^{\mathrm{T}} ,$$

$$\underline{\mu}_y \approx \sum_i \omega_i \cdot \underline{g}(\mathcal{X}_i) \,, \qquad \mathbf{C}_{xy} \approx \sum_i \omega_i \cdot \left(\mathcal{X}_i - \underline{\mu}_x\right)\left(\underline{g}(\mathcal{X}_i) - \underline{\mu}_y\right)^{\mathrm{T}}$$

are the sample means and covariances determined by means of the set of sample points $\mathcal{L}_x = \{\omega_i, \mathcal{X}_i\}$.

The error of the linearization (2.26) is given by

$$\underline{\boldsymbol{e}} = \underline{g}(\underline{\boldsymbol{x}}) - \mathbf{G} \cdot \underline{\boldsymbol{x}} - \underline{b} \tag{2.27}$$

and depends on the nonlinear function $\underline{g}(.)$ as well as on the state $\underline{\boldsymbol{x}}$. Both are main sources for linearization errors, which become large for strong nonlinearities or when the covariance of the state and thus its spread is large. It was shown in [112] that for LRKFs the error (2.27) has zero mean and a covariance matrix

$$\mathbf{C}_e = \mathbf{C}_y - \mathbf{G}\mathbf{C}_x\mathbf{G}^{\mathrm{T}} \,. \tag{2.28}$$

The latter is a potential and easy to evaluate indicator of the linearization error. If $\mathbf{C}_e$ is a zero matrix, the density of the error $\underline{\boldsymbol{e}}$ corresponds to a Dirac delta distribution [136] and the transformation $\underline{g}(.)$ is affine with $\underline{g}(\underline{\boldsymbol{x}}) = \mathbf{G} \cdot \underline{\boldsymbol{x}} + \underline{b}$. Accordingly, LRKFs are exact for linear/affine transformations and thus, degenerate to a standard Kalman filter.

2.3 Gaussian Smoothing

So far, the focus was on performing predictions and measurement updates for Gaussian filters. Now, the missing smoothing step is derived. According to (1.9), smoothing is a backward recursion for incorporating not only the current but also future measurements in the state density. For Gaussian filters it is assumed that the smoothed density is Gaussian, i.e.,

$$f_k^s(\underline{x}_k) = f_k^x\left(\underline{x}_k \middle| \underline{\hat{z}}_{0:K}, \underline{u}_{0:K-1}\right) \approx \mathcal{N}\left(\underline{x}_k; \underline{\mu}_k^s, \mathbf{C}_k^s\right) \tag{2.29}$$

with appropriate mean vector $\underline{\mu}_k^s$ and covariance matrix $\mathbf{C}_k^s$. The derivation of the smoothing recursion for calculating (2.29) is based on the so-called *Rauch-*

Tung-Striebel smoother (RTSS, see [144]) for linear systems and follows loosely [54, 84].

2.3.1 General Formulation

It is assumed that both the smoothed Gaussian $\mathcal{N}\big(\underline{x}_{k+1};\underline{\mu}^s_{k+1},\mathbf{C}^s_{k+1}\big)$ and the posterior Gaussian $\mathcal{N}\big(\underline{x}_k;\underline{\mu}^e_k,\mathbf{C}^e_k\big)$ are already given. As smoothing is an off-line task that is usually performed after $K \geq 1$ prediction and measurement update steps, the assumption of a known smoothed Gaussian is valid since for the time step K, the smoothed density coincides with the posterior Gaussian, i.e., $f^e_K\big(\underline{x}_k\big) \equiv f^s_K\big(\underline{x}_k\big) = \mathcal{N}\big(\underline{x}_K;\underline{\mu}^s_K,\mathbf{C}^s_K\big)$ and thus knowing $\mathcal{N}\big(\underline{x}_{k+1};\underline{\mu}^s_{k+1},\mathbf{C}^s_{k+1}\big)$ follows by induction.

Similar to Section 2.2.1, the interest is in the joint Gaussian

$$\begin{bmatrix}\underline{\boldsymbol{x}}_k\\ \underline{\boldsymbol{x}}_{k+1}\end{bmatrix} \sim \mathcal{N}\Big(\underline{\mu}_x,\mathbf{C}_x\Big) \quad \text{with } \underline{\mu}_x \triangleq \begin{bmatrix}\underline{\mu}^s_k\\ \underline{\mu}^s_{k+1}\end{bmatrix}, \mathbf{C}_x \triangleq \begin{bmatrix}\mathbf{C}^s_k & \mathbf{C}_{k|k+1}\\ \mathbf{C}^{\mathrm{T}}_{k|k+1} & \mathbf{C}^s_{k+1}\end{bmatrix}, \tag{2.30}$$

where the desired smoothed mean vector and covariance matrix for time step k can be retrieved from the first row of (2.30). To obtain both quantities, the joint Gaussian is rewritten as

$$\mathcal{N}\Big(\big[\underline{x}_k,\underline{x}_{k+1}\big]^{\mathrm{T}};\underline{\mu}_x,\mathbf{C}_x\Big) = \mathcal{N}\big(\underline{x}_k;\underline{\mu},\mathbf{C}\big)\cdot\mathcal{N}\big(\underline{x}_{k+1};\underline{\mu}^s_{k+1}\mathbf{C}^s_{k+1}\big), \tag{2.31}$$

which is the product of the known smoothed Gaussian and the conditional Gaussian $\mathcal{N}\big(\underline{x}_k;\underline{\mu},\mathbf{C}\big) \triangleq f\big(\underline{x}_k|\underline{x}_{k+1},\underline{\hat{z}}_{0:k}\big)$, where the latter is independent of the future measurements $\underline{\hat{z}}_{k+1:K}$ due to conditioning on $\underline{\boldsymbol{x}}_{k+1}$. However, the conditional Gaussian is unknown, but it can be obtained from (2.9) when replacing $\underline{\boldsymbol{x}}$ with $\underline{\boldsymbol{x}}_k \sim \mathcal{N}\big(\underline{\mu}^e_k,\mathbf{C}^e_k\big)$, $\underline{\boldsymbol{y}}$ with $\underline{\boldsymbol{x}}_{k+1} \sim \mathcal{N}\big(\underline{\mu}^p_{k+1},\mathbf{C}^p_{k+1}\big)$ and the nonlinear transformation $\underline{g}(.)$ with $\underline{a}_k(.)$. Hence, the unknown mean vector and covariance matrix in (2.31) are given by

$$\begin{aligned}\underline{\mu} &= \underline{\mu}^e_k + \mathbf{J}_k\cdot\Big(\underline{x}_{k+1} - \underline{\mu}^p_{k+1}\Big),\\ \mathbf{C} &= \mathbf{C}^e_k - \mathbf{J}_k\mathbf{C}^{\mathrm{T}}_{k|k+1},\end{aligned} \tag{2.32}$$

respectively, with gain matrix $\mathbf{J}_k = \mathbf{C}_{k|k+1}\left(\mathbf{C}_{k+1}^p\right)^{-1}$ and cross-covariance matrix $\mathbf{C}_{k|k+1}$ according to

$$\begin{aligned}\mathbf{C}_{k|k+1} &= \mathrm{Cov}\{\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{x}}_{k+1}\} \\ &= \int \left(\underline{x}_k - \underline{\mu}_k^e\right)\left(\underline{a}_k(\underline{x}_k, \underline{u}_k) - \underline{\mu}_{k+1}^p\right)^{\mathrm{T}} \cdot \mathcal{N}\left(\underline{x}_k; \underline{\mu}_k^e, \mathbf{C}_k^e\right) \mathrm{d}\underline{x}_k \,. \end{aligned} \tag{2.33}$$

Substituting the mean and covariance of (2.32) into (2.31) and solving the product[4] leads to the desired joint Gaussian of (2.30), where the smoothed mean and covariance at time step k are given by

$$\begin{aligned}\underline{\mu}_k^s &= \underline{\mu}_k^e + \mathbf{J}_k \cdot \left(\underline{\mu}_{k+1}^s - \underline{\mu}_{k+1}^p\right) \,, \\ \mathbf{C}_k^s &= \mathbf{C}_k^e + \mathbf{J}_k\left(\mathbf{C}_{k+1}^s - \mathbf{C}_{k+1}^p\right)\mathbf{J}_k^{\mathrm{T}} \,, \end{aligned} \tag{2.34}$$

respectively. It is obvious that all quantities in (2.34) except of the gain matrix $\mathbf{J}_k$ are known from predictions and measurement updates or are calculated in a previous smoothing step. The matrix $\mathbf{J}_k$ requires solving the integral in (2.33), which is not possible in closed form in general due to the nonlinear system function $\underline{a}_k(.)$.

2.3.2 Linear Case

In case of linear dynamic and measurement models as in (2.11), a closed-form expression of the integral (2.33) and thus of the smoothing recursion (2.34) can be found. The resulting RTS smoother or sometimes *Kalman smoother* coincides with (2.34), where the gain matrix is

$$\mathbf{J}_k = \mathbf{C}_k^e \mathbf{A}_k^{\mathrm{T}}\left(\mathbf{C}_{k+1}^p\right)^{-1} \tag{2.35}$$

and the predicted as well as the posterior parameters are

$$\begin{aligned}\underline{\mu}_{k+1}^p &= \mathbf{A}_k \cdot \underline{\mu}_k^e \,, & \underline{\mu}_k^e &= \underline{\mu}_k^p + \mathbf{K}_k \cdot \left(\underline{\hat{z}}_k - \mathbf{H}_k \cdot \underline{\mu}_k^p\right) \,, \\ \mathbf{C}_{k+1}^p &= \mathbf{A}_k \mathbf{C}_k^e \mathbf{A}_k^{\mathrm{T}} + \mathbf{C}_k^w \,, & \mathbf{C}_k^e &= \mathbf{C}_k^p - \mathbf{K}_k \mathbf{H}_k \mathbf{C}_k^p \,. \end{aligned}$$

The latter correspond to the Kalman filter predictions (2.12) and measurement updates (2.13), respectively.

4 A detailed solution of this product can be found in Paper J, Section 3.1.

2.3.3 Nonlinear Case

Similar to the predictions and measurement updates in case of nonlinear models, it is sufficient for Gaussian smoothing to find a linear approximation of the nonlinear system function $\underline{a}_k(.)$ or to solve the integral (2.33). Hence, all the nonlinear Gaussian filters discussed in Sections 2.2.3–2.2.5 can be employed, as they explicitly or implicitly provide a linear approximation and solve (2.33), respectively. Accordingly, for any of the nonlinear Gaussian filters a corresponding nonlinear Gaussian smoother can be found that exploits the respective approximation technique of the filter for determining the required quantities in (2.34). See for instance [8] for the extended Kalman smoother, [12] for the cubature Kalman smoother, or [158, 171] for the unscented RTS smoother.

2.4 Rao-Blackwellization

The perspective of solving a given Bayesian filtering problem was so far very coarse. Either the models were identified as being linear and the exact solution can be found, or the models are nonlinear and one has to rely on approximations. Actually, besides these black and white decisions, there are many gray scale values in between.

Example 2: Linear Substructure

Consider the measurement model

$$\underline{z} = \underline{h}(\underline{\boldsymbol{x}}_n) + \mathbf{H}(\underline{\boldsymbol{x}}_n) \cdot \underline{\boldsymbol{x}}_l + \underline{\boldsymbol{v}} \,, \tag{2.36}$$

where the measurement function comprises nonlinear parts denoted by $\underline{h}(.)$ as well as linear substructures indicated by the matrix $\mathbf{H}(.)$. Accordingly, the state $\underline{\boldsymbol{x}}$ consists of two *sub-states* according to

$$\underline{\boldsymbol{x}} = \begin{bmatrix} \underline{\boldsymbol{x}}_l \\ \underline{\boldsymbol{x}}_n \end{bmatrix} \,, \tag{2.37}$$

where $\underline{\boldsymbol{x}}_n$ comprises the nonlinear state variables, while $\underline{\boldsymbol{x}}_l$ comprises all state variables with *conditionally linear* dynamics, i.e., when conditioning on $\underline{\boldsymbol{x}}_n$, the model (2.36) becomes a linear model.

By exploiting the linear substructure in the above example, it is possible to solve some of the filtering equations analytically, while an approximation is merely required for the nonlinear parts. The estimation performance gain of this decomposed processing is stated by the following theorem.

Theorem 1 (Rao-Blackwell, [127], Ch. 24) *Let $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$ be dependent variables, and $g(\underline{\boldsymbol{x}}, \underline{\boldsymbol{y}})$ be some scalar function. Then*

$$\mathrm{var}_{\underline{x},\underline{y}}\left\{g(\underline{\boldsymbol{x}}, \underline{\boldsymbol{y}})\right\} \geq \mathrm{var}_{\underline{x}}\left\{\mathrm{E}_{\underline{y}}\left\{g(\underline{\boldsymbol{x}}, \underline{\boldsymbol{y}})\middle|\underline{\boldsymbol{x}}\right\}\right\} . \qquad \square$$

According to this so-called *Rao-Blackwell theorem*, a Bayesian filter utilizing this decomposition will result in a lower variance than a filter without it. This idea has been employed in many Bayesian filtering approaches like in particle filters [9, 40, 165] or in LRKFs [125]. Even though these filters employ this theorem mainly in case of linear-nonlinear substructures as in (2.36), it is important to note that this theorem is not restricted to those.

2.5 Contributions

In this section, the main contributions of the Papers A–D are summarized. The first two contributions in Section 2.5.1 and Section 2.5.2 exploited the aforementioned Rao-Blackwellization for Gaussian filtering. For Sections 2.5.3–2.5.5, it is assumed that the nonlinear function $\underline{g}(.)$ is either given as a polynomial and can be approximated by a polynomial. In doing so, efficient moment calculation algorithms are proposed.

2.5.1 Combining Rao-Blackwellization with Observed-Unobserved Decomposition

Measurement models with the same structure as in Example 2 are considered. In addition it is assumed that the state not only comprises the nonlinear and conditionally linear variables, but also contains state variables $\underline{\boldsymbol{x}}_u$ that are *not directly observable*, i.e., there is no functional relation between the measurement vector $\underline{\boldsymbol{z}}$ and $\underline{\boldsymbol{x}}_u$ through the measurement model (2.36). Hence, the (predicted) state vector has the form

$$\underline{\boldsymbol{x}}^p = \begin{bmatrix} \underline{\boldsymbol{x}}_o^p \\ \underline{\boldsymbol{x}}_u^p \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \underline{\mu}_o^p \\ \underline{\mu}_u^p \end{bmatrix}, \begin{bmatrix} \mathbf{C}_o^p & \mathbf{C}_{ou}^p \\ \mathbf{C}_{uo}^p & \mathbf{C}_u^p \end{bmatrix} \right) , \qquad (2.38)$$

where the *observed* part comprises the nonlinear and conditionally linear variables

$$\underline{\boldsymbol{x}}_o^p = \begin{bmatrix} \underline{\boldsymbol{x}}_n^p \\ \underline{\boldsymbol{x}}_l^p \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \underline{\mu}_n^p \\ \underline{\mu}_l^p \end{bmatrix}, \begin{bmatrix} \mathbf{C}_n^p & \mathbf{C}_{nl}^p \\ \mathbf{C}_{ln}^p & \mathbf{C}_l^p \end{bmatrix} \right) . \qquad (2.39)$$

Such kind of state compositions appear in many application, where one is exemplified next.

Example 3: Object Localization

Consider a filtering problem, where the pose—location and orientation—and the corresponding velocities of an object in 3D are of interest. Sensors typically used for such localization problems are inertial sensors like gyroscopes and absolute localization techniques based on for instance multilateration. In such a situation, the system state may comprise the object pose $\underline{\boldsymbol{x}}_n = [x\ y\ z\ \alpha\ \beta\ \gamma]^{\mathrm{T}}$, the translational velocities $\underline{\boldsymbol{x}}_u = [\dot{x}\ \dot{y}\ \dot{z}]^{\mathrm{T}}$, and the angular velocities $\underline{\boldsymbol{x}}_l = [\dot{\alpha}\ \dot{\beta}\ \dot{\gamma}]^{\mathrm{T}}$ (see e.g. [18]). Hence, even though the state as a whole is propagated via a dynamics model reflecting the motion of the object, the translational velocities cannot be observed directly.

Although there is no functional relation through the measurement model, the unobserved state variables are still updated thanks to the correlation between observed and unobserved states, which is reflected by the cross-covariance $\mathbf{C}^{ou}$. However, due to the missing functional relation, it seems to be a waste of computations to utilizes the state as a whole for evaluating the integrals (2.8), especially when the unobserved state is of high dimension. To reduce the computational load, a two-step decomposition of the filtering problem is proposed:

1. Updating the directly observed state $\underline{\boldsymbol{x}}_o$ first with the current measurement. In order to improve the estimation performance, the nonlinear and conditionally linear structure in accordance with Rao-Blackwellization is exploited.

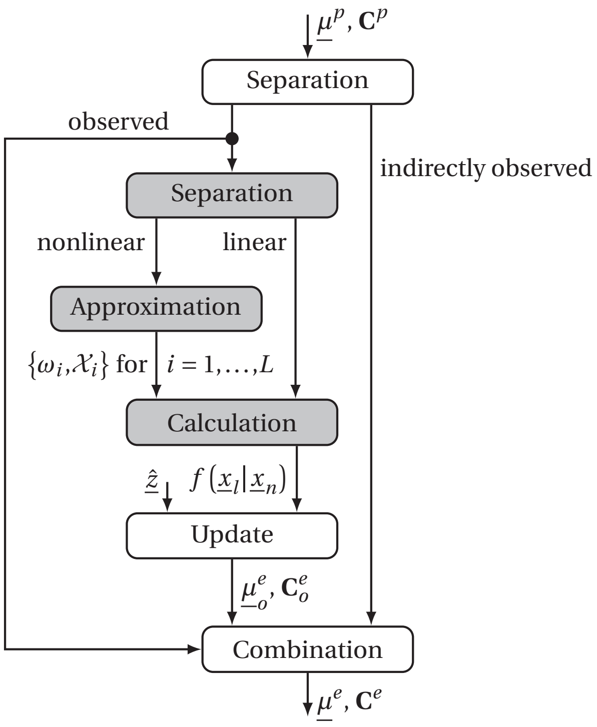

$\underline{\mu}^p$, $\mathbf{C}^p$
Separation
observed
indirectly observed
Separation
nonlinear
linear
Approximation
$\{\omega_i, \mathcal{X}_i\}$ for $i = 1, \ldots, L$
Calculation
$\hat{\underline{z}}$
$f(\underline{x}_l | \underline{x}_n)$
Update
$\underline{\mu}_o^e$, $\mathbf{C}_o^e$
Combination
$\underline{\mu}^e$, $\mathbf{C}^e$

Figure 2.3: Information flow of the measurement update. The gray boxes indicate components that can be reused for the prediction step.

2. Given the updated observed state, update the unobserved state $\underline{\boldsymbol{x}}_u$ by exploiting the correlation between $\underline{\boldsymbol{x}}_o$ and $\underline{\boldsymbol{x}}_u$.

The information flow of this two-step processing is depicted in Figure 2.3, which corresponds to the decomposition

$$f^e(\underline{x}) = f(\underline{x}|\hat{\underline{z}}) = f(\underline{x}_u, \underline{x}_o|\hat{\underline{z}}) = f(\underline{x}_u|\underline{x}_o) \cdot \underbrace{f(\underline{x}_o|\hat{\underline{z}})}_{= f(\underline{x}_o^e) = f\left(\underline{x}_n^e, \underline{x}_l^e | \hat{\underline{z}}\right)} \tag{2.40}$$

of the conditional Gaussian (2.9).

Update of Observed State

Updating the observed state corresponds to calculating the conditional Gaussian $\underline{\boldsymbol{x}}_o^e \sim \mathcal{N}\big(\underline{\mu}_o^e, \mathbf{C}_o^e\big)$ according to (2.9), which requires to solve the integrals in (2.8). The procedure for this is shown exemplary for the measurement mean vector $\underline{\mu}_z$. Given the measurement function in (2.36), the mean vector is given by

$$\underline{\mu}_z = \int \big(\underline{h}(\underline{x}_n) + \mathbf{H}(\underline{x}_n) \cdot \underline{x}_l\big) \cdot \underbrace{f\big(\underline{x}_n, \underline{x}_l\big)}_{= f(\underline{x}_l|\underline{x}_n) \cdot f(\underline{x}_n)} \, \mathrm{d}\underline{x}_o \,, \tag{2.41}$$

where $f\big(\underline{x}_l|\underline{x}_n\big) = \mathcal{N}\Big(\underline{x}_l; \underline{\mu}_{l|n}\big(\underline{x}_n\big), \mathbf{C}_{l|n}\Big)$ with mean vector and covariance matrix

$$\begin{aligned} \underline{\mu}_{l|n}\big(\underline{x}_n\big) &= \underline{\mu}_l^p + \mathbf{C}_{ln}^p \left(\mathbf{C}_n^p\right)^{-1} \left(\underline{x}_n - \underline{\mu}_n^p\right) \,, \\ \mathbf{C}_{l|n} &= \mathbf{C}_l^p - \mathbf{C}_{ln}^p \left(\mathbf{C}_n^p\right)^{-1} \mathbf{C}_{nl}^p \,, \end{aligned} \tag{2.42}$$

respectively. Hence, the mean of the conditional linear Gaussian $f\big(\underline{x}_l|\underline{x}_n\big)$ is a function of the nonlinear state.

Due to the nonlinear substate $\underline{\boldsymbol{x}}_n$, solving the above integral in closed form is not possible in general. To approximate the solution, the deterministic sampling techniques of the LRKFs can be employed here. In doing so, the Gaussian density $f\big(\underline{x}_n\big)$ is approximated by a mixture of Dirac delta distributions $\sum_i \omega_i \cdot \delta\left(\underline{x}_n - \mathcal{X}_i\right)$. Substituting this sample representation in (2.41) and utilizing the sifting properties of the Dirac delta distributions leads to

$$\begin{aligned} \underline{\mu}_z &= \sum_{i=1}^{L} \omega_i \cdot \int \big(\underline{h}(\mathcal{X}_i) + \mathbf{H}(\mathcal{X}_i) \cdot \underline{x}_l\big) \cdot f\big(\underline{x}_l|\mathcal{X}_i\big) \, \mathrm{d}\underline{x}_l \\ &= \sum_{i=1}^{L} \omega_i \cdot \underbrace{\Big(\underline{h}(\mathcal{X}_i) + \mathbf{H}(\mathcal{X}_i) \cdot \underline{\mu}_{l|n}(\mathcal{X}_i)\Big)}_{\triangleq \underline{\mu}_i^z} \,. \end{aligned}$$

The second equation follows from (2.42) and corresponds to a Kalman prediction thanks to the conditionally linear measurement function and the Gaussian density $f(\underline{x}_l | \mathcal{X}_i)$. Similarly the desired covariance $\mathbf{C}^z$ and cross-covariance $\mathbf{C}^{zx}$ are obtained

$$\mathbf{C}^z = \sum_{i=1}^{L} \omega_i \cdot \left(\left(\underline{\mu}_i^z - \underline{\mu}_z \right) \left(\underline{\mu}_i^z - \underline{\mu}_z \right)^{\mathrm{T}} + \mathbf{H}(\mathcal{X}_i) \mathbf{C}_{l|n} \mathbf{H}(\mathcal{X}_i)^{\mathrm{T}} \right) ,$$
$$\mathbf{C}^{oz} = \sum_{i=1}^{L} \omega_i \cdot \left(\begin{bmatrix} \mathbf{O} \\ \mathbf{C}_{l|n} \mathbf{H}(\mathcal{X}_i)^{\mathrm{T}} \end{bmatrix} + \left(\begin{bmatrix} \mathcal{X}_i \\ \underline{\mu}_{l|n}(\mathcal{X}_i) \end{bmatrix} - \underline{\mu}_o \right) \left(\underline{\mu}_i^z - \underline{\mu}_z \right) \right) ,$$

where again Kalman predictions have been used. Conditioning on the measurement $\underline{\boldsymbol{z}}$ according to (2.9) yields the desired updated observed state $\underline{\boldsymbol{x}}_o^e$.

Update of Unobserved State

For updating the indirectly observed state, the mean vector $\underline{\mu}_o^e$ and covariance matrix $\mathbf{C}_o^e$ of the posterior Gaussian $f(\underline{x}_o | \hat{\underline{z}})$ are used. According to [113], the mean vector of the unobserved state is updated via

$$\underline{\mu}_u^e = \underline{\mu}_u^p + \mathbf{J} \cdot \left(\underline{\mu}_o^e - \underline{\mu}_o^p \right) \tag{2.43}$$

with gain matrix $\mathbf{J} = \mathbf{C}_{uo}^p \left(\mathbf{C}_o^p \right)^{-1}$. The posterior covariance matrix of the unobserved state and cross-covariance matrix between observed and unobserved state are given by

$$\begin{aligned} \mathbf{C}_u^e &= \mathbf{C}_u^p + \mathbf{J} \left(\mathbf{C}_o^e - \mathbf{C}_o^p \right) \mathbf{J}^{\mathrm{T}} , \\ \mathbf{C}_{uo}^e &= \mathbf{J} \mathbf{C}_o^e , \end{aligned} \tag{2.44}$$

respectively. By comparison with (2.34), it is apparent that the above update equations coincide with an RTS smoother.

2.5.2 Semi-Analytical Filtering

The *semi-analytical Gaussian filter* (SAGF) introduced next takes Rao-Blackwellization to its extremes. Instead of merely restricting the decomposition to linear and nonlinear substructures, the SAGF exploits nonlinear-nonlinear decompositions, where for some nonlinear state variables an analytical solution of the Gaussian filtering problem can be found. Nonlinear functions for which analytical solutions exist are for example

- Monomials x^i with $i \in \mathbb{N}$,
- Trigonometric functions $\sin(x)$ and $\cos(x)$,
- Squared exponential functions $\exp\left(\underline{c}^{\mathrm{T}} \cdot \underline{\phi}\left(\underline{x}\right)\right)$ with $\underline{c} \in \mathbb{R}^3$ and $\underline{\phi}\left(\underline{x}\right) \triangleq \left[1 \; x \; x^2\right]^{\mathrm{T}}$,
- and linear combinations of the above functions,

assuming that the state density is Gaussian. The next example demonstrates the difference between a closed-form calculation of the moments in (2.8) for a quadratic function and the solution of an LRKF.

Example 4: Quadratic Transformation

For the quadratic transformation $\boldsymbol{y} = \boldsymbol{x}^2 + \boldsymbol{v}$ with $\boldsymbol{x} \sim \mathcal{N}\left(\mu_x, \sigma_x^2\right)$ and $\boldsymbol{v} \sim \mathcal{N}\left(0, \sigma_v^2\right)$ the moments in (2.8) can be calculated exactly to

$$\mu_y = \mu_x^2 + \sigma_x^2 \,, \qquad \sigma_y^2 = 2\sigma_x^2 \cdot \left(\sigma_x^2 + 2\mu_x^2\right) + \sigma_v^2 \,, \qquad \sigma_{xy} = 2 \cdot \sigma_x^2 \cdot \mu_x \,.$$

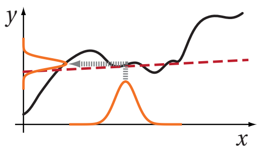
Using the UKF with parameter $\kappa = 0$ the same moments are calculated to

$$\mu_y = \mu_x^2 + \sigma_x^2 \,, \qquad \sigma_y^2 = 4\sigma_x^2 \cdot \mu_x^2 + \sigma_v^2 \,, \qquad \sigma_{xy} = 2 \cdot \sigma_x^2 \cdot \mu_x \,.$$

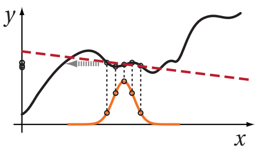
The mean μ_y is correct, which is not surprising as the unscented transform is exact for monomials up to order three. As the variance calculation for a quadratic function corresponds to an expectation calculation for a monomials of order four, the UKF introduces an error. More precisely, the variance σ_y^2 is underestimated as the term $2\sigma_x^4$ is missing and thus, the UKF is overconfident in this example.

On the one hand, analytical moment matching provides exact solutions to (2.8), but is restricted to a few nonlinear transformations, while on the other hand, LRKFs are generally applicable but may introduce severe linearization errors. The difference of LRKFs and analytical moment calculation from a linearization perspective is depicted in Figure 2.4.

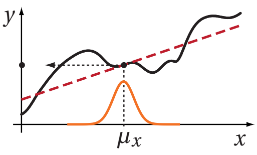
The key idea of the SAGF is to combine both worlds by means of Rao-Blackwellization. Only some dimensions of the state $\underline{\boldsymbol{x}}$ are sampled via an LRKF and



(a) Analytic Moment Matching



(b) Sample-based linearization (LRKF)



(c) Linearization via Taylor-series (EKF)

Figure 2.4: Illustration of different Gaussian filtering approaches: the nonlinear function (black) and its linearized versions (red dashed). Analytic moment matching utilizes the entire density $f(\underline{x})$ for (implicit) linearization, while the linearization of an LRKF is based on an approximate sample representation of $f(\underline{x})$. Thus, although the mean and covariance of $\underline{\boldsymbol{x}}$ are captured exactly by the samples, the same is not true for higher-order moment due to the finite number of samples. The EKF even linearizes the nonlinear function only around a single nominal point.

thus, only some parts of the nonlinear transformation (2.6) have to be treated approximately. For this purpose, the transformation is rearranged to the mapping

$$\underline{\boldsymbol{y}} = \underline{g}(\underline{\boldsymbol{x}}_a, \underline{\boldsymbol{x}}_s) + \underline{\boldsymbol{w}} , \tag{2.45}$$

where the Gaussian state $\underline{\boldsymbol{x}}^{\mathrm{T}} = [\underline{\boldsymbol{x}}_a^{\mathrm{T}}\ \underline{\boldsymbol{x}}_s^{\mathrm{T}}]$ consists of the substates $\underline{\boldsymbol{x}}_a$ (analytically integrable) and $\underline{\boldsymbol{x}}_s$ (sampled) with mean and covariance

$$\underline{\mu}_x = \begin{bmatrix} \underline{\mu}_a \\ \underline{\mu}_s \end{bmatrix} , \quad \mathbf{C}_x = \begin{bmatrix} \mathbf{C}_a & \mathbf{C}_{as} \\ \mathbf{C}_{sa} & \mathbf{C}_s \end{bmatrix} , \tag{2.46}$$

respectively. As there exists no closed-form expression for the desired moments (2.8), the decomposition into $\underline{\boldsymbol{x}}_a$ and $\underline{\boldsymbol{x}}_s$ is chosen in such a way that the moment integrals can be calculated analytically exactly for any given fixed value of $\underline{\boldsymbol{x}}_s$. Hence, the function $\underline{g}(.,.)$ is denoted to be *conditionally integrable*. For determining a sample-based representation of $\underline{\boldsymbol{x}}_s$, the sampling via LRKFs is applied.

It is worth mentioning that analytic moment matching and LRKFs are extreme cases of the SAGF: if $\underline{\boldsymbol{x}}_s$ is an empty vector, SAGF performs analytic moment matching and if $\underline{\boldsymbol{x}}_a$ is empty, the SAGF degenerates to an LRKF.

For the transformation given by (2.45), the moment calculation is shown exemplary for the mean vector $\underline{\mu}_y$

$$\underline{\mu}_y = \int \underline{g}(\underline{x}_a, \underline{x}_s) \cdot f(\underline{x}_a, \underline{x}_s)\,\mathrm{d}\underline{x} = \int \underline{g}(\underline{x}_a, \underline{x}_s) \cdot f(\underline{x}_a | \underline{x}_s) \cdot f(\underline{x}_s)\,\mathrm{d}\underline{x} \qquad (2.47)$$

with the conditional Gaussian $f(\underline{x}_a | \underline{x}_s) = \mathcal{N}\left(\underline{x}_a; \underline{\mu}_{a|s}, \mathbf{C}_{a|s}\right)$ with mean and covariance

$$\begin{aligned} \underline{\mu}_{a|s} &= \underline{\mu}_a + \mathbf{C}_{as} \cdot \mathbf{C}_s^{-1} \cdot \left(\underline{x}_s - \underline{\mu}_s\right) , \\ \mathbf{C}_{a|s} &= \mathbf{C}_a - \mathbf{C}_{as} \cdot \mathbf{C}_s^{-1} \cdot \mathbf{C}_{sa} . \end{aligned} \qquad (2.48)$$

By approximating the density $f(\underline{x}_s)$ of the sub-state $\underline{\boldsymbol{x}}_s$ with a mixture of Dirac delta distributions and by exploiting the sifting property, (2.47) simplifies to

$$\underline{\mu}_y \approx \sum_i \omega_i \cdot \underline{\mu}_i^y \quad \text{with} \quad \underline{\mu}_i^y = \int \underline{g}(\underline{x}_a, \mathcal{X}_i) \cdot f(\underline{x}_a | \mathcal{X}_i)\,\mathrm{d}\underline{x}_a .$$

It is important to note that this integral can be evaluated analytically as the function $\underline{g}(\cdot,\cdot)$ is conditionally integrable. Furthermore, solving these integrals is an off-line task and the solution is characterized by a parametric representation of the moments (2.48) and the sample points $\mathcal{X}_i$ for efficient on-line evaluation.

Example 5: Falling Body

To demonstrate improved estimation performance, the estimation of the altitude $\boldsymbol{\alpha}_k$, velocity $\boldsymbol{\beta}_k$, and constant ballistic coefficient $\boldsymbol{\gamma}_k$ of a falling body is considered [93, 173]. The system equation is given by

$$\underline{\boldsymbol{x}}_{k+1} = \begin{bmatrix} \boldsymbol{\alpha}_k \\ \boldsymbol{\beta}_k \\ \boldsymbol{\gamma}_k \end{bmatrix} + \Delta_t \cdot \begin{bmatrix} -\boldsymbol{\beta}_k \\ -\mathrm{e}^{-\rho \cdot \boldsymbol{\alpha}_k} \cdot (\boldsymbol{\beta}_k)^2 \cdot \boldsymbol{\gamma}_k \\ 0 \end{bmatrix} + \underline{\boldsymbol{w}}_k , \qquad (2.49)$$

where $\underline{\boldsymbol{x}}_k = [\boldsymbol{\alpha}_k\ \boldsymbol{\beta}_k\ \boldsymbol{\gamma}_k]^{\mathrm{T}}$ is the state vector, $\Delta_t = 1$ the time discretization constant, $\rho = 5 \cdot 10^{-5}$ a constant factor. The noise $\underline{\boldsymbol{w}}_k$ is zero-mean Gaussian with covariance matrix $\mathbf{C}_k^w = 0.1 \cdot \mathbf{I}$. The initial state of the falling body

Table 2.1: Average rmse and its standard deviation over all simulation runs.

	Altitude	Velocity	Ballistic coefficient
SAGF	**12.6 ± 8.3**	**59.3 ± 143.7**	**0.016 ± 0.063**
UKF	14.2 ± 7.9	100.1 ± 212.4	**0.016 ± 0.058**
GPF 100 p.	13.0 ± 8.0	60.2 ± 134.9	0.029 ± 0.098
GPF 1000 p.	**12.7 ± 8.2**	**59.3 ± 142.1**	0.019 ± 0.066

is $\underline{x}_0^{\mathrm{T}} = \begin{bmatrix} 3\cdot 10^5 & 2\cdot 10^4 & 10^{-3} \end{bmatrix}$. The initial mean and covariance of the estimators for all simulation runs is set to be

$$\underline{\mu}^x = \begin{bmatrix} 3\cdot 10^5 \\ 2\cdot 10^4 \\ 10^{-5} \end{bmatrix} \quad , \quad \mathbf{C}^x = \begin{bmatrix} 10^6 & 0 & 0 \\ 0 & 4\cdot 10^6 & 0 \\ 0 & 0 & 20 \end{bmatrix} .$$

The state variables can be decomposed into $\underline{\boldsymbol{x}}_a = [\boldsymbol{\beta}_k \, \boldsymbol{\gamma}_k]^{\mathrm{T}}$ and $\boldsymbol{x}_s = \boldsymbol{\alpha}_k$. By conditioning on $\boldsymbol{x}_s$, the model (2.49) becomes a polynomial of order two.

A linear measurement equation is considered, where the altitude is measured directly according to

$$\boldsymbol{z}_k = \boldsymbol{\alpha}_k + \boldsymbol{v}_k \, , \quad \boldsymbol{v}_k \sim \mathcal{N}\left(0, \sigma_v^2\right) .$$

Due to the linearity of the measurement equation, the measurement update can be performed via the Kalman filter.

In Table 2.1, the average rmses of 1000 Monte Carlo simulation runs for three Gaussian filters, namely the proposed SAGF, the UKF, and the Gaussian particle filter (GPF, [105]), are listed. In case of the GPF 100 and 1000 particles are employed. The SAGF provides the most accurate estimate for all three state variables. Only the GPF with 1000 particles can compete with the SAGF, which however comes with a high computational load for the GPF. In terms of run time, the SAGF is two times faster than the GPF with 100 particles and four times faster than the UKF.

2.5.3 Chebyshev Polynomial Kalman Filtering

As mentioned in the previous section, analytic moment matching is for instance possible for polynomial nonlinearities. To apply this fact more generally, the following contribution consists of a two-step approach to allow Gaussian filtering for arbitrary nonlinear functions: First, the given nonlinear function is expanded in a series of Chebyshev polynomials. In the second step, which is discussed in more detailed in Section 2.5.4, exact expressions for the moment integrals (2.8) are provided in a computationally efficient vector-matrix notation. This approach named *Chebyshev polynomial Kalman filter* (CPKF) facilitates function approximation and Bayesian filtering in a black-box fashion without the need of manual operations or manual inspection similar to LRKFs, but with a potentially higher estimation performance.

Chebyshev Polynomials

The key idea behind the CPKF is the approximation of the nonlinear function (2.6) by means of a *truncated Chebyshev series expansion* according to

$$g(\boldsymbol{x}) \approx \sum_{i=0}^{n} c_i \cdot T_i(\boldsymbol{x}) \tag{2.50}$$

on the interval $\Omega \triangleq [-1,1]$, where $T_n(x)$ are Chebyshev polynomials of the first kind, which are defined compactly as

$$T_n(x) = \cos(n \cdot \arccos x) \ , \quad i = 0,1,\ldots$$

or equivalently by means of the recursion

$$T_n(x) = 2x \cdot T_{n-1}(x) - T_{n-2}(x) \ , \quad n = 2,3,\ldots \ , \tag{2.51}$$

with initial conditions

$$T_0(x) = 1 \ , \quad T_1(x) = x \ . \tag{2.52}$$

It is easy to deduce from (2.51) that the function $T_n(x)$ is a polynomial of degree n. If n is even (odd), then $T_n(x)$ is a sum of even (odd) monomials, i.e., $T_n(x)$ is of the form

$$T_n(x) = \sum_{i=0}^{n} \alpha_{n,i} \cdot x^i = \sum_{j=0}^{\lfloor n/2 \rfloor} \alpha_{n,n-2j} \cdot x^{n-2j} \ , \tag{2.53}$$

where $\alpha_{n,i}$ is the *Chebyshev coefficient* of the ith monomial of the nth Chebyshev polynomial. The coefficient $\alpha_{n,i}$ is non-zero only if i is even (odd).

The quantities c_i in (2.50) are the *series coeffcients*

$$c_i = \frac{\langle g(x), T_i(x)\rangle}{\langle T_i(x), T_i(x)\rangle} \approx \tfrac{2-\delta_{0,n}}{n} \sum_{m=1}^{n} g(x_m) \cdot T_i(x_m) \tag{2.54}$$

for $i = 0,1,\ldots,n$. In (2.54) $\langle g(x), f(x)\rangle \triangleq \int_\Omega \left(1-x^2\right)^{-1/2} \cdot g(x) \cdot f(x)\,\mathrm{d}x$. The right-hand side follows from the discrete orthogonality property of the Chebyshev polynomials, where $x_m = \cos\left(\pi(m-0.5)/i\right) \in \Omega$, $m = 1\ldots i$, are the zeros of the Chebyshev polynomial $T_i(x)$ and $\delta_{i,j}$ is the Kronecker delta.

Employing polynomial series expansions in Bayesian filtering is not completely new. For instance higher-order Taylor-series expansion is employed for the second-order EKF [159, 173] or Fourier-Hermite series is used in Gaussian filtering in [160]. While the first approach is limited in terms of the order of the polynomial series, the second approach still requires numerical integration for moment calculation. Chebyshev series expansions are better suited for approximating nonlinear functions in the context of Gaussian filtering thanks to the following reasons:

- Chebyshev polynomials form a *complete orthogonal system* on Ω. As a consequence, the series coefficients (2.54) can be calculated independent of each other. This for instance is not true for Taylor-series expansions.
- In addition to the continuous orthogonality, Chebyshev polynomials are also *discrete orthogonal*. This property allows a very efficient calculation of the series coefficients by means of the well-known *discrete cosine transform* [30], for which a plethora of efficient algorithms exists. Similar approximate calculation schemes of series coefficients are typically not available for other orthogonal polynomials series like the Fourier-Hermite series.
- A truncated Chebyshev series satisfies the *near-minimax approximation* property, i.e., a truncated Chebyshev series of degree n is very close to the best possible polynomial approximation of the same degree. While the best polynomial representation of $g(.)$ is typically difficult to obtain, the Chebyshev series expansion is very close to the best solution and thanks to the discrete orthogonality very easy to calculate.

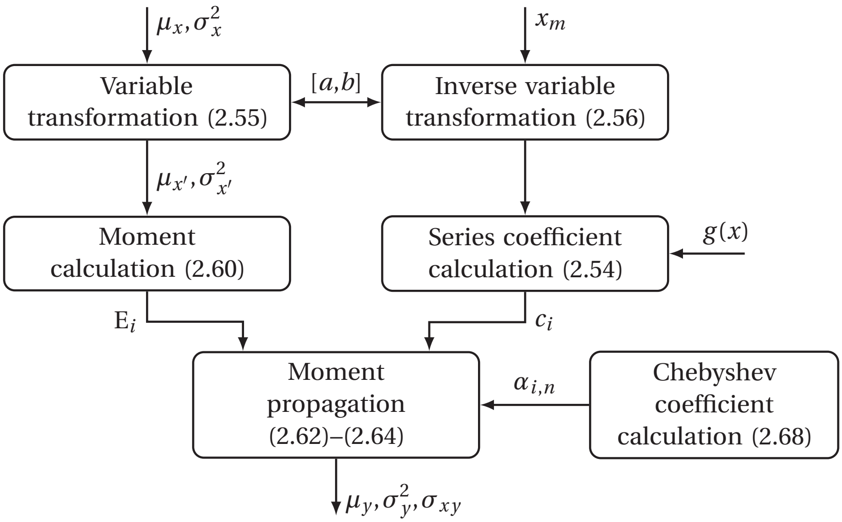

μ_x, σ_x^2
x_m
Variable transformation (2.55)
$[a,b]$
Inverse variable transformation (2.56)
$\mu_{x'}, \sigma_{x'}^2$
Moment calculation (2.60)
Series coefficient calculation (2.54)
$g(x)$
E_i
c_i
Moment propagation (2.62)–(2.64)
$\alpha_{i,n}$
Chebyshev coefficient calculation (2.68)
$\mu_y, \sigma_y^2, \sigma_{xy}$

Figure 2.5: Flow chart of closed-form moment propagation for Chebyshev polynomial Kalman filter.

More detailed information about Chebyshev polynomials and their properties can be found for instance in [119].

Structure

The building blocks of the CPKF are depicted in Figure 2.5. The blocks on the top are required due to the limitation that Chebyshev polynomials are only orthogonal on the interval Ω, while the function $g(.)$ can have an arbitrary support $[a,b] \subseteq \mathbb{R}$. Thus, the function $g(.)$ and the state $\boldsymbol{x}$ have to undergo first the affine transformation

$$x' = \tfrac{2}{b-a} \cdot x - \tfrac{a+b}{b-a} , \tag{2.55}$$

which yields a transformed Gaussian $\boldsymbol{x}' \sim \mathcal{N}\big(\mu_{x'}, \sigma_{x'}^2\big)$. Furthermore, the zeros x_m required for calculating the series coefficients (2.54) have to be mapped to the interval $[a,b]$, which is carried out by the inverse transformation

$$x = \tfrac{b-a}{2} \cdot x' + \tfrac{a+b}{2} . \tag{2.56}$$

of (2.55). By means of these transformations, arbitrary functions $g(.)$ can be treated by means of the CPKF.

The blocks “Moment calculation", “Moment propagation", and “Chebyshev coefficient calculation" are explained in detail in the following section.

2.5.4 Efficient Moment Propagation for Polynomials

At first a general polynomial representation of the function $g(.)$ according to

$$\boldsymbol{y} = g(\boldsymbol{x}) + \boldsymbol{w} = \sum_{i=0}^{n} c_i \cdot \boldsymbol{x}^i + \boldsymbol{w} \tag{2.57}$$

is considered. Chebyshev polynomials are treated as a special case at the end of this section.

General Solution

When propagating the Gaussian state $\boldsymbol{x}$ through the polynomial transformation $g(.)$ in (2.57), the mean of $\boldsymbol{y}$ can be expressed as

$$\mu_y = \mathrm{E}\{g(\boldsymbol{x})\} = \sum_{i=0}^{n} c_i \cdot \int x^i \cdot \mathcal{N}(x; \mu_x, \sigma_x^2)\, \mathrm{d}x = \sum_{i=0}^{n} c_i \cdot \underbrace{\mathrm{E}\{\boldsymbol{x}^i\}}_{\triangleq \mathrm{E}_i} . \tag{2.58}$$

Thus, the mean μ_y results in a weighted sum of non-central moments $\mathrm{E}_i = \mathrm{E}\{\boldsymbol{x}^i\}$ of order $i = 0,1,\ldots,n$. Formulae for calculating these moments of a Gaussian random vector are well-know (see for instance [116]), but require the evaluation of binomial coefficients, powers of the mean value, and weighted scalar products of the coefficient vector $\underline{\eta}$. Algebraically and computationally less demanding moment calculations can be found, however, when considering a special member of the exponential family in Section 2.1.3. By choosing the sufficient statistic to consist only of monomials of order up to n, i.e.,

$$\underline{\phi}(x) = \begin{bmatrix} 1 & x & x^2 & \cdots & x^n \end{bmatrix}^{\mathrm{T}} , \tag{2.59}$$

the following recursion proposed in [28] can be exploited. Although the moments of this special exponential density cannot be expressed in closed form, the ith order moment follows the recursion

$$\mathrm{E}_i = -\sum_{j=1}^{n} \frac{j}{i+1} \eta_j \, \mathrm{E}_{i+j} \ .$$

Thus, if the n lower-order moments $\underline{\mathrm{E}}_{0:n-1}^{\mathrm{T}} \triangleq [\mathrm{E}_0, \ldots, \mathrm{E}_{n-1}]$ are given and the moments up to E_m, $m \geq n$ are of interest, solving the linear system of equations

$$\mathbf{Q}\big(\underline{\eta}\big) \cdot \underline{\mathrm{E}}_{0:n-1} = \mathbf{R}\big(\underline{\eta}\big) \cdot \underline{\mathrm{E}}_{n:m} \tag{2.60}$$

gives the desired higher-order moments $\underline{\mathrm{E}}_{n:m}^{\mathrm{T}} \triangleq [\mathrm{E}_n \ \ldots \ \mathrm{E}_m]$. Here, $\mathbf{Q}\big(\underline{\eta}\big)$ is an rectangular matrix and $\mathbf{R}\big(\underline{\eta}\big)$ is a lower triangular matrix with elements (see [73])

$$Q_{i,j} = \begin{cases} 1 & \text{if } i = j \\ \frac{j-1}{i} \eta_{j-i} & \text{if } i < j \\ 0 & \text{otherwise} \end{cases} \ , \quad R_{i,j} = \begin{cases} 1 & \text{if } i - j = n \\ \frac{i-j-n}{i} \eta_{n+j-i} & \text{if } 0 \leq i - j < n \\ 0 & \text{otherwise} \end{cases} \ , \tag{2.61}$$

respectively. Thus, the matrix $\mathbf{R}\big(\underline{\eta}\big)$ is zero everywhere except of the main diagonal and the n diagonals below the main. Thanks to this special structure, the linear system of equations (2.60) can be efficiently solved by means of forward substitution.

The Gaussian density is a special case of (2.59) for $\underline{\phi}(x) = \big[1 \ x \ x^2\big]^{\mathrm{T}}$ where the first two moments $\mathrm{E}_0 = 1$ and $\mathrm{E}_1 = \mu_x$ are known. Hence, by solving (2.60), the mean calculation (2.58) becomes

$$\mu_y = \underline{c}_n^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:n} = \underline{c}_n^{\mathrm{T}} \cdot \begin{bmatrix} \mathbf{I}_2 \\ \mathbf{L} \end{bmatrix} \cdot \underline{\mathrm{E}}_{0:1} \ , \tag{2.62}$$

with $\mathbf{L} \triangleq \big(\mathbf{R}(\underline{\eta})\big)^{-1} \mathbf{Q}\big(\underline{\eta}\big)$, where the parameter vector $\underline{\eta}$ comprises the parameters of a (normalized) Gaussian density as defined in (2.5). Furthermore, $\underline{c}_n^{\mathrm{T}} \triangleq [c_0 \ c_1 \ \ldots \ c_n]$ is the vector of polynomial coefficients.

In a similar fashion, the variance σ_y^2 of $\boldsymbol{y}$ and the covariance σ_{xy} between $\boldsymbol{x}$ and $\boldsymbol{y}$ can be determined. The variance becomes

$$\begin{aligned}\sigma_y^2 &= \left(\underline{c}_n * \underline{c}_n\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:2n} - \mu_y^2 + \sigma_w^2 \\ &= \left(\mathbf{T} \cdot \underline{c}_n\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:2n} - \mu_y^2 + \sigma_w^2 \ ,\end{aligned} \tag{2.63}$$

where $*$ is the discrete convolution operator. The second equality indicates an efficient matrix-vector realization of the convolution by means of the matrix $\mathbf{T}$ with entries $t_{i,j} = t_{i+1,j+1} = c_{i-j}$ if $i \in [j, j+n]$ and $t_{i,j} = 0$ otherwise, where $i = 1,2,\ldots,2n+1$ and $j = 1,2,\ldots,n+1$. Hence, $\mathbf{T}$ is special type of matrix, namely a triangular *Toeplitz matrix* with only the mean diagonal and n diagonals below the main diagonal being non-zero and all elements on individual diagonals being equal.

The covariance can be simplified to

$$\sigma_{xy} = \underline{c}_n^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{1:n+1} - \mu_x \cdot \mu_y \ , \tag{2.64}$$

where μ_y is already known from (2.62). The first summand in (2.64) is almost identical to the mean calculation in (2.58) except for the shift by one in the order of the involved moments.

Given all the required moments, a Gaussian filter for polynomial nonlinearities is complete and listed in Algorithm 1. It is important to note that the polynomial order of the system function $a_k(.)$ and the measurement function $h_k(.)$ need not to be the same. Accordingly, the coefficient vectors $\underline{c}_{n_p}^{p}$ corresponding to the system function and $\underline{c}_{n_e}^{e}$ corresponding to the measurement function are of different dimension.

Example 6: Chaotic Synchronization

The proposed polynomial Kalman filter (PKF) is evaluated for the polynomial system model

$$\boldsymbol{x}_{k+1} = T_4(\boldsymbol{x}_k) + \boldsymbol{w}_k \tag{2.65}$$

as used in [116], where $T_i(x)$ is the ith Chebyshev polynomial (2.51). It is known that models as in (2.65) generate chaotic sequences [151], which are of practical use in securing communication systems. The true initial state $\boldsymbol{x}_0$ at time step $k = 0$ is assumed to be Gaussian with mean $\mu_0^x = 0.3$ and variance $\left(\sigma_0^x\right)^2 = 0.25$.

Algorithm 1 Polynomial Kalman Filter (PKF)

```
    ▷ *Prediction*
1:  Determine moment vector $\underline{\mathrm{E}}_{0:2n_p}$ of posterior state $\boldsymbol{x}_{k-1}^e$ by solving (2.60)
2:  Predicted mean: $\mu_k^p = \left(\underline{c}_{n_p}^p\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:n_p}$
3:  Predicted variance: $\left(\sigma_k^p\right)^2 = \left(\mathbf{T} \cdot \underline{c}_{n_p}^p\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:2n_p} - \left(\mu_k^p\right)^2 + \left(\sigma_k^w\right)^2$
    ▷ *Measurement Update*
4:  Determine moment vector $\underline{\mathrm{E}}_{0:2n_e}$ of predicted state $\boldsymbol{x}_k^p$ by solving (2.60)
5:  Measurement mean: $\mu_k^z = \left(\underline{c}_{n_e}^e\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:n_e}$
6:  Measurement variance: $\left(\sigma_k^z\right)^2 = \left(\mathbf{T} \cdot \underline{c}_{n_e}^e\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:2n_e} - \left(\mu_k^z\right)^2 + \left(\sigma_k^v\right)^2$
7:  Covariance: $\sigma_k^{xz} = \left(\underline{c}_{n_e}^e\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{1:n_e+1} - \mu_k^p \cdot \mu_k^z$
8:  Kalman gain: $K_k = \sigma_k^{xz} / \left(\sigma_k^z\right)^2$
9:  Posterior mean: $\mu_k^e = \mu_k^p + K_k \cdot \left(\hat{z}_k - \mu_k^z\right)$
10: Posterior variance: $\left(\sigma_k^e\right)^2 = \left(\sigma_k^p\right)^2 - K_k \cdot \sigma_k^{xz}$
```

Furthermore, a linear measurement model

$$\boldsymbol{z}_k = \boldsymbol{x}_k + \boldsymbol{v}_k \tag{2.66}$$

is employed, with measurement noise variance $(\sigma_v)^2 = 10^{-2} \cdot (\sigma_w)^2$ and system noise variance being $(\sigma_w)^2 = 10^{-2}$ (high noise) or $(\sigma_w)^2 = 10^{-3}$ (low noise). The PKF is compared against EKF, UKF, and a particle filter (PF) with systematic resampling [36] and 500 samples. The latter is the only non-Gaussian filter. For all filters, 50 Monte Carlo simulation runs with identical noise sequences are performed, where the estimates are calculated for 50 time steps.

In Table 2.2, the average rmse, nees, and runtime over all Monte Carlo runs are listed for all filters and for both noise cases. For high noise, the proposed PKF outperforms all Gaussian filters in terms of rmse and nees, i.e., its estimates are closest to the true system state (low rmse) and at the same time the estimates are not overly confident (low nees). Furthermore, the matrix-vector terms proposed for the PKF allow for a runtime being close to the EKF, which is known to be the fastest Gaussian filter.

Table 2.2: Average rmse, nees, and runtime for system model (2.65).

	$\sigma_w^2 = 10^{-2}$				$\sigma_w^2 = 10^{-3}$			
	EKF	UKF	PF	PKF	EKF	UKF	PF	PKF
rmse	0.410	0.336	**0.292**	0.316	0.148	**0.118**	0.268	**0.118**
nees	4.737	1.550	1.168	**1.041**	7.279	**1.110**	–	1.129
time	**0.016**	0.038	0.109	0.017	**0.017**	0.037	0.102	0.018

For the low noise case, UKF performs best in terms of estimation error, but PKF is very close to it. PF occasionally suffers from particle depletion, i.e., most of the particles converge towards the same state, which coincides with an overconfident estimate and thus an exceedingly high nees value. Even significantly increasing the number of particles or using different resampling techniques yields no improvement.

Special Case: Chebyshev Polynomials

For a truncated Chebyshev series expansion of an arbitrary nonlinear function $g(.)$ as in (2.50) the moments calculations (2.62)–(2.64) could be applied directly. Therefore, the Chebyshev series has to be transformed into the standard polynomial form as in (2.57), for which the so-called *Clenshaw algorithm* [44] can be used. This procedure, however, has severe drawbacks: Evaluating Chebyshev series in the standard form (2.57) requires significantly more algebraic operations as the sparse structure of the Chebyshev polynomials is no longer exploited. Furthermore, the polynomial coefficients in (2.57) can be large numbers as they are products of multiple Chebyshev polynomial coefficients, which by themself already can be significant. For instance, the leading coefficient of $T_i(x)$ is 2^{i-1}.

To avoid these issues, it is recommended to reformulate (2.62)–(2.64) by exploiting the recursive definition of the Chebyshev polynomials. The mean μ_y for instance can be expressed as

$$\mu_y = \mathrm{E}\{g(\boldsymbol{x})\} = \int g(x) \cdot \mathcal{N}(x; \mu_x, \sigma_x^2)\,\mathrm{d}x$$

$$\overset{(2.50),(2.53)}{\approx} \sum_{i=0}^{n} c_i \sum_{j=0}^{i} \alpha_{i,j} \int x^j \cdot \mathcal{N}(x; \mu_x, \sigma_x^2)\,\mathrm{d}x = \underline{c}_n^{\mathrm{T}} \cdot \mathbf{A}_n \cdot \underline{\mathrm{E}}_{0:n} \,. \qquad (2.67)$$

In contrast to (2.62), $\underline{c}_n \triangleq [c_0\ c_1\ \dots\ c_n]^{\mathrm{T}}$ now is the vector of series coefficients (2.54). Further, $\mathbf{A}_n$ is the $(n+1)\times(n+1)$ matrix of Chebyshev coefficients defined by

$$\mathbf{A}_n \triangleq \begin{bmatrix}\underline{\alpha}_{0,n} & \underline{\alpha}_{1,n} & \cdots & \underline{\alpha}_{n,n}\end{bmatrix}^{\mathrm{T}} .$$

Here, $\underline{\alpha}_{i,n} \triangleq [\alpha_{i,0}\ \alpha_{i,1}\ \dots\ \alpha_{i,n}]^{\mathrm{T}} \in \mathbb{N}^{n+1}$, $i = 0,1,\dots,n$ comprises all coefficients of the ith Chebyshev polynomial up to and including the nth monomial. It is calculated via the recursion

$$\underline{\alpha}_{i,n} = 2\cdot\Big[0\ \ \underline{\alpha}^{\mathrm{T}}_{i-1,i-1}\ \ \underbrace{0\ \dots\ 0}_{n-i\text{ times}}\Big]^{\mathrm{T}} + \Big[\underline{\alpha}^{\mathrm{T}}_{i-2,i-2}\ \ \underbrace{0\ \dots\ 0}_{n-i+2\text{ times}}\ \Big]^{\mathrm{T}} , \tag{2.68}$$

where the recursion commences from

$$\underline{\alpha}_{0,n} = \begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}^{\mathrm{T}} ,\quad \underline{\alpha}_{1,n} = \begin{bmatrix}0 & 1 & 0 & \dots & 0\end{bmatrix}^{\mathrm{T}}$$

and exploits the definition of the Chebyshev polynomials (2.51). According to (2.53), the coefficients $\alpha_{i,j}$ are zero for $j > i$. Thus, $\mathbf{A}_n$ is a sparse lower triangular matrix, which significantly reduces the computations of the matrix-vector products in (2.67).

Analogously, the variance σ_y^2 becomes

$$\sigma_y^2 \approx \left(\underline{c}_n \otimes \underline{c}_n\right)^{\mathrm{T}} \cdot \mathbf{P}_{2n} \cdot \underline{\mathrm{E}}_{0:2n} - \mu_y^2 + \sigma_w^2 , \tag{2.69}$$

with $\otimes$ being the Kronecker product and $\mathbf{P}_{2n}$ being an $(n+1)^2 \times (2n+1)$ matrix comprising the coefficients resulting from all possible products $T_i(x)\cdot T_j(x)$, $i,j = 0,1,\dots,n$ of the Chebyshev series expansion of $g(x)$.

Finally the covariance between $\boldsymbol{x}$ and $\boldsymbol{y}$ is given by

$$\sigma_{xy} = \underline{c}_n^{\mathrm{T}} \cdot \mathbf{A}_n^* \cdot \underline{\mathrm{E}}_{0:n+1} - \mu_x \cdot \mu_y , \tag{2.70}$$

where the $(n+1)\times(n+2)$ matrix $\mathbf{A}_n^*$ is given by

$$\mathbf{A}^*_{k,n} \triangleq \tfrac{1}{2}\left(\begin{bmatrix}\underline{0} & (b-a)\cdot\mathbf{A}_n\end{bmatrix} + \begin{bmatrix}(a+b)\cdot\mathbf{A}_n & \underline{0}\end{bmatrix}\right) .$$

This matrix includes the mapping back to the interval $[a,b]$ by means of the inverse variable transform (2.56).

Example 7: TV Commercial Effectiveness

In this example, real-world data from monitoring the advertising effectiveness of a TV commercial campaign for a single product is considered [123, 208]. This data is obtained by means of weekly surveys, where a given number of individuals from the population of TV viewers in UK is sampled in order to count the number being aware of current or recent TV commercials for the product. The result of each survey is measured in standardized units known as television ratings (TVRs) denoted by u_k.

The TVR measurements drive the nonlinear dynamics equation

$$\underline{\boldsymbol{x}}_{k+1} = \underline{a}\left(\underline{\boldsymbol{x}}_k + \underline{\boldsymbol{w}}_k, u_k\right) , \quad \underline{\boldsymbol{w}}_k \sim \mathcal{N}\left(\underline{0}, 0.03 \cdot \mathbf{C}_k^x\right) ,$$

with system function

$$\underline{a}(\underline{\boldsymbol{x}}, u) = \left[\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \boldsymbol{x}_3 \quad \boldsymbol{x}_4(\boldsymbol{x}_2 - \boldsymbol{x}_1) - (\boldsymbol{x}_2 - \boldsymbol{x}_1 - \boldsymbol{x}_3 \cdot \boldsymbol{x}_5) \cdot \exp(-\boldsymbol{x}_4 \cdot u)\right]^{\mathrm{T}} .$$

The state vector $\underline{x} \in \mathbb{R}^5$ comprises the minimum level of awareness $\boldsymbol{x}_1$, maximum level of awareness $\boldsymbol{x}_2$, memory decay rate $\boldsymbol{x}_3$, penetration $\boldsymbol{x}_4$, and effect of TVR on the awareness $\boldsymbol{x}_5$ (for details see [208]). The measurement equation is given by

$$\boldsymbol{z}_k = \boldsymbol{x}_{1,k} + \boldsymbol{x}_{5,k} + \boldsymbol{v}_k = \mathbf{H} \cdot \underline{\boldsymbol{x}}_k + \boldsymbol{v}_k , \quad \boldsymbol{v}_k \sim \mathcal{N}(0, 0.05)$$

with $\mathbf{H} \triangleq [1, 0, 0, 0, 1]$, where $\boldsymbol{z}_k$ corresponds to the awareness proportion. The initial state estimate is given by $\underline{\boldsymbol{x}}_0 \sim \mathcal{N}\left(\underline{\mu}_0^x; \mathbf{C}_0^x\right)$ with mean vector and covariance matrix

$$\underline{\mu}_0^x = \begin{bmatrix} 0.10 \\ 0.85 \\ 0.90 \\ 0.02 \\ 0.30 \end{bmatrix} \quad \text{and} \quad \mathbf{C}_0^x = \begin{bmatrix} 6.25 & 6.25 & 0 & 0 & 0 \\ 6.25 & 406.25 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2.25 & 0 \\ 0 & 0 & 0 & 0 & 100 \end{bmatrix} ,$$

respectively.

As the system state has dimension five, the nonlinear-nonlinear decomposition proposed in Section 2.5.2 is employed with $\boldsymbol{x}_a \triangleq \boldsymbol{x}_4$ and $\underline{\boldsymbol{x}}_s^{\mathrm{T}} \triangleq [\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ \boldsymbol{x}_3 \ \boldsymbol{x}_5]$. The sampled state $\underline{\boldsymbol{x}}_s$ is processed by means of the UKF. Ad-

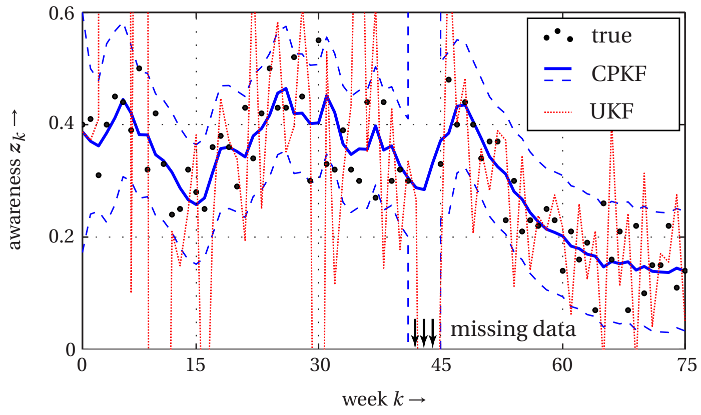


Figure 2.6: Predicted awareness proportions of the CPKF (blue solid line) with 95% confidence region (blue dashed) as well as the predictions of the UKF (red dotted). The true awareness proportion values are indicated by the black dots. For the weeks $k = 42, 43, 44$ no awareness measurements are available.

ditionally a UKF, where the unscented transform is applied to all five state dimensions is used.

The predictions of the awareness proportion $\boldsymbol{z}_k$ of the CPKF and UKF are compared before updating the state estimates with the true awareness value $\hat{z}_k$. The true awareness proportion values for performing the update step are taken from [208]. It is important to note that for the weeks 42, 43, and 44 no awareness measurements are available.

In Figure 2.6, the true and predicted awareness proportions are depicted. It is obvious that the UKF behaves very unsteady and is heavily fluctuating. The resulting awareness predictions are very inaccurate. This effect can be explained by overly confident estimates, i.e., the covariance matrix of the system state contains too small variances.

The behavior of the CPKF is different, which is surprising as the CPKF is merely applied on $\exp(-\boldsymbol{x}_4 \cdot u)$, while the remaining parts of the system equation are processed via the UKF. Thus, the CPKF has a stabilizing effect on the UKF resulting in awareness predictions that accurately follow the ground truth. Furthermore, the CPKF is not overconfident as the predicted measurement variances $\left(\sigma_k^z\right)^2$ are sufficiently large to capture the true awareness proportions. Even for the weeks with missing data, the predictions of CPKF are meaningful as the variances grow and thus, indicate an increasing uncertainty. Though, the trend is still correct.

2.5.5 Homotopic Moment Matching for Polynomial Measurement Models

Every Bayesian filter discussed so far in this chapter makes two different Gaussian assumptions. First, it assumes the predicted or posterior density to be Gaussian. Second, in order to perform the measurement update, it assumes that the joint density of state and measurement is Gaussian as well. The PKF for instance would be an exact Gaussian assumed density filter if only the first Gaussian assumption would be in place, as it performs moment matching, i.e., the mean and variance calculated by PKF coincide with the true mean and variance. The additional joint Gaussian assumption, however, can result in a poor approximation of the true mean and variance, which may cause a significant loss in estimation performance or even a divergence of the estimator.

Example 8: Joint Gaussian Flaw

To demonstrate the effect of the joint Gaussian assumption on the estimation performance, the polynomial measurement model

$$\boldsymbol{z} = \boldsymbol{x}^i + \boldsymbol{v} \tag{2.71}$$

is considered in the following, where $i > 0$ is even and the state is $\boldsymbol{x} \sim \mathcal{N}(0, \sigma_x^2)$. According to (2.62), (2.63), and (2.64), the mean μ_z, variance σ_z^2, and covariance σ_{xz} are given by

$$\mu_z = \mathrm{E}_i \ , \quad \sigma_z^2 = \mathrm{E}_{2i} - \mathrm{E}_i + \sigma_v^2 \ , \quad \sigma_{xz} = \mathrm{E}_{i+1} \ , \tag{2.72}$$

respectively. Since $\boldsymbol{x}$ has zero mean, it follows that all even moments of $\boldsymbol{x}$ are non-zero and all odd moments are zero, i.e., $\mathrm{E}_i \neq 0$ and $\mathrm{E}_{i+1} = 0$ for all i being even. Hence, the covariance σ_{xz} in (2.72) is zero. As a result, the state $\boldsymbol{x}$ and measurement $\boldsymbol{z}$ are uncorrelated and the joint Gaussian of state and measurement is axis-aligned. In Figure 2.7a, the joint Gaussian for $i = 2$, $\sigma_x^2 = 1$, and $\sigma_v^2 = 0.1$ is depicted.

As the covariance σ_{xz} is zero, the Kalman gain K in line 8 of Algorithm 1 is zero as well and no update of the predicted state occurs, i.e., the posterior state $\boldsymbol{x}^e$ is identical to the predicted state $\boldsymbol{x}^p$. In this case, a given measurement value has no impact on the estimation.

Without the joint Gaussian assumption the missing update will not occur. In order to demonstrate this, the measurement update is now viewed from a full Bayesian perspective. Here, the posterior state $\boldsymbol{x}$ is represented by the posterior density $f^e(x)$ according to Bayes' rule (recall (1.7))

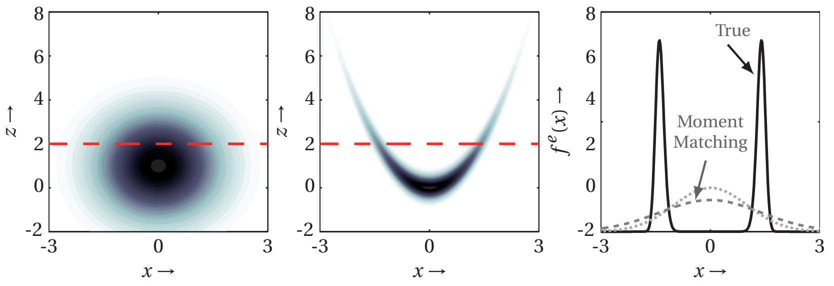
$$f^e(x) = \frac{f(z|x) \cdot f^p(x)}{f(z)} = \frac{f(x,z)}{f(z)} , \tag{2.73}$$

with the predicted Gaussian density $f^p(x) = \mathcal{N}\big(x; \mu^p, (\sigma^p)^2\big)$.

For the considered model (2.71) with a state $\boldsymbol{x}$ having zero mean, the joint Gaussian assumption leads to a factorization of the joint density $f(x,z) = f^p(x) \cdot f(z)$ as $\boldsymbol{x}$ and $\boldsymbol{z}$ are uncorrelated, which is equivalent to independence for Gaussian random variables. Hence, the Bayesian update in (2.73) degenerates to $f^e(x) = f^p(x)$. Actually, the joint density $f(x,z)$ is an exponential density with monomial sufficient statistics according to (2.59). This follows from the fact that the likelihood $f(z|x) = \mathcal{N}\big(z; x^i, \sigma_v^2\big)$ according to (1.8). The product of likelihood and prior density leads to the exponential density

$$\begin{aligned} f(x,z) &= f(z|x) \cdot f(x) = \mathcal{N}\big(z; x^i, \sigma_v^2\big) \cdot \mathcal{N}\big(x; 0, \sigma_x^2\big) \\ &= \exp\Big(-\log(2\pi\sigma_x\sigma_v) - \tfrac{1}{2\sigma_v^2} \cdot \big(z^2 + \tfrac{\sigma_v^2}{\sigma_x^2} x^2 - 2zx^i + x^{2i}\big)\Big) . \end{aligned} \tag{2.74}$$

This exponential joint density is depicted in Figure 2.7b for $i = 2$. By comparing Figure 2.7a with Figure 2.7b the difference between the true joint density and its Gaussian approximation becomes apparent. Given a measurement value $\hat{z} = 2$, Figure 2.7c depicts the posterior densities obtained for the Gaussian joint density and the true exponential joint density. It can be seen that the true posterior is bimodal, which only can be coarsely approximated by a Gaussian density. Furthermore, due to the joint Gaussian assumption, the Gaussian posterior does not even match the true posterior mean and variance.



(a) Gaussian approximation of the joint density.

(b) True joint density.

(c) True posterior density (black) and Gaussian approximations.

Figure 2.7: Joint density $f(x,z)$ and posterior density $f^e(x)$ for $i = 2$, i.e., for a quadratic polynomial. The red line indicates the measurement value $\hat{z} = 2$. In (c), one Gaussian approximation is obtained based on the joint Gaussian assumption (dotted) and the other via moment matching (dashed), i.e., its mean and variance coincide with the true posterior moments.

Homotopy Continuation

To overcome the limitations of the joint Gaussian assumption, a new method for directly calculating the moments of the posterior density for *polynomial measurement models* $h(x) = \sum_{i=0}^{n_e} c_i^e \cdot x^i$ according to (2.57) will be introduced. This method does not require the joint Gaussian assumption and provides almost exact posterior mean and variance.

The key idea is to transform the known moments of the prior Gaussian density continuously into the desired posterior moments. For this purpose, *homotopy continuation* for calculating the moments of exponential densities as proposed in [146] is exploited. By means of a so-called progression parameter $\gamma \in [0\ 1]$ the posterior density $f^e(x)$ is parameterized in such a way that for $\gamma = 0$ the posterior density corresponds to prior Gaussian density $f^p(x)$ and for $\gamma = 1$ the posterior density corresponds to the true exponential density. For the initial value $\gamma = 0$, the moments are known as they coincide with the moments of the Gaussian prior. Incrementing the progression parameter causes moment variations described by means of a *system of ordinary differential equations* (ODEs). Solving this system of ODEs for $\gamma \in [0\ 1]$ gives the desired posterior moments.

To allow for homotopy continuation, the Bayesian measurement update is parametrized according to[5]

$$f^e(x;\gamma) \propto f\big(x,\hat{z};\underline{\eta}(\gamma)\big) \triangleq f(\hat{z}|x)^\gamma \cdot f^p(x) \tag{2.75}$$

for a given measurement value $\hat{z}$. The *parametrized joint density* $f\big(x,\hat{z};\underline{\eta}(\gamma)\big) = \exp\big(\underline{\eta}(\gamma)^{\mathrm{T}} \cdot \underline{\phi}(x)\big)$ with $\underline{\phi}(x) \in \mathbb{R}^{2n_e+1}$ as in (2.59) is an exponential density similar to (2.74). The parameter vector is defined as

$$\underline{\eta}(\gamma) \triangleq \underline{\eta}^p + \gamma \cdot \underline{\eta}^l \in \mathbb{R}^{2n_e+1}$$

depending on γ. Here, $\underline{\eta}^p$ is the parameter vector of the Gaussian prior $f^p(x)$ and $\underline{\eta}^l$ is the parameter vector of the likelihood $f(\hat{z}|x)$.

In (2.75), $f^e(x;\gamma)$ is a parametrized version of the posterior density. For $\gamma = 1$, this parametrized measurement update corresponds to the standard Bayes' rule, while for $\gamma = 0$, the prior density $f^p(x)$ is directly assigned to the posterior density, i.e., no measurement update is performed.

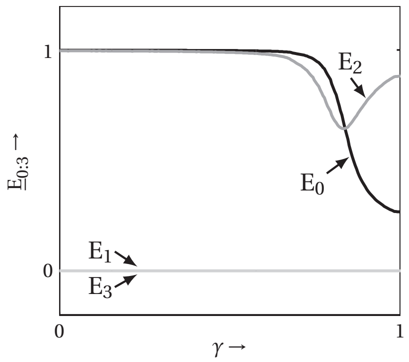
System of Ordinary Differential Equations

By a continuous modification of the progression parameter γ, a continuous variation of the parameter vector $\underline{\eta}(\gamma)$ is achieved. This in turn results in a variation of the moments $\mathrm{E}_i\big(\underline{\eta}(\gamma)\big)$, $i = 0,\ldots,2n_e - 1$, of the parametrized joint density $f(x,\hat{z};\underline{\eta}(\gamma))$. These moment variations depending on γ can be described by means of a system of ODEs by calculating the partial derivatives $\dot{\mathrm{E}}_i \triangleq \partial \mathrm{E}_i\big(\underline{\eta}(\gamma)\big)/\partial\gamma$ for $i = 0,\ldots,2n_e - 1$. The partial derivative of the ith-order moment is given by

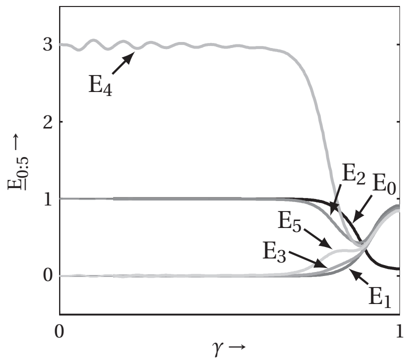
$$\dot{\mathrm{E}}_i = \frac{\partial \mathrm{E}_i\big(\underline{\eta}(\gamma)\big)}{\partial\gamma} = \Big[\mathrm{E}_i\big(\underline{\eta}(\gamma)\big) \quad \mathrm{E}_{i+1}\big(\underline{\eta}(\gamma)\big) \quad \cdots \quad \mathrm{E}_{i+2n_e}\big(\underline{\eta}(\gamma)\big)\Big] \cdot \underline{\eta}^l\,, \tag{2.76}$$

which relates the variation of the ith-order moment to moments of order up to $i + 2n_e$. In the following, $\mathrm{E}_i^{(\gamma)} \triangleq \mathrm{E}_i\big(\underline{\eta}(\gamma)\big)$ is used as shorthand term.

5 To simplify the following calculations merely the proportional relation is considered. The normalization constant $1/\mathrm{E}_0 = 1/f(\hat{z})$ can be incorporated ex post without any disadvantages.



(a) Quadratic model $\boldsymbol{z} = \boldsymbol{x}^2 + \boldsymbol{v}$.



(b) Cubic model $\boldsymbol{z} = \boldsymbol{x}^3 + \boldsymbol{v}$.

Figure 2.8: Trajectories of posterior moments.

With the result in (2.76), the system of ODEs comprising the moment variations of all moments up to order $2n_e - 1$ is

$$\underline{\dot{\mathrm{E}}}_{0:2n_e-1} = \left(\mathbf{T}\left(\underline{\eta}^l\right)\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}^{(\gamma)}_{0:4n_e-1} = \mathbf{T}^l \cdot \underline{\mathrm{E}}^{(\gamma)}_{0:2n_e-1} + \mathbf{T}^h \cdot \underline{\mathrm{E}}^{(\gamma)}_{2n_e:4n_e-1} ,$$

where $\mathbf{T}(\underline{\eta}^l) = \left[\mathbf{T}^l\ \mathbf{T}^h\right]^{\mathrm{T}}$ is a Toeplitz matrix with entries $t_{i,j} = t_{i+1,j+1} = \eta^l_{i-j}$ if $i \in [j, j+2n_e]$ and $t_{i,j} = 0$ otherwise, where $i = 1,2,\ldots,4n_e$ and $j = 1,2,\ldots,2n_e$. Besides the lower-order moments $\underline{\mathrm{E}}^{(\gamma)}_{0:2n_e-1}$, the system of ODEs also depends on the higher-order moments $\underline{\mathrm{E}}^{(\gamma)}_{2n_e:4n_e-1}$. Fortunately, with the result of (2.60), the dependence on the higher-order moments can be resolved. In doing so, the system of ODEs can be reformulated into

$$\underline{\dot{\mathrm{E}}}_{0:2n_e-1} = \left(\mathbf{T}^l + \mathbf{T}^h \left(\mathbf{R}\left(\underline{\eta}(\gamma)\right)\right)^{-1} \mathbf{Q}\left(\underline{\eta}(\gamma)\right)\right) \cdot \underline{\mathrm{E}}^{(\gamma)}_{0:2n_e-1} \tag{2.77}$$

with starting solution $\underline{\mathrm{E}}^{(0)0:2n_e-1}$ comprising the predicted moments. The matrices $\mathbf{R}(\underline{\eta}(\gamma))$ and $\mathbf{Q}(\underline{\eta}(\gamma))$ corresponding to (2.61), which vary with γ as they depend on the parameters of the parametrized joint density $f\left(x,\hat{z};\underline{\eta}(\gamma)\right)$.

The system of ODEs in (2.77) describes the moment variations caused by homotopy continuation of the Bayesian measurement update (2.75) in a very elegant manner. For solving this system of ODEs, standard numerical solvers based on

Algorithm 2 Homotopic Polynomial Gaussian Filter (HPGF)

```
    ▷ Prediction
 1: Determine moment vector $\underline{\mathrm{E}}_{0:2n_p}$ of posterior state $\boldsymbol{x}_{k-1}^e$ by solving (2.60)
 2: Predicted mean: $\mu_k^p = \left(\underline{c}_{n_p}^p\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:n_p}$
 3: Predicted variance: $\left(\sigma_k^p\right)^2 = \left(\mathbf{T} \cdot \underline{c}_{n_p}^p\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:2n_p} - \left(\mu_k^p\right)^2 + \left(\sigma_k^w\right)^2$
    ▷ Measurement Update
 4: Determine initial solution via first-order Taylor-series expansion around $\gamma = 0$
 5: Solve system of ODEs (2.77) for $\gamma \in [\Delta\gamma; 1]$ with $\Delta\gamma \ll 1$
 6: Calculate posterior mean $\mu_k^e = \alpha \cdot \mathrm{E}_1^{(1)}$
 7: Calculate posterior variance $\left(\sigma_k^e\right)^2 = \alpha \cdot \mathrm{E}_2^{(1)} - \left(\mu_k^e\right)^2$
```

the Runge-Kutta method [139] can be employed. The solution describes a trajectory of the moments $\underline{\mathrm{E}}_{0:2n_e-1}^{(\gamma)}$ depending on different values of the progression parameter γ. The desired moments of the posterior density $f^e(x)$ are obtained for $\gamma = 1$, i.e., $\underline{\mathrm{E}}_{0:2n_e-1}^{(1)}$ comprises the result.

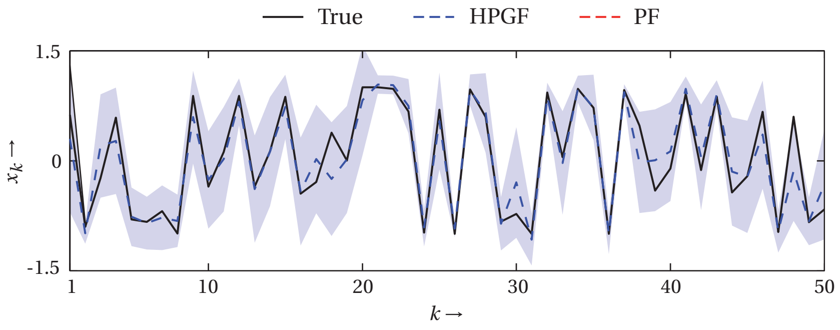
Example 9: Moment Trajectories

The polynomial measurement model of Example 8 is revisited, where now merely the quadratic (order $i = 2$) and the cubic (order $i = 3$) case are considered. Furthermore, the state $\boldsymbol{x} \sim \mathcal{N}(0,1)$ is standard Gaussian distributed, the measurement value is $\hat{z} = 1$, and $(\sigma_v)^2 = 0.1$ is the variance of the measurement noise. In Figure 2.8 on the previous page, the trajectories of the posterior moments resulting from the homotopy continuation are shown. It can be seen how the moments of the prior Gaussian are transformed into the true posterior moments.

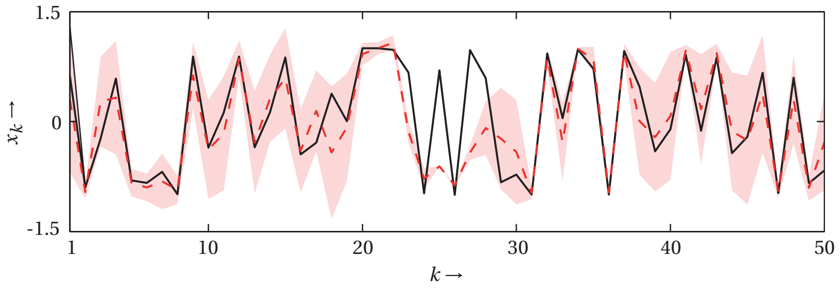
As mentioned above, the moments in $\underline{\mathrm{E}}_{0:2n_e-1}^{(1)}$ are unnormalized as merely the proportional relation (2.75) was considered. Multiplying $\underline{\mathrm{E}}_{0:2n_e-1}^{(1)}$ with the normalization constant

$$\alpha \triangleq \tfrac{1}{f(\hat{z})} = \tfrac{1}{\mathrm{E}_0^{(1)}}$$

yields the actual posterior moments. The entire Gaussian filter employing moment homotopy is listed in Algorithm 2 and is named *homotopic polynomial*



(a) Homotopic polynomial Gaussian filter (HPGF).



(b) Particle filter (PF).

Figure 2.9: State trajectory (black, solid line) and the estimates of HPGF and PF together with the corresponding 2-sigma confidence regions.

Gaussian filter (HPGF). The prediction step coincides with the prediction of the PKF since the PKF provides the exact predicted mean and variance. Before solving the ODE (2.77), a initialization step is required in line 4 as the matrix $\mathbf{R}\big(\underline{\eta}(\gamma)\big)$ is singular for $\gamma = 0$. Hence, the ODE is merely solved on the interval $[\Delta\gamma, 1]$, where $\Delta\gamma$ is a very small positive value. The posterior mean and variance calculated in line 6 and 7, respectively, are almost exact.

Example 10: Chaos Synchronization (Cont'd)

The estimation problem of Example 6 is revisited, but instead of the linear measurement model (2.66) the cubic measurement model

$$\boldsymbol{z}_k = \frac{\boldsymbol{x}_k^3}{20} + \boldsymbol{v}_k\,, \quad \text{with } \boldsymbol{v}_k \sim \mathcal{N}\big(0, 10^{-5}\big)$$

is employed. Given this measurement model, it turns out that *all* Gaussian filters relying on the joint Gaussian assumption diverge. The HPGF, however, is able to provide valid estimates. In Figure 2.9a on the previous page, an exemplary state trajectory is depicted. The estimates of HPGF accurately follow the true state. Furthermore, the true state is always within the 2-sigma confidence region of the estimates. The result of the PF depicted in Figure 2.9b is less accurate and shows sample depletion from time step $k = 20$ to $k = 27$.

2.6 Summary

In this chapter, the state-of-the-art in Gaussian filtering has been reviewed. Essentially, all Gaussian filters aim at approximating three particular moment integrals. Approximation techniques applied for this purpose are for instance linearization via Taylor-series expansions or deterministic sampling. The introduced approximation errors can be minimized by means of exploiting Rao-Blackwellization. This is starting point for the contributions made in this chapter:

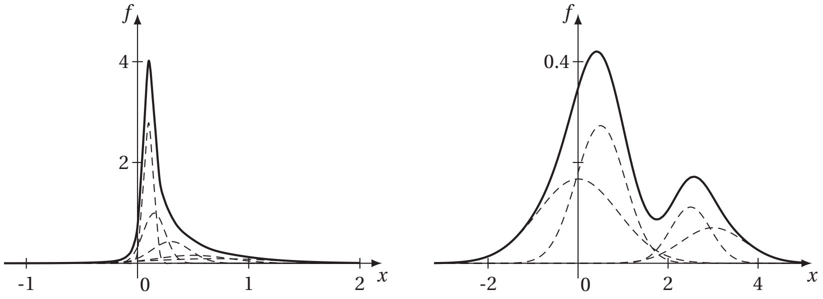
- *Combining Rao-Blackwellization with decomposition of observed and unobserved states:* Unobserved state variables can be excluded from the approximation which reduces the computational load and the approximation error.
- *Rao-Blackwellization for nonlinear-nonlinear decomposition:* Instead of the commonly employed linear-nonlinear decomposition, a novel nonlinear-nonlinear decomposition is proposed. Therefore the concept of conditionally integrable functions was introduced, i.e., functions that comprise a nonlinear substructure for which analytic moment expressions exist.

- *Approximation of arbitrary nonlinear models with Chebyshev polynomials:* Polynomials are one class of functions for which analytic moment matching is possible. Chebyshev polynomials are well suited for transforming arbitrary nonlinearities into polynomials thanks to their orthogonality, sparseness, and efficient coefficient calculation procedure.
- *Closed-form moment calculation for polynomial nonlinearities*: Novel matrix-vector expressions for analytical moment calculation for polynomial dynamic and measurement models have been proposed. These expressions allow exact predictions for Gaussian filters.
- *Almost exact posterior moments by homotopy continuation:* For polynomial measurement models the joint Gaussian assumption can be avoided. For this purpose a homotopy continuation method was proposed that yields almost optimal posterior moments.

Every contribution on its own can significantly improve the estimation performance. A boosting in performance, however, can be obtained by exploiting the strong interrelations between the several methods. For instance, after employing the aforementioned decompositions there will remain nonlinear substructures for which no analytic moment matching is possible. However, approximating these nonlinearities with Chebyshev polynomials allows accurate Gaussian filtering for the entire nonlinear model as shown in Example 7. Given a polynomial measurement model—either polynomial by definition, generated via Chebyshev series expansion, or as a resulting substructure after Rao-Blackwellization—the novel moment homotopy can be employed.

3 Gaussian Mixture Filtering

Although Gaussian filters show a good estimation performance in many practical applications, they are clearly limited when the true distribution of the state takes a complex shape, e.g., multiple modes, strong skewness, or heavy tailes. Such situations may for instance arise in multi-target tracking or financial forecasts. To also provide a consistent filter in these applications, a natural extension of Gaussian filtering is Gaussian mixture filtering. In this chapter, a brief introduction to Gaussian mixture densities is given at first. Then the extension of the Gaussian filters discussed in Section 3.2 is derived. Due to the usage of multiple Gaussians, Gaussian mixtures filters require additional operations regarding the adaptation of the number of mixture components. A statement of this additional problem is given in the Section 3.3. The contributions made by the Papers E–G are summarized in Section 3.4.



(a) Heavily skewed density consisting of five components.

(b) Multimodal density consisting of four components.

Figure 3.1: Two exemplary Gaussian mixture densities. The individual Gaussian components are plotted as dashed lines.

3.1 Gaussian Mixtures

A Gaussian mixture density is defined as a weighted sum of Gaussian densities according to

$$f(\underline{x}) = \sum_{i=1}^{L} \omega_i \cdot \mathcal{N}(\underline{x}; \underline{\mu}_i, \mathbf{C}_i) \tag{3.1}$$

with non-negative weighting coefficients ω_i that sum up to one. In Figure 3.1, some examples of Gaussian mixture densities are shown. Sometimes this density type is also called Gaussian sum density or mixture of Gaussians. Obviously, the Gaussian density is a special case of (3.1) for $L = 1$. By defining the two vectors $\underline{\omega} \triangleq [\omega_1 \; \ldots \; \omega_L]^{\mathrm{T}}$ and $\underline{f}(\underline{x}) \triangleq [\mathcal{N}(\underline{x}; \underline{\mu}_1, \mathbf{C}_1) \; \ldots \; \mathcal{N}(\underline{x}; \underline{\mu}_L, \mathbf{C}_L)]^{\mathrm{T}}$, a more compact version of (3.1) can be found via

$$f(\underline{x}) = \underline{\omega}^{\mathrm{T}} \cdot \underline{f}(\underline{x}) \,. \tag{3.2}$$

There are mainly three reasons for the widespread use of Gaussian mixtures in Bayesian filtering: First, as they consist of multiple Gaussians, they possess a straightforward parametrization by means of the weights ω_i, mean vectors $\underline{\mu}_i$ and covariance matrices $\mathbf{C}_i$ of the individual Gaussian components. Second, the practically relevant moments mean and covariance of the Gaussian mixture can be calculated in closed form by means of

$$\begin{aligned}\underline{\mu}_x &= \sum_{i=1}^{L} \omega_i \cdot \underline{\mu}_i\ , \\ \mathbf{C}_x &= \sum_{i=1}^{L} \omega_i \cdot \left(\mathbf{C}_i + \underline{\mu}_i \cdot \underline{\mu}_i^{\mathrm{T}}\right) - \underline{\mu}_x \cdot \underline{\mu}_x^{\mathrm{T}}\ .\end{aligned}$$

And finally, any continuous density function $\tilde{f}(\underline{x})$ can be approximated by means of a Gaussian mixture as closely as required with respect to the *Lissack-Fu* distance

$$\int \left|\tilde{f}(\underline{x}) - f(\underline{x})\right| \mathrm{d}\underline{x}$$

by increasing the number of components L and when $\mathbf{C}_i$ approaches the zero matrix [7, 180]. Thus, a Gaussian mixture density can be considered as *universal approximator* for density functions [121].

3.2 Nonlinear Filtering

For a Gaussian mixture filter it is assumed that both the predicted and the posterior density of the state are represented as Gaussian mixtures according to

$$f_k^\bullet(\underline{x}_k) = \sum_{i=1}^{L_k^\bullet} \omega_{k,i}^\bullet \cdot \mathcal{N}\left(\underline{x}_k; \underline{\hat{x}}_{k,i}^\bullet, \mathbf{C}_{k,i}^\bullet\right)\ , \text{ with } \bullet \in \{e, p\}\ , \tag{3.3}$$

for every time step k, where $L_k^\bullet$ is the number of mixture components. To calculate the parameters of these densities, several approaches exist that can be grouped in three major classes as depicted in Figure 3.2. In *model approximating* approaches, the transition density (1.6) and the likelihood (1.8) are approximated by means of a Gaussian mixture, which is typically done off-line, before the actual filtering. For instance [83, 132] propose techniques for approximating the transition density, while likelihood approximation is content of [5, 86, 191].

Density approximation approaches instead focus on directly approximating the true predicted or posterior density by means of Gaussian mixtures. Here, one can distinguish between *joint approximation*, where all Gaussian components

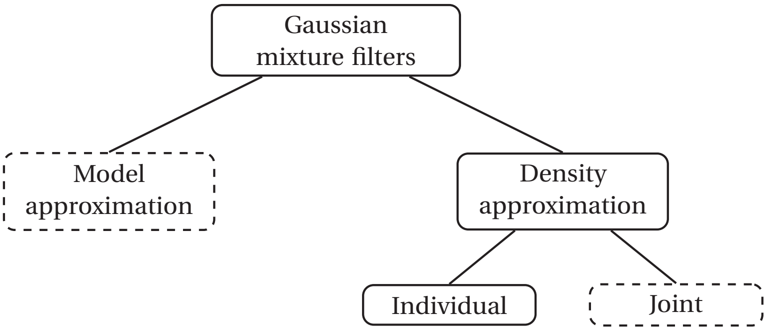


Figure 3.2: Taxonomy for Gaussian mixture filters. The dashed classes are not considered in this thesis.

and their respective parameters are calculated jointly, typically by solving an optimization problem as proposed in [75, 76]. In case of *individual approximations*, each Gaussian component is processed separately through the prediction and measurement update.

3.2.1 Individual Approximation

In this thesis, the focus is on individual approximation techniques as they allow a direct utilization of the Gaussian filtering algorithms discussed in the previous chapter. In doing so, the calculation of the desired parameters of the predicted and posterior Gaussian mixtures can be performed efficiently without any demanding off-line approximations. In the following, the prediction and measurement update of a generic Gaussian mixture filter based on individual approximation are derived.

Prediction

For calculating the predicted density, the Gaussian mixture representing the posterior distribution $f_k^e(\underline{x}_k)$ is plugged into (1.5), which yields

$$f_{k+1}^p(\underline{x}_{k+1}) = \int f(\underline{x}_{k+1}|\underline{x}_k, \underline{u}_k) \cdot \left(\sum_{i=1}^{L_k^e} \omega_{k,i}^e \cdot \mathcal{N}\left(\underline{x}_k; \underline{\hat{x}}_{k,i}^e, \mathbf{C}_{k,i}^e\right) \right) \mathrm{d}\underline{x}_k$$

$$\approx \sum_{i=1}^{L_k^e} \underbrace{\omega_{k,i}^e}_{\equiv\, \omega_{k+1,i}^p} \cdot \underbrace{\int f\left(\underline{x}_{k+1} | \underline{x}_k, \underline{u}_k\right) \cdot \mathcal{N}\left(\underline{x}_k; \hat{\underline{x}}_{k,i}^e, \mathbf{C}_{k,i}^e\right) \mathrm{d}\underline{x}_k}_{\approx\, \mathcal{N}\left(\underline{x}_{k+1}; \underline{\mu}_{k+1,i}^p, \mathbf{C}_{k+1,i}^p\right)} \, . \tag{3.4}$$

with $L_k^p \equiv L_k^e$. The integral cannot be solved in closed form except for the linear case. To simplify the integration, for each individual component of the posterior mixture the solution of the integral is approximated by means of the Gaussian $\mathcal{N}\left(\underline{x}_{k+1}; \underline{\mu}_{k+1,i}^p, \mathbf{C}_{k+1,i}^p\right)$. Thus, it remains to calculate the mean vector $\underline{\mu}_{k+1,i}^p$ and covariance matrix $\mathbf{C}_{k+1,i}^p$. For this purpose any of the Gaussian filters of Chapter 2 can be employed. The weights remain unchanged. The resulting predicted density $f_{k+1}^p\left(\underline{x}_{k+1}\right)$ is then again a Gaussian mixture.

Measurement Update

In case of the measurement update, the predicted mixture (3.4) is substituted in (1.7) according to

$$f_k^e\left(\underline{x}_k\right) = c_k \cdot f\left(\hat{\underline{z}}_k | \underline{x}_k\right) \cdot \left(\sum_{i=1}^{L_k^p} \omega_{k,i}^p \cdot \mathcal{N}\left(\underline{x}_k; \underline{\mu}_{k,i}^p, \mathbf{C}_{k,i}^p\right) \right) \tag{3.5}$$

Due to the nonlinearity of the measurement equation, the measurement update cannot be solved analytically, i.e., the normalization constant as well as the product of the likelihood with the predicted mixture possess no closed-form expression in general.

The normalization constant $c_k = {}^1/_{f(\hat{\underline{z}}_k)}$ corresponds to the reciprocal probability of the measurement. This probability can be approximated as in the prediction step according to

$$\begin{aligned} f\left(\hat{\underline{z}}_k\right) = \frac{1}{c_k} &= \int f\left(\hat{\underline{z}}_k | \underline{x}_k\right) \cdot \left(\sum_{i=1}^{L_k^p} \omega_{k,i}^p \cdot \mathcal{N}\left(\underline{x}_k; \underline{\mu}_{k,i}^p, \mathbf{C}_{k,i}^p\right) \right) \mathrm{d}\underline{x}_k \\ &\approx \sum_{i=1}^{L_k^p} \omega_{k,i}^p \cdot \mathcal{N}\left(\hat{\underline{z}}_k; \underline{\mu}_{k,i}^z, \mathbf{C}_{k,i}^z\right) , \end{aligned} \tag{3.6}$$

where the parameters $\underline{\mu}_{k,i}^z$ $\mathbf{C}_{k,i}^z$ are determined individually by means of a Gaussian filter.

To approximate the product between likelihood and predicted mixture in (3.5), the equation is extended with the components $\mathcal{N}\big(\underline{\hat{z}}_k; \underline{\mu}^z_{k,i}, \mathbf{C}^z_{k,i}\big)$ from (3.6), which results in

$$f^e_k(\underline{x}_k) \approx \sum_{i=1}^{L^p_k} \underbrace{c_k \cdot \omega^p_{k,i} \cdot \mathcal{N}\Big(\underline{\hat{z}}_k; \underline{\mu}^z_{k,i}, \mathbf{C}^z_{k,i}\Big)}_{\triangleq\, \omega^e_{k,i}} \cdot \underbrace{\frac{f(\underline{\hat{z}}_k|\underline{x}_k) \cdot \mathcal{N}\Big(\underline{x}_k; \underline{\mu}^p_{k,i}, \mathbf{C}^p_{k,i}\Big)}{\mathcal{N}\Big(\underline{\hat{z}}; \underline{\mu}^z_{k,i}, \mathbf{C}^z_{k,i}\Big)}}_{\approx\, \mathcal{N}\big(\underline{x}_k; \underline{\mu}^e_{k,i}, \mathbf{C}^e_{k,i}\big)} \tag{3.7}$$

with $L^p_k \equiv L^e_k$. The fraction in (3.7) corresponds to a Bayesian measurement update for each individual predicted Gaussian and is approximated with a Gaussian density $\mathcal{N}\big(\underline{x}_k; \underline{\mu}^e_{k,i}, \mathbf{C}^e_{k,i}\big)$. The mean vector and covariance matrix of these individual posterior Gaussian components are determined by means of a Gaussian filter by employing the conditioning in (2.9). Therefore, the parameters $\underline{\mu}^z_{k,i}$ $\mathbf{C}^z_{k,i}$ are required, which are already available from (3.6). Only the cross-covariance matrices $\mathbf{C}^{xz}_{k,i}$ need to be calculated in addition.

3.2.2 Generic Gaussian Mixture Filter

Thanks to the individual approximation, both the prediction and measurement update of the Gaussian mixture filter boil down to a *bank of Gaussian filters*, where each individual Gaussian filter tracks the evolution of its assigned Gaussian component. The Gaussian mixture in each estimation step is the linear combination of the individual results [73].

Besides the pure estimation steps, a generic Gaussian mixture filter requires additional operations for maintaining an accurate density approximation and an adequate run-time. In Algorithm 3, these additional operations are listed for both prediction and measurement update. The *refinement* replaces components of the given Gaussian mixture with one or more new components. This might be necessary to overcome strong nonlinearities in the system model or measurement model. As all Gaussian filters either explicitly or implicitly perform a linearization, the linearization error can be reduced by refining components [3, 63, 145].

The *reapproximation* comprises two operations: weight optimization and reduction. The individual weights of the posterior Gaussian components in (3.7) are merely an approximation of the true weights. This observation follows from the approximate calculation of the normalization in (3.6). The normalization is

Algorithm 3 Generic Gaussian mixture filter

```
1: Initialize state density $f_0(\underline{x})$ with a Gaussian mixture
2: for each time step $k$ do
     ▷ Prediction
3:     Refinement: Introduce additional components
4:     Estimation: Compute predicted Gaussian mixture $f_k^p(\underline{x}_k)$
5:     Reapproximation: Weight optimization and component reduction
     ▷ Measurement Update
6:     Refinement: Introduce additional components
7:     Estimation: Compute posterior Gaussian mixture $f_k^e(\underline{x}_k)$
8:     Reapproximation: Weight optimization and component reduction
9: end for
```

one factor forming the posterior weights. To improve the weights, on additional *weight optimization* can be performed after the measurement update.

The *reduction* step removes Gaussian components from the predicted and posterior mixture. The need for this operation has many reasons: Components may become negligible due to very low weights. Furthermore, due to the refinement, the number of mixture components grows over time. This growth will become a severe problem, when in addition the noise components $\underline{\boldsymbol{w}}_k$ and $\underline{\boldsymbol{v}}_k$ are itself represented as Gaussian mixtures[1]. As the noise terms form the transition density and likelihood, respectively, the multiplication of $f(\underline{x}_{k+1}|\underline{x}_k, \underline{u}_k)$ with $f_k^e(\underline{x}_k)$ in the prediction step (3.4) and the multiplication of $f(\hat{\underline{z}}_k|\underline{x})$ with $f_k^p(\underline{x}_k)$ in the measurement update (3.7) will lead to an exponential growth of the components. Here, the reduction is a must to maintain a feasible algorithm.

3.3 Component Adaptation

In this section, a detailed overview of the refinement and reapproximation operations appearing in Algorithm 3. At first, the two sub-operations weight optimization and reduction of the reapproximation step are discussed. The weight

1 Even if the noise is Gaussian, an approximation of the noise by means of a Gaussian mixture might be reasonable if the noise covariance is large. In case of a large covariance, the individual processing considered above becomes prone to large linearization errors. An approximation of the noise by a Gaussian mixture leads to components with lower covariances [7].

optimization part is kept short as it is not considered further in this thesis and thus, it is merely mentioned for the sake of completeness.

3.3.1 Weight Optimization

The posterior weight calculation in (3.7) facilitates the individual processing of the Gaussian components, but it is merely approximate as discussed above. To minimize the deviation between the mixture and the true posterior density, [90] proposed to adjust the weights by minimizing the norm

$$\int \left\| f_k^e\left(\underline{x}_k\right) - \sum_{i=1}^{L_k^e} \omega_{k,i}^e \cdot \mathcal{N}\left(\underline{x}_k; \underline{\mu}_{k,i}^e, \mathbf{C}_{k,i}^e\right) \right\| \mathrm{d}\underline{x}_k \,. \tag{3.8}$$

As the true posterior $f_k^e\left(\underline{x}_k\right)$ is not known and cannot be calculated analytically, this norm is evaluated at so-called *collocation points*. A natural choice of these collocation points are the mean vectors $\underline{\mu}_{k,i}^e$ of the posterior mixture. By this choice of collocation points and by using the L_2 norm in (3.8), the weight adjustment becomes a least squares optimization problem.

At a first glance, the weights obtained after the prediction seem to be correct. No approximation with respect to the weights is involved. However, no update of the weights is performed in the prediction; they coincide with the posterior weights. This procedure is optimal only for linear models, but becomes suboptimal when performing individual linearizations by means of a bank of Gaussian filters, especially when the covariances of the individual components is large and thus, the components may have a large overlap. For this situation, [193, 194] proposed a weight optimization procedure for the prediction step similar to the above posterior weight optimization. Instead of collocation points to evaluate (3.8), [193, 194] replace the true predicted density with the approximate posterior mixture density of the previous measurement update.

3.3.2 Reduction

To bound the growth of the number of Gaussian mixture components, *mixture reduction* aims at replacing a given Gaussian mixture $f\left(\underline{x}\right)$ with L components as in (3.1) by a mixture $\tilde{f}\left(\underline{x}\right)$ with M components, where M is significantly smaller than L, i.e., $M \ll L$. At the same time, the deviation between the true and the reduced mixture should kept at a minimum.

Deviation Measures

A very natural deviation measure between two densities from an information theoretic perspective is the *Kullback-Leibler divergence* (KLD)

$$G\left(f(\underline{x}) \| \tilde{f}(\underline{x})\right) \triangleq \int f(\underline{x}) \cdot \log \frac{f(\underline{x})}{\tilde{f}(\underline{x})} \,\mathrm{d}\underline{x} \,. \tag{3.9}$$

It can be interpreted as a quantification of the likelihood that data drawn from the true density is spuriously considered as be drawn from the reduced mixture and thus, it measures the difficulty of discriminating two densities. As discussed in [155, 210], thanks to this maximum likelihood interpretation and its scale-independence, the KLD should be the preferred choice for Gaussian mixture reduction. Unfortunately, the KLD is not a distance measure or norm in a strict sense as it is not symmetric. Furthermore, due to the logarithm in (3.9), the KLD cannot be evaluated in closed form for Gaussian mixtures.

To overcome the restrictions of the KLD, the *integrated squared difference* (ISD) is often employed as an alternative. It is defined as

$$D\left(f(\underline{x}), \tilde{f}(\underline{x})\right) \triangleq \int \left(f(\underline{x}) - \tilde{f}(\underline{x})\right)^2 \mathrm{d}\underline{x} \,. \tag{3.10}$$

Thus, the ISD is actually an L_2 norm. In contrast to the KLD, it has a closed-form expression for Gaussian mixtures, which is given by

$$\begin{aligned} D\left(f(\underline{x}), \tilde{f}(\underline{x})\right) = & \sum_{i=1}^{L} \sum_{j=1}^{L} \omega_i \cdot \omega_j \cdot \mathcal{N}\left(\underline{\mu}_i ; \underline{\mu}_j, \mathbf{C}_i + \mathbf{C}_j\right) + \\ & \sum_{i=1}^{L} \sum_{j=1}^{M} \omega_i \cdot \tilde{\omega}_j \cdot \mathcal{N}\left(\underline{\mu}_i ; \underline{\tilde{\mu}}_j, \mathbf{C}_i + \tilde{\mathbf{C}}_j\right) + \\ & \sum_{i=1}^{M} \sum_{j=1}^{M} \tilde{\omega}_i \cdot \tilde{\omega}_j \cdot \mathcal{N}\left(\underline{\tilde{\mu}}_i ; \underline{\tilde{\mu}}_j, \tilde{\mathbf{C}}_i + \tilde{\mathbf{C}}_j\right) , \end{aligned} \tag{3.11}$$

where the first term is called *self-similarity* of the true mixture, the second is the *cross-similarity* between the true and the reduced mixture, and the last term is the *self-similarity* of the reduced mixture [210].

State-of-the-Art

Depending on the optimization that is performed by a Gaussian mixture reduction algorithm, one can distinguish three basic classes: local, global, and pseudo-global algorithms. In the following, the key features and typical approaches of every class are discussed.

Local reduction algorithms typically perform an iterative *merging* of two or more components to a single Gaussian until a pre-defined threshold on the maximum allowed number of components is reached.

Example 11: Moment-preserving Merge

Let $\omega_i \cdot \mathcal{N}(\underline{x}; \underline{\mu}_i, \mathbf{C}_i)$ and $\omega_j \cdot \mathcal{N}(\underline{x}; \underline{\mu}_j, \mathbf{C}_j)$ be the two Gaussians that are considered for merging. By merging these two components, it should in addition be ensured that the zeroth-order (probability mass), first-order (mean) and second-order (covariance) moments of the entire mixture remain unchanged. Given these constraints, a *moment-preserving merge* of the two Gaussians yields the parameters

$$
\begin{aligned}
\tilde{\omega} &= \omega_i + \omega_j \; , \\
\underline{\tilde{\mu}} &= \tfrac{1}{\tilde{\omega}} \cdot \left(\omega_i \cdot \underline{\mu}_i + \omega_j \cdot \underline{\mu}_j \right) \; , \\
\tilde{\mathbf{C}} &= \tfrac{1}{\tilde{\omega}} \cdot \left(\omega_i \cdot \mathbf{C}_i + \omega_j \cdot \mathbf{C}_j + \tfrac{\omega_i \cdot \omega_j}{\tilde{\omega}} \cdot \left(\underline{\mu}_i - \underline{\mu}_j \right)\left(\underline{\mu}_i - \underline{\mu}_j \right)^{\mathrm{T}} \right) \; .
\end{aligned}
\tag{3.12}
$$

of the resulting single Gaussian $\tilde{\omega} \cdot \mathcal{N}(\underline{x}; \underline{\tilde{\mu}}, \tilde{\mathbf{C}})$.

The above moment-preserving merge can be easily extended for merging more than two Gaussians. The major difference of the most local algorithms is the criterion based on which components are selected for a merge. In [135, 157, 207] for instance, the Mahalanobis distance between Gaussians is utilized, while [188] employes a pair-wise version of the Hellinger metric and [39] a pair-wise version of the ISD. All these criteria only measure some similarity between components without consideration of the induced global deviation between the true and reduced mixture due to the merge, i.e., neither the KLD nor the ISD are monitored. Certainly, local algorithms are well-suited for real-time applications due to their low computational overhead.

In contrast, *global reduction* algorithms directly aim at minimizing the KLD or the ISD. In doing so, this class of algorithms tries to find the globally optimal reduced mixture. The reduction method proposed in [89] optimizes the parameters of the reduced mixture with respect to the ISD. This approach is constructive as it begins with a single Gaussian and adds components at locations where the deviation between the true and reduced mixture is too large. The algorithms in [31, 201] utilize the KLD, where the first is an expectation-maximization like approach and the second relies on variational Bayes. Merging components—when not performed iteratively as done by the local algorithms—can also yield globally optimal results as demonstrated in [46]. Here, all possible merges are first calculated and then the optimal solution is selected, which directly gives the reduced mixture. This procedure, however, is only feasible for L being a low number. In general, global algorithms suffer from a high computational load.

A compromise between both worlds is the third class of *pseudo-global* reduction algorithms. Here, iterative merging is performed as in local algorithms, but at the same time the KLD or ISD is monitored. [210] for instance considers that pair of Gaussians for a merge that introduces the smallest error into the reduced mixture in an ISD sense. [155] instead employes an upper-bound of the KLD and [167] uses only the cross-likeliness of the ISD. Because every possibly merge is inspected before actually performing the merge, pair-wise merging is globally optimal in a single step. Though, this does not guarantee global optimality after multiple reduction steps. Global optimality is only achieved by the aforementioned procedure proposed in [46]. To compensate this drawback, some algorithms like those in [69, 162] perform a dedicated optimization after merging, where the parameters of the reduced mixture are optimized with respect to the ISD. Alternatively to pair-wise merging, clustering-based or k-means based reductions perform iterated assignments of components of the true mixture to clusters. When no improvement in terms of the KLD or ISD is gained anymore, the components assigned to a cluster a merged which yields the reduced mixture.

Open Issues

Most of the aforementioned algorithms require a pre-defined threshold on the resulting number of mixture components, where an appropriate choice is difficult for the user. If the threshold is chosen too high, the entire Gaussian mixture filter becomes computationally demanding. A too low threshold may lead to poor estimates or even to a diverging filter. Ideally, this so-called *model selection* problem should be solved by the reduction algorithm itself, which so far is only achieved by [31, 89]. Both algorithms, however, are computationally and algorithmically very complex. Reductions of these algorithms and also of other global and pseudo-global approaches, are often more demanding than the actual predictions and measurement updates.

3.3.3 Refinement

While reduction aims at removing components from the mixture, the refinement step introduces new ones. The refinement becomes necessary when severe linearization errors of the individual Gaussian filters threaten the estimation performance.

Nonlinearity Measures

In order to avoid a blind adding of new components, the potential linearization error or the "strength" of the nonlinearity needs to be quantified. For some Gaussian filters, dedicated *nonlinearity measures* have been proposed in the past. For a brief overview, the nonlinear transformation in (2.6) is revisited, where now $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$ are Gaussian mixtures as in (3.1).

If for instance an EKF is employed as Gaussian filter, [3, 92] propose the measure

$$N\left(\underline{\mu}_i, \mathbf{C}_i\right) \triangleq \frac{\sqrt{\mathrm{Tr}\left(\mathbf{G}_{xx}\left(\underline{\mu}_i\right)\mathbf{C}_i\mathbf{G}_{xx}\left(\underline{\mu}_i\right)\mathbf{C}_i\right)}}{\sigma_w} ,$$

where $\mathbf{G}_{xx}\left(\underline{x}\right) \triangleq \partial^2 \underline{g}(\underline{x})/\partial\underline{x}\partial\underline{x}^\mathrm{T}$ is the Hessian matrix of $\underline{g}(.)$. Unfortunately, this measure is restricted to EKFs and to scalar $\boldsymbol{y}$ only. A measure based on the KLD (3.9) that is not limited by the dimension of $\boldsymbol{y}$ is proposed in [145]. Here, every mixture component of $\underline{\boldsymbol{y}}$ is compared against the true density. As the true density is not given in closed-form, numerical integration is necessary, which is demanding for high dimensions.

For the UKF, [62] proposed the measure

$$\sum_{i=1}^{n_y} \eta_i , \quad \text{with } \eta_i \triangleq \tfrac{1}{2}\left\|\mathcal{Y}_i + \mathcal{Y}_{n_x+i} - \mathcal{Y}_{2n_x+1}\right\|_2^2$$

that quantifies the goodness of fit of the propagated sample points $\mathcal{Y}_i = \underline{g}(\mathcal{X}_i)$, $i = 1 \ldots n_x$, to a linear regression model. The measure is close to zero if $\underline{g}(.)$ is approximately linear.

Adding New Components

By means of the nonlinearity measures it is possible to locate the Gaussian at the strongest nonlinearity or linearization error. A typical action to attenuate the linearization error is *splitting*, i.e., the identified Gaussian component $\omega \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}, \mathbf{C}\right)$ is replaced by a Gaussian mixture according to

$$\omega \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}, \mathbf{C}\right) \approx \sum_{i=1}^{L} \omega_i \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_i, \mathbf{C}_i\right) , \tag{3.13}$$

where the components on the right-hand side possess smaller covariances than the original component on the left-hand side, which is necessary for reducing the linearization error.

It is worth mentioning that (3.13) has two interpretations. Reading the equation from left to right corresponds to splitting, but reading the equation from right to left is nothing else then mixture reduction. Hence, splitting can be considered the dual operation to mixture reduction.

It can be easily verified that for $L > 1$, the number of free parameters on the right-hand side of (3.13), i.e., weights, mean vectors, and covariance matrices, is larger than the number of given parameters. Hence, splitting a Gaussian is an *ill-posed problem*.

The solution to this problem proposed in [6] is based on the UKF for determining sample points of the Gaussian that has to be split. The sample points with corresponding weights are used as the mean vectors $\underline{\mu}_i$ and mixture weights ω_i in (3.13), respectively. The covariances $\mathbf{C}_i$ are chosen to be identical. In [7, 62], the new Gaussians are placed as a regular grid with identical covariance matrices. The weights are chosen to be $\omega_i = \mathcal{N}\left(\underline{\mu}_i; \underline{\mu}, \mathbf{C}\right)$, i.e., the normalized probability value of the original Gaussian at the means $\underline{\mu}_i$ of the new Gaussian. A so-called splitting library is used in [75, 85], i.e., an off-line calculated approximation of a standard Gaussian by a mixture of Gaussians. This approximation is adjusted on-line by means of straightforward scaling operations.

Open Issues

The discussed nonlinearity measures are either restricted to very specific Gaussian filters like the UKF and EKF or they are difficult to evaluate for arbitrary models. Except of the splitting library all splitting approaches suffer from scalability problems as the number of components scales with the dimension of the state space. Furthermore, splitting is performed very “generous”, i.e., new components are introduced at every dimension of the state space, although the linearization error might affect only some state variables.

3.4 Contributions

The contributions made by the Papers E–G are discussed in the following sections. At first, the nonlinear-nonlinear state decomposition approach that was proposed in Section 2.5.2 is extended to Gaussian mixture filters. In Section 3.4.2, a new measure of the degree of nonlinearity and a new splitting approach are proposed. Both together form the so-called *adaptive Gaussian mixture filter*. Finally, a global Gaussian mixture reduction algorithm that exploits the ISD and the curvature of the true density is introduced.

3.4.1 Semi-Analytic Gaussian Mixture Filter

As the generic Gaussian mixture filter in Algorithm 3 uses a bank of Gaussian filters, the SAGF proposed in Section 2.5.2 can be directly applied in order to obtain a *semi-analytic Gaussian mixture filter* (SAGMF). As in Section 2.5.2, the conditionally integrable nonlinear transformation

$$\underline{\boldsymbol{y}} = \underline{g}(\underline{\boldsymbol{x}}_a, \underline{\boldsymbol{x}}_s) + \underline{\boldsymbol{w}}$$

is considered, where the state $\underline{\boldsymbol{x}}$ comprises the conditionally integrable state $\underline{\boldsymbol{x}}_a$ and the sampled state $\underline{\boldsymbol{x}}_s$. The density of the state is

$$f(\underline{x}) = \sum_{i=1}^{L} \omega_i \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_i^x, \mathbf{C}_i^x\right) \quad \text{with} \quad \underline{\mu}_i^x = \begin{bmatrix} \underline{\mu}_i^a \\ \underline{\mu}_i^s \end{bmatrix}, \mathbf{C}_i^x = \begin{bmatrix} \mathbf{C}_i^a & \mathbf{C}_i^{as} \\ \mathbf{C}_i^{sa} & \mathbf{C}_i^s \end{bmatrix}. \tag{3.14}$$

To obtain the Gaussian mixture density $\sum_{i=1}^{L} \omega_i \cdot \mathcal{N}\left(\underline{y}; \underline{\mu}_i^y, \mathbf{C}_i^y\right)$ of $\underline{\boldsymbol{y}}$, it is necessary to solve the moment integrals (2.8) for every component of (3.14). At first, for every component $i = 1 \ldots L$, a sample approximation of the marginal Gaussian $\mathcal{N}\left(\underline{x}_s; \underline{\mu}_i^s, \mathbf{C}_i^s\right)$ is calculated by means of an LRKF, which yields

$$\mathcal{N}\left(\underline{x}_s; \underline{\mu}_i^s, \mathbf{C}_i^s\right) \approx \sum_{j=1}^{N} \omega_{ij} \cdot \delta\left(\underline{x}^s - \mathcal{X}_{ij}\right) ,$$

where $\mathcal{L} = \left\{\omega_{ij}, \mathcal{X}_{ij}\right\}$ for $j = 1 \ldots N$ are the sample points of the ith Gaussian component.

Based on this sample representation, it is now possible to determine the mean and covariance of the ith component of $\underline{\boldsymbol{y}}$ according to

$$\underline{\mu}_i^y \approx \sum_{j=1}^{N} \omega_{ij} \cdot \underline{\mu}_{ij}^y , \tag{3.15}$$

$$\mathbf{C}_i^y \approx \sum_{j=1}^{N} \omega_{ij} \cdot \left(\mathbf{C}_{ij}^y - \underline{\mu}_{ij}^y \left(\underline{\mu}_i^y\right)^{\mathrm{T}} - \underline{\mu}_i^y \left(\underline{\mu}_{ij}^y\right)^{\mathrm{T}} + \underline{\mu}_i^y \left(\underline{\mu}_i^y\right)^{\mathrm{T}}\right) , \tag{3.16}$$

with

$$\begin{aligned} \underline{\mu}_{ij}^y &= \int \underline{g}\left(\underline{x}_a, \mathcal{X}_{ij}\right) \cdot f_i\left(\underline{x}_a \middle| \mathcal{X}_{ij}\right) \mathrm{d}\underline{x}_a , \\ \mathbf{C}_{ij}^y &= \int \underline{g}\left(\underline{x}_a, \mathcal{X}_{ij}\right) \cdot \underline{g}\left(\underline{x}_a, \mathcal{X}_{ij}\right)^{\mathrm{T}} \cdot f_i\left(\underline{x}_a \middle| \mathcal{X}_{ij}\right) \mathrm{d}\underline{x}_a , \end{aligned} \tag{3.17}$$

where $f_i\left(\underline{x}_a \middle| \mathcal{X}_{ij}\right) \triangleq \mathcal{N}\left(\underline{x}_a; \underline{\mu}_i^{a|s}, \mathbf{C}_i^{a|s}\right)$ is the conditional density of the ith component with mean and covariance as in (2.48). As $\underline{g}(.)$ is conditionally integrable, the integrals in (3.17) can be solved analytically.

To obtain the cross-covariance $\mathbf{C}_i^{xy} = \left[\mathbf{C}_i^{ay} \quad \mathbf{C}_i^{sy}\right]^{\mathrm{T}}$ it is more convenient to calculate its sub-matrices separately according to

$$\begin{aligned} \mathbf{C}_i^{ay} &= \sum_{j=1}^{N} \omega_{ij} \cdot \left(\mathbf{C}_{ij}^{ay} - \underline{\mu}_{ij}^{a|s} \left(\underline{\mu}_i^y\right)^{\mathrm{T}} + \underline{\mu}_i^a \left(\underline{\mu}_i^y - \underline{\mu}_{ij}^y\right)^{\mathrm{T}}\right) , \\ \mathbf{C}_i^{sy} &= \sum_{j=1}^{N} \omega_{ij} \cdot \left(\mathcal{X}_{ij} - \underline{\mu}_i^s\right) \cdot \left(\underline{\mu}_{ij}^y - \underline{\mu}_i^y\right)^{\mathrm{T}} , \end{aligned} \tag{3.18}$$

with

$$\mathbf{C}_{ij}^{ay} = \int \underline{x}_a \cdot \underline{g}\left(\underline{x}_a, \mathcal{X}_{ij}\right)^{\mathrm{T}} \cdot f_i\left(\underline{x}_a \middle| \mathcal{X}_{ij}\right) \mathrm{d}\underline{x}_a ,$$

where $\underline{\mu}_{ij}^{y}$ is given by (3.17) and $\underline{\mu}_{ij}^{a|s}$ is calculated according to (2.48) with $\underline{x}_s$ being replaced by $\mathcal{X}_{ij}$.

With the mean vectors (3.15), covariance matrices (3.16), and the sub-matrices (3.18) of the cross-covariance matrix all ingredients are given that are necessary for calculating the individual components of the predicted and posterior Gaussian mixture densities.

Example 12: Tricycle Kinematics

A robot with tricycle kinematics has to be localized. The nonlinear kinematics model is defined as

$$\begin{aligned}\boldsymbol{p}_{k+1}^{x} &= \boldsymbol{p}_k^{x} + \left(u_k^{v} + \underline{\boldsymbol{w}}_k^{v}\right)\cdot\cos\left(\boldsymbol{\phi}_k + u_k^{\alpha}\right),\\ \boldsymbol{p}_{k+1}^{y} &= \boldsymbol{p}_k^{y} + \left(u_k^{v} + \underline{\boldsymbol{w}}_k^{v}\right)\cdot\sin\left(\boldsymbol{\phi}_k + u_k^{\alpha}\right),\\ \boldsymbol{\phi}_{k+1} &= \boldsymbol{\phi}_k + \left(u_k^{\alpha} + \underline{\boldsymbol{w}}_k^{\alpha}\right),\end{aligned}$$

with state $\underline{\boldsymbol{x}}_k = \left[\boldsymbol{p}_k^{x}\ \boldsymbol{p}_k^{y}\ \boldsymbol{\phi}_k\right]^{\mathrm{T}}$, where $\boldsymbol{p}_k^{x}$ and $\boldsymbol{p}_k^{y}$ describe the Cartesian position of the robot and $\boldsymbol{\phi}_k$ its orientation. The initial position of the robot at time step $k = 0$ is $\underline{x}_0 = [5\ 3\ 0.2]^{\mathrm{T}}$. The known control inputs are the velocity $u_k^{v} = 0.1$ and the turning angle $u_k^{\alpha} = 0.1$. $\underline{\boldsymbol{w}}_k^{v}$ and $\underline{\boldsymbol{w}}_k^{\alpha}$ are noise processes affecting the corresponding control inputs. They are assumed to be zero-mean Gaussian with variances $\sigma_{wv}^2 = 0.1$ and $\sigma_{w\alpha}^2 = 0.01$, respectively.

The state is decomposed into $\underline{\boldsymbol{x}}^{a} = \left[\boldsymbol{p}^{x}\ \boldsymbol{p}^{y}\ \underline{\boldsymbol{w}}^{v}\ \underline{\boldsymbol{w}}^{\alpha}\right]^{\mathrm{T}}$ and $\boldsymbol{x}^{b} = \boldsymbol{\phi}$. Hence, by conditioning on $\boldsymbol{x}_s$, the system model becomes linear and can be solved via the prediction of the Kalman filter.

Range measurements to landmarks are performed for localizing the robot. The measured range $\boldsymbol{r}_k$ is defined by the nonlinear measurement model

$$\boldsymbol{r}_k = \sqrt{\left(\boldsymbol{p}_k^{x} - L^{x} + \boldsymbol{v}_k^{x}\right)^2 + \left(\boldsymbol{p}_k^{y} - L^{y} + \boldsymbol{v}_k^{y}\right)^2},$$

where $\underline{L} = \left[L^{x}\ L^{y}\right]^{\mathrm{T}}$ is the position of the landmark. Four landmarks at the positions

$$\left[\underline{L}_1\quad \underline{L}_3\quad \underline{L}_3\quad \underline{L}_4\right] = \begin{bmatrix}0 & 2 & 5 & 10\\ 0 & 2 & 5 & 10\end{bmatrix}.$$

are given. The measurement noise $\underline{\boldsymbol{v}}_k = \left[\boldsymbol{v}_k^{x}\ \boldsymbol{v}_k^{y}\right]^{\mathrm{T}}$ is zero-mean Gaussian with covariance $\mathbf{C}^{v} = \sigma_v^2\cdot\mathbf{I}_2$. For the variance σ_v^2, the three noise levels 0.5,

Table 3.1: Average rmse and standard deviation for the different filters at different noise levels and numbers of components.

Noise 0.5	64	8	1
FAGMF	**2.02 ± 1.34**	**3.17 ± 1.98**	**3.76 ± 2.01**
SAGMF	2.03 ± 1.34	**3.16 ± 1.99**	3.76 ± 2.09
UGMF	2.41 ± 1.79	4.08 ± 3.02	4.31 ± 3.58
EGMF	2.64 ± 1.86	4.08 ± 4.99	6.40 ± 9.64

Noise 1.0	64	8	1
FAGMF	**2.05 ± 1.36**	3.16 ± 1.97	**3.70 ± 2.07**
SAGMF	**2.06 ± 1.35**	**3.15 ± 1.97**	**3.70 ± 2.07**
UGMF	2.44 ± 1.80	4.04 ± 3.05	4.25 ± 3.61
EGMF	2.95 ± 1.73	4.40 ± 4.43	6.72 ± 8.71

Noise 2.0	64	8	1
FAGMF	**2.16 ± 1.35**	**3.22 ± 2.02**	**3.78 ± 2.16**
SAGMF	2.16 ± 1.36	**3.22 ± 2.02**	**3.78 ± 2.16**
UGMF	2.61 ± 1.77	4.11 ± 3.08	4.35 ± 3.65
EGMF	3.42 ± 1.55	4.95 ± 4.14	7.21 ± 7.60

1, and 2 are considered. The measurement step is performed analytically by means of the closed-form solution derived in Section 5.1.

The proposed semi-analytic Gaussian mixture filter (SAGMF) is compared against Gaussian mixture filters employing EKFs and UKFs. These Gaussian mixture filters are denoted EGMF and UGMF in the following. Furthermore, a fully analytical Gaussian mixture filter (FAGMF) is employed as a baseline that performs component-wise moment matching for the prediction. All GMFs are initialized with different numbers of components, namely 1, 8, and 64 components. For each combination of noise level and number of components, 1000 simulation runs are performed, where each run consists of 50 time steps. The results are listed in Table 3.1.

The SAGMF clearly outperforms both EGMF and UGMF independent of the noise level or the used number of components. Furthermore, the estimation errors are almost identical with the baseline provided by the FAGMF. In terms of run-time, EGMF and AGMF are the fastest filters, but SAGMF is significantly faster than the UGMF, which suffers from the necessity of calculating high-dimensional matrix square roots. Furthermore, it requires a high number of on-line evaluations of the kinematics and measurement model, while in case of the SAGMF the number of functions evaluations is reduced significantly due to the small dimension of the sub-state $\boldsymbol{x}_s$.

It is important to point out that the outstanding performance of FAGMF is limited to a small number of applications. The SAGMF instead is of greater benefit as it is a general purpose filter, where both UGMF and FAGMF are its limiting cases.

3.4.2 Adaptive Gaussian Mixture Filter

The estimation error of Gaussian mixture filters significantly depends on the number of Gaussian components used. This number is typically defined by the user. The novel *adaptive Gaussian mixture filter* (AGMF) depicted in Figure 3.3 on the next page adapts the number of components dynamically and on-line and thus, directly tackles the refinement step of Algorithm 3. The nonlinear system and measurement models are linearized locally by means of statistical linear regression (see Section 2) at each component of the Gaussian mixture. The induced linearization error is quantified by means of the linearization error covariance matrix (2.28). Based on this error, a novel moment-preserving splitting procedure is proposed for introducing new mixture components. The component causing the highest linearization error is selected, while splitting is performed in direction of the strongest nonlinearity, i.e., the strongest deviation between the nonlinear model and its linearized version. Both linearization and splitting are independent of the used statistical linear regression method, which makes the proposed filter versatilely applicable.

Component Selection

A straightforward way to select a Gaussian component for splitting is to consider the weights ω_i, $i = 1 \ldots L$. The component with the highest weight is then split. This however does not take the nonlinearity of $\underline{g}(\cdot)$ in the support of the selected component into account. Since linearization is performed individually and locally, a more reasonable selection would be to consider also the induced

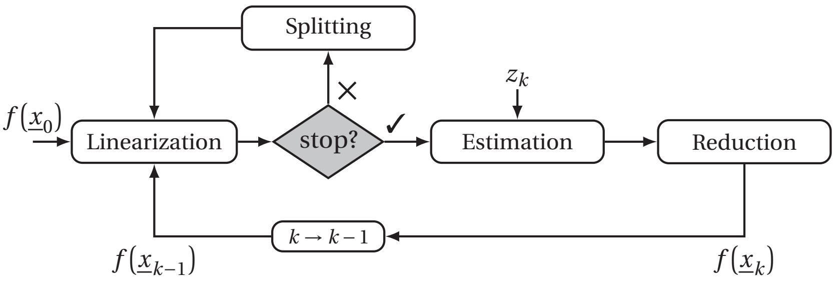


Figure 3.3: Flow chart of the adaptive Gaussian mixture filter.

linearization error of each component. For this purpose, statistical linear regression already provides an appropriate measure of the linearization error in form of the covariance matrix $\mathbf{C}_e$ in (2.28).

In order to easily assess the linearization error in the multi-dimensional case, the trace operator is applied to $\mathbf{C}_e$, which gives the measure

$$\varepsilon = \mathrm{Tr}(\mathbf{C}_e) \in [0, \infty) \ . \tag{3.19}$$

Geometrically speaking, the trace is proportional to circumference of the covariance ellipsoid corresponding to $\mathbf{C}_e$. The larger $\mathbf{C}_e$ and thus the linearization error, the larger is ε. Conversely, the trace is zero, if and only if $\mathbf{C}_e$ is the zero matrix, i.e., $\varepsilon = 0 \Leftrightarrow \mathbf{C}_e = \mathbf{0}$. Hence, (3.19) is only zero, when there is no linearization error.

Example 13: Quantified Linearization Error

The error measure in (3.19) is applied to both scalar functions

$$g(\boldsymbol{x}) = \cos(\boldsymbol{x}) \quad \text{and} \quad g(\boldsymbol{x}) = \exp\left(-\boldsymbol{x}^2\right) \ ,$$

where $\boldsymbol{x} \sim \mathcal{N}(\mu, 1)$ has constant variance and the mean is moved along the x-axis. The corresponding linearization error values for both functions are depicted in Figure 3.4 on the next page. The measure approaches zero whenever the function $g(.)$ is almost linear. This is the case for $\cos(x)$ if $x = 0$ and for $\exp\left(-x^2\right)$ if $x = \pm\sqrt{2}/2$ and $x \to \pm\infty$.

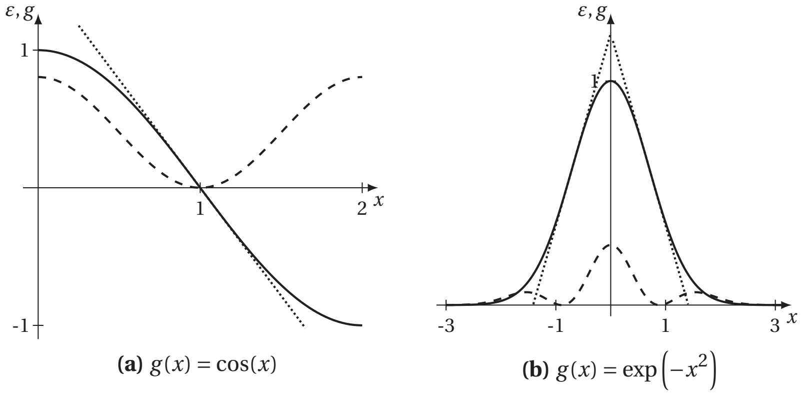


Figure 3.4: Two examples of the linearization error quantification by means of the measures (3.19). The solid lines correspond to the function $g(.)$, the dashed lines to the measure (3.19), and the dotted lines indicate linearized versions of $g(.)$

Besides the linearization error, the contribution of a component to the nonlinear transformation is important as well. That is, the probability mass of the component, which is given by its weight ω_i, has also to be taken into account. This avoids splitting of irrelevant components. Both ingredients are combined in the so-called *selection criterion*

$$i^* = \arg \max_{i=1\ldots L} \{s_i\}\ , \quad s_i \triangleq \omega_i^\gamma \cdot \left(1 - \exp(-\varepsilon_i)\right)^{1-\gamma} \tag{3.20}$$

Here, the term $1 - \exp(-\varepsilon_i)$ normalizes the linearization measure (3.19) to the interval $[0,1]$. For a geometric interpolation between weight and linearization error of component i, the parameter $\gamma \in [0,1]$ used. With $\gamma = 0$, selecting a component for splitting only focuses on the linearization error, while $\gamma = 1$ considers the weight only.

Splitting

Once a component causing a large linearization error is identified, new components are introduced by means of splitting the identified component. As mentioned above, splitting is an ill-posed problem due to the high number of free parameters. To reduce the degrees of freedom, splitting is performed given the following constraints:

1. *Along principal axes of a Gaussian:* This reduces the splitting problem of a multivariate Gaussian to the one-dimensional case. Furthermore, splitting becomes computationally cheap and numerically stable compared to arbitrary splitting directions.
2. *Splitting into two components:* Allows trading the reduction of the linearization error off against the increase of mixture components.
3. *Symmetric around the mean:* This minimizes the error introduced by splitting a Gaussian.
4. *Moment preserving:* The mean vector and covariance matrix of the split Gaussian and thus, of the entire mixture remains unchanged.

The restriction to split a Gaussian into two components is no limitation. As splitting is performed recursively by the AGMF within a time step, the newly introduced components can be split again in the next rounds if the linearization error is still too high.

Let $\omega \cdot \mathcal{N}(\underline{x}; \underline{\mu}, \mathbf{C})$ be the component considered for splitting. It is replaced by two components according to (3.13) with parameters

$$\begin{aligned} &\omega_1 = \omega_2 = \tfrac{\omega}{2}\ , \\ &\underline{\mu}_1 = \underline{\mu} + \sqrt{\lambda} \cdot \alpha \cdot \underline{v}\ , \quad \underline{\mu}_2 = \underline{\mu} - \sqrt{\lambda} \cdot \alpha \cdot \underline{v}\ , \\ &\mathbf{C}_1 = \mathbf{C}_2 = \mathbf{C} - \lambda \cdot \alpha \cdot \underline{v}\,\underline{v}^{\mathrm{T}}\ , \end{aligned} \tag{3.21}$$

where $\alpha \in [-1, 1]$ is a free parameter. The parametrization in (3.21) ensures moment preservation, i.e., the original Gaussian component and its split counterpart have the same mean and covariance. Furthermore, λ and $\underline{v}$ in (3.21) are a particular eigenvalue and eigenvector, respectively, of $\mathbf{C}$.

Splitting Direction

What remains an open question is the selection of an appropriate eigenvector for splitting. A straightforward choice might be the eigenvector with the largest eigenvalue as in [63, 85]. But since (3.20) determines the Gaussian component that causes the largest linearization error, merely splitting along the eigenvector with the largest eigenvalue does not take this error into account.

The key idea of the proposed criterion is to evaluate the deviation between the nonlinear transformation $\underline{g}(.)$ and its linearized version (2.26) along each eigenvector. The eigenvector with the largest deviation is then considered for splitting, i.e., the Gaussian is split in direction of the largest deviation in order to cover this direction with more Gaussians, which will reduce the error in subsequent linearization steps.

By means of the error term (2.27), the desired criterion for the splitting direction is defined as

$$d_l \triangleq \int_{\mathbb{R}} \underline{\boldsymbol{e}}\big(\underline{x}_l(\nu)\big)^{\mathrm{T}} \cdot \underline{\boldsymbol{e}}\big(\underline{x}_l(\nu)\big) \cdot \mathcal{N}\big(\underline{x}_l(\nu); \underline{\mu}, \mathbf{C}\big)\, \mathrm{d}\nu \qquad (3.22)$$

with $\underline{x}_l(\nu) \triangleq \underline{\mu} + \nu \cdot \underline{v}_l$, $l = 1 \ldots n_x$, and $\underline{v}_l$ being the lth eigenvector $\mathbf{C}$. The integral in (3.22) cumulates the squared deviations along the lth eigenvector under the consideration of the probability at each point $\underline{x}_l(\nu)$. The eigenvector that maximizes (3.22) is then chosen for splitting. Unfortunately, due to the nonlinear transformation $\underline{g}(\cdot)$ this integral cannot be solved in closed-form in general. For an efficient and approximate solution, the sample point calculation schemes described in Section 2.2.5 are employed to approximate the Gaussian in (3.22) in direction of $\underline{v}_l$ by means of Dirac delta distributions. This automatically leads to a discretization of the integral at a few but carefully chosen points.

Splitting Termination

As indicated in Figure 3.3, in every splitting round a *stopping criterion* is evaluated. Splitting stops, if at least one of the three following thresholds is reached:

Error threshold: The value s_i in the selection criterion (3.20) drops below $s_{\max} \in [0,1]$ for *every* component.

Component threshold: The number of mixture components excels $L_{\max}$.

Deviation threshold: The deviation between the original mixture $f\big(\underline{x}\big)$ and the mixture obtained via splitting $\tilde{f}\big(\underline{x}\big)$ excels $d_{\max} \in [0,1]$.

The deviation considered for the latter threshold is determined by means of the normalized version of the ISD according to

$$\bar{D}\big(f(\underline{x}), \tilde{f}(\underline{x})\big) \triangleq \frac{D\big(f(\underline{x}), \tilde{f}(\underline{x})\big)}{\int f(\underline{x})^2\, \mathrm{d}\underline{x} + \int \tilde{f}(\underline{x})^2\, \mathrm{d}\underline{x}} \in [0,1]\,. \qquad (3.23)$$

Since splitting always introduces an approximation error to the original mixture, continuously monitoring the deviation limits this error.

Example 14: Shape Approximation

In order to demonstrate the effectiveness of splitting in direction of the strongest nonlinearity, the nonlinear growth process

$$\boldsymbol{y} = g(\underline{\boldsymbol{x}}) = \frac{\boldsymbol{\xi}}{2} + 5 \cdot \frac{\boldsymbol{\xi}}{1+\boldsymbol{\xi}^2} + \boldsymbol{w}$$

adapted from [101] is considered, where $\underline{\boldsymbol{x}} = [\boldsymbol{\xi}\ \boldsymbol{w}]^{\mathrm{T}} \sim \mathcal{N}\left([1,0]^{\mathrm{T}}, \mathbf{I}_2\right)$. To approximate the density of $\boldsymbol{y}$, the Gaussian $f(\underline{x})$ is split recursively into a Gaussian mixture, where the number of components is always doubled until a maximum of 64 components is reached. No mixture reduction and no thresholds $s_{\max}$, $d_{\max}$ are used. The true density of $\boldsymbol{y}$ is calculated via numerical integration.

Two different values for the parameter γ of the selection criterion (3.20) are used: $\gamma = 0.5$, which makes no preference between the component weight and the linearization error and $\gamma = 1$, which considers the weight only. Furthermore, a rather simple selection criterion is considered for comparison, where selecting a Gaussian for splitting is based on the weights only (as it is the case for $\gamma = 1$), while the splitting is performed in direction of the eigenvector with the largest eigenvalue.

Table 3.2 shows the KLD between the true density of $\boldsymbol{y}$ and the approximations obtained by splitting. The approximations of the proposed splitting scheme are significantly better than the approximations of the largest eigenvalue scheme. This follows from the fact that the proposed scheme not only considers the spread of a component. It also takes the linearization errors

Table 3.2: Approximation error (KLD × 10) for different splitting schemes and numbers of components.

splitting	number of Gaussians						
scheme	1	2	4	8	16	32	64
max. eigenvalue	2.01	0.77	0.64	0.47	0.39	0.21	0.26
$\gamma = 1$	2.01	0.77	0.59	0.34	0.20	0.12	0.07
$\gamma = 0.5$	2.01	0.77	0.40	0.22	0.07	0.03	0.02

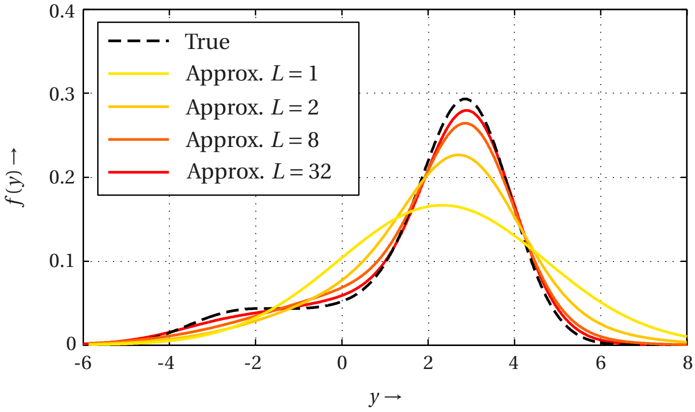


Figure 3.5: True density function of $\boldsymbol{y}$ (black, dashed) and approximations with an increasing number of mixture components for $\gamma = 0.5$.

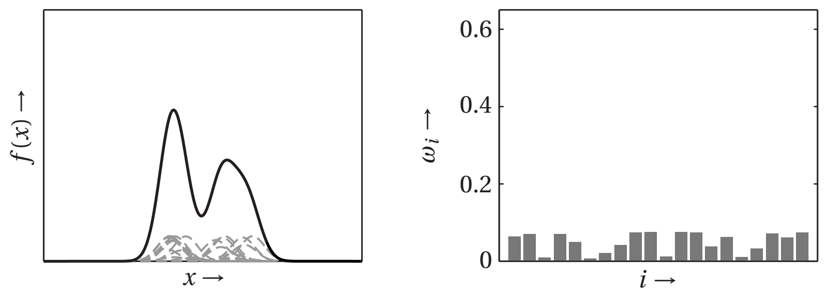
into account. In doing so, the Gaussians are always split along the eigenvector corresponding to $\boldsymbol{\xi}$, since this variable is transformed nonlinearly, while $\boldsymbol{w}$ is not. This is different for the largest eigenvalue scheme, which wastes nearly half of the splits on $\boldsymbol{w}$.

The inferior approximation quality for $\gamma = 1$ compared to $\gamma = 0.5$ results from splitting components, which may have a high importance due to their weight but which do not cause severe linearization errors. Thus, splitting these components will not improve the approximation quality much.

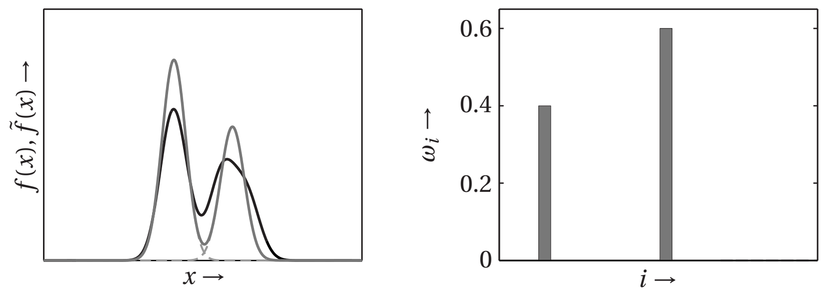
In Figure 3.5, the approximate density of $\boldsymbol{y}$ is depicted for different numbers of mixture components for $\gamma = 0.5$. With an increasing number of components, the approximation approaches the true density very well.

3.4.3 Curvature-based Gaussian Mixture Reduction

Basically, the AGMF proposed above can operate with any Gaussian mixture reduction algorithm, but in order to maintain a high estimation performance, the employed reduction algorithm should provide a very good approximation of the original mixture. At the same time the reduction should be large enough in order



(a) Original Gaussian mixture consisting of 20 components.



(b) Reduced Gaussian mixture consisting of two components.

Figure 3.6: A Gaussian mixture and its reduced version. Left: original Gaussian mixture (black), reduced mixture (dark gray), and the mixture components (light gray, dashed). Right: the mixture weights.

to allow computationally feasible estimation. As mentioned in Section 3.3.2, global and pseudo-global reduction algorithms can provide an accurate approximation, but they suffer from a high complexity. This drawback becomes even more severe, when the number of mixture components is initially very large. This situation, however, appears quite often in filtering applications, especially when Gaussian mixtures are multiplied or when splitting adds many components due to a high state dimension.

Especially in cases, where the number of components is large, it is possible to find a reduced mixture merely by setting the weights ω_i of many components to zero and still, the reduced mixture approximates the original mixture sufficiently well.

Example 15: Weight Optimization

Consider a Gaussian mixture consisting of 20 components as depicted in Figure 3.6a on the previous page. Intuitively one would guess that two to three Gaussians should be sufficient to provide an approximation of the original mixture that captures both modes. The remaining components seem to be redundant. In Figure 3.6b the extreme case is depicted, where the weights of all components except of two are set to zero. The weights of the remaining two components are merely normalized such that their sum is equal to one. The resulting reduced mixture captures both modes. However, there is still room for improvement, e.g., by optimizing the weight or by allowing one more component.

The *superficial Gaussian mixture reduction* (SGMR) introduced next is a global reduction algorithm that is based on minimizing the curvature—maximizing the smoothness—of the reduced mixture while keeping the ISD low. By means of using the curvature it is possible to identify similar components globally and to remove these components from the density. Carrying the idea that similar components may be dropped a step further, the trade-off between curvature and approximation error is minimized by merely optimizing the weights, i.e., assigning zero weights to reduced components. This approach is computationally feasible as no other parameters of the mixture need to be optimized. It further allows a simple and efficient implementation based on standard quadratic program (QP) solvers. Additionally, this weight-only optimization alleviates the model selection problem as the final number of components is automatically derived from setting the trade-off between error and roughness. This hyperparameter may be automatically optimized as well. In cases where the resulting number of components still is too large from a computational perspective, existing global reduction algorithm can be employed ex post. Thanks to the already reduced mixture, the computational load of the global algorithm can be lowered significantly.

Quadratic Program

By focusing on the weights only, the mixture reduction problem can be formulated as a quadratic program

$$\min_{\underline{\tilde{\omega}}} \quad \underline{\tilde{\omega}}^{\mathrm{T}} \mathbf{Q} \, \underline{\tilde{\omega}} - \underline{q}^{\mathrm{T}} \underline{\tilde{\omega}} \tag{3.24}$$

$$\begin{aligned} \text{s.t.} \quad & \underline{1}^{\mathrm{T}} \cdot \underline{\tilde{\omega}} = 1 \ , \\ & \underline{0} \preceq \underline{\tilde{\omega}} \ , \\ & \underline{0} = \textstyle\sum_{i=1}^{L} \underline{\mu}_i \cdot (\omega_i - \tilde{\omega}_i) \ , \end{aligned}$$

where $\underline{\tilde{\omega}}^{\mathrm{T}} \triangleq [\tilde{\omega}_1 \ \ldots \ \tilde{\omega}_L]$ is the vector of all weights of $\tilde{f}(\underline{x})$ to be optimized. The symmetric matrix $\mathbf{Q} \triangleq \mathbf{D} + \lambda \mathbf{R}$ comprises the positive semi-definite matrices $\mathbf{D}$ and $\mathbf{R}$, where $\mathbf{D}$ originates from the ISD and $\mathbf{R}$ is a *roughness penalty* measuring the curvature of $\tilde{f}$. The vector $\underline{q} \triangleq 2\,\underline{d}$, where $\underline{d}^{\mathrm{T}} \triangleq \underline{\tilde{\omega}}^{\mathrm{T}}\mathbf{D}$ depends on the ISD as well. The hyperparameter λ governs the trade-off between $\mathbf{D}$ and $\mathbf{R}$. It needs to be determined by means of generic model selection algorithms, see e.g. [163]. For small values of λ, the ISD will be weighted relatively higher than the curvature. This results in more components in the reduced mixture $\tilde{f}$ and less approximation error. For large values of λ, the curvature will be weighted higher enforcing more reduction and approximation error.

The constraints in the QP assert the integration of the probability mass to one, the positivity of the density, and that $\tilde{f}$ and f have identical means.

It is important to note the number of components in f and $\tilde{f}$ is identical. After solving (3.24), many weights will be zero or close to zero. Thus, the corresponding components no longer contribute to the mixture and can be discarded.

The components of the ISD in (3.24) are obtained, when examining (3.11) for all parameters but the weights of $\tilde{f}$ as fixed. It turns out that (3.11) merely consists of linear and quadratic terms of the weights $\tilde{\omega}_i$. Thus, the ISD can be written as

$$D\big(f(\underline{x}), \tilde{f}(\underline{x})\big) = \underline{\tilde{\omega}}^{\mathrm{T}}\mathbf{D}\,\underline{\tilde{\omega}} - 2\underline{d}^{\mathrm{T}}\underline{\tilde{\omega}} + c \ , \tag{3.25}$$

where the matrix $\mathbf{D}$ corresponds to the self-similarity of $\tilde{f}$, the vector $\underline{d}$ encodes the cross-similarity between f and $\tilde{f}$, and the constant $c \triangleq \underline{\omega}^{\mathrm{T}}\mathbf{D}\,\underline{\omega}$ corresponds to the self-similarity of f.

The roughness of $\tilde{f}$ is interpreted as the curvature κ of the density's surface. Since the curvature (see e.g. [35]) is signed and a function of the position on the surface, a quantification in terms of the integral squared curvature is sought. The key idea is to derive an (approximate) upper bound of the squared curvature of the mixture. The derivation is based on the point-wise squared curvature $\kappa(\underline{x})$ for a density $\tilde{f}$, for which an upper bound $\breve{\kappa}(\underline{x})$ is determined, integrated over the entire domain of $\underline{x}$, i.e., $R\big(\tilde{f}(\underline{x})\big) = \int \breve{\kappa}(\underline{x})^2 \,\mathrm{d}\underline{x}$. For the 1D and 2D case, the upper bounds are

$$\breve{\kappa}(x)^2 \triangleq \left(\underline{\tilde{\omega}}^{\mathrm{T}} \underline{\tilde{f}}_{xx}\right)^2 , \quad \breve{\kappa}\left(\underline{x}\right)^2 \triangleq \left(\underline{\tilde{\omega}}^{\mathrm{T}} \underline{\tilde{f}}_{xx} - 2\,\underline{\tilde{\omega}}^{\mathrm{T}} \underline{\tilde{f}}_{x}\,\underline{\tilde{\omega}}^{\mathrm{T}} \underline{\tilde{f}}_{y}\,\underline{\tilde{\omega}}^{\mathrm{T}} \underline{\tilde{f}}_{xy} + \underline{\tilde{\omega}}^{\mathrm{T}} \underline{\tilde{f}}_{yy}\right)^2 ,$$

where $\underline{\tilde{f}}_x$ and $\underline{\tilde{f}}_{xy}$ being the first-order and second-order derivatives of $\underline{\tilde{f}}$, with $\underline{\tilde{f}}$ defined as in (3.2). These upper bounds of the integral squared mean curvature are obtained by dropping the denominator and positive summands.

For the weight optimization, the upper bound of the curvature is formulated as a quadratic form $\underline{\tilde{\omega}}^{\mathrm{T}}\,\mathbf{R}\,\underline{\tilde{\omega}}$. The elements of $\mathbf{R}$ may be obtained as follows. For the 1D case the elements of $\mathbf{R}$ are given by $\mathbf{R}_{ij} = \int_{\mathbb{R}} \tilde{f}_{xx}^{(i)}\, f_{xx}^{(j)} \,\mathrm{d}x$, where $\tilde{f}^{(i)}$ refers to the ith mixture component of $\tilde{f}$. For the 2D case, the following approximation is used

$$\begin{aligned}&\left(\underline{\tilde{\omega}}^{\mathrm{T}} \underline{\tilde{f}}_{xx} - 2\,\underline{\tilde{\omega}}^{\mathrm{T}} \underline{\tilde{f}}_{x}\,\underline{\tilde{\omega}}^{\mathrm{T}} \underline{\tilde{f}}_{y}\,\underline{\tilde{\omega}}^{\mathrm{T}} \underline{\tilde{f}}_{xy} + \underline{\tilde{\omega}}^{\mathrm{T}} \underline{\tilde{f}}_{yy}\right)^2 \\ &\qquad\qquad \approx \left(\underline{\tilde{\omega}}^{\mathrm{T}} \left[\underline{\tilde{f}}_{xx} - 2\,\underline{\tilde{f}}_{x}\,\underline{\tilde{f}}_{y}^{\mathrm{T}}\,\underline{\tilde{f}}_{xy} + \underline{\tilde{f}}_{yy}\right]\right)^2 ,\end{aligned}$$

which leads to the elements

$$\mathbf{R}_{ij} = \int_{\mathbb{R}^2} \left(\tilde{f}_{xx}^{(i)} - 2\,\tilde{f}_{x}^{(i)}\,\tilde{f}_{y}^{(i)}\,\tilde{f}_{xy}^{(i)} + \tilde{f}_{yy}^{(i)}\right) \cdot \left(\tilde{f}_{xx}^{(j)} - 2\,\tilde{f}_{x}^{(j)}\,\tilde{f}_{y}^{(j)}\,\tilde{f}_{xy}^{(j)} + \tilde{f}_{yy}^{(j)}\right) \mathrm{d}\underline{x} \,. \quad (3.26)$$

For the 1D curvature, the $\mathbf{R}_{ij}$ may be calculated in closed form. Note for a 2D density the curvature is not unique, as it is calculated from the minimum and maximum curvature in the principal directions at each point $\underline{x}$, which may be multiplied (Gaussian curvature) or averaged (mean curvature) [35]. For arbitrary Gaussian mixture densities, the terms in (3.26) may only be calculated numerically or need to be approximated further.

Algorithm

The entire SGMR algorithm comprises three parts: the *pre-processing* of the components of the quadratic forms and the hyperparameter, the weight optimization by the solution of (3.24), and a fast *post-optimization* of the already reduced set of weights from (3.24).

The *pre-processing* consists of calculating the matrix $\mathbf{D}$ and vector $\underline{d}$ corresponding to the ISD as well as the matrix $\mathbf{R}$ describing the curvature of the mixture's surface.

As the matrix $\mathbf{Q}$ is positive semi-definite, the QP is convex and can be solved efficiently and globally optimal by any standard solver (see Appendix C). The resulting weights are compared against a threshold $\varepsilon \ll 1$. This leads to reduced weights $\tilde{\omega}_i^+ \geq \varepsilon$, $i = 1 \ldots M \ll L$.

The purpose of the *post-optimization* is an adaptation of the already reduced weights $\underline{\tilde{\omega}}^+$, aimed at improving the accuracy, by neglecting the curvature and only minimizing the ISD with respect to $\underline{\tilde{\omega}}^+$. The corresponding QP is similar to (3.24) but without the curvature matrix $\mathbf{R}$.

Example 16: 1D Reduction

The proposed SGMR is compared against the following reduction algorithms: *Pruning* [20] of all but the components with the highest weights, *West*'s merging [207], *Salmond*'s clustering algorithm [157], *Williams*' merging algorithm [210], *Runnalls*' algorithm [155], and *PGMR* [89]. For SGMR, two variants are considered, one with post-optimization and one without post-optimization.

For evaluation, univariate Gaussian mixtures with a number of components $L \in \{40, 80, 120, 160, 200\}$ are used. The mixture parameters are drawn uniformly at random from the intervals $\tilde{\alpha} \in [0.05, 0.5]$, $\tilde{\mu} \in [0, 3]$, and $\tilde{\sigma} \in [0.09, 0.5]$. For each number of components L, 50 Monte Carlo simulation runs are performed. For SGMR, the hyperparameter λ is set to 500 and the deletion threshold ε is $1e^{-4}$. The maximum error threshold of PGMR is set to 1%.

Pruning, West, Salmond, Williams, and Runnalls require a user-defined threshold on the number of components to which the given Gaussian mixture has to be reduced. Since SGMR reduces a Gaussian mixture in a completely different fashion and thus, to ensure a fair comparison, the number of components resulting from SGMR with post-optimization is used as threshold for these approaches. In order to quantify the reduction error, the normalized ISD (3.23) is used.

In Figure 3.7, the average reduction errors and the average computation times for all L are shown. It can be seen that SGMR provides the lowest reduction error. Closest to SGMR is Williams' algorithm, but this algorithm

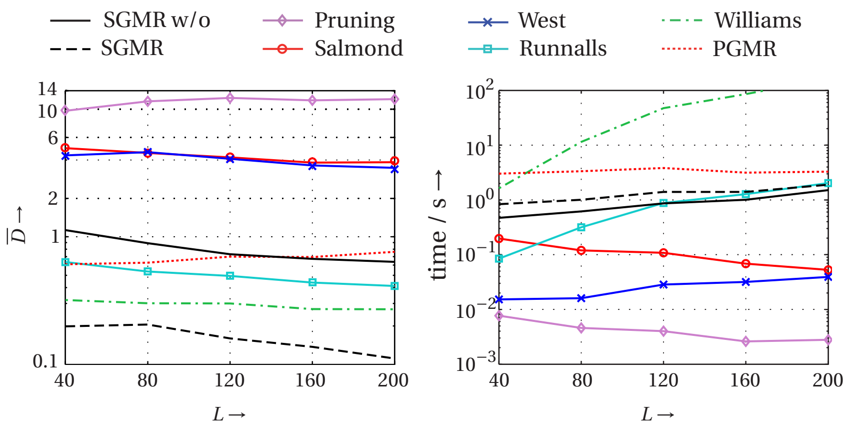


Figure 3.7: Reduction error (left) and runtime (right) of different reduction algorithms for increasing number of components. The results are averages over 50 Monte Carlo runs. The average reduction error is multiplied by 100 for better readability.

clearly suffers from its high computational demand. Salmond's and West's methods perform similarly. Both are very fast, but their approximation quality is the worst except of pruning. In terms of the reduction error, the results of Runnalls' method are in between of SGMR with and without post-optimization. But for an increasing number of components L in the original mixture it becomes computationally more expensive than both SGMR methods. Overall, SGMR provides the best trade-off between reduction error and computation time.

The reduction performance of SGMR improves with a larger number of components in the original mixture. As listed in Table 3.3, SGMR reduces to about 50% if the number of components of the original mixture is $L = 40$, while for $L = 200$ only 22% of the components remain. Since SGMR merely adapts the weights $\underline{\tilde{\omega}}$, a larger number of components is advantageous for SGMR for a better exploitation of redundancies. This leads to a stronger reduction by a simultaneously lower reduction error. Furthermore, the comparison between SGMR and SGMR without post-optimization shows that the post-optimization always lowers both the reduction error and the number of components.

Table 3.3: Number of components in the reduced Gaussian mixture for different reduction algorithms. The results are averages over 50 MC runs.

L	SGMR w/o	PGMR	SGMR all others
40	21.66	7.34	20.54
80	30.88	7.42	29.5
120	37.18	7.58	35.86
160	42.86	6.78	41.8
200	45.68	6.7	44.98

For 1D mixtures, PGMR clearly is the best reduction algorithm. The reduction error is close to SGMR without post-optimization, but the number of components in the reduced mixture is significantly lower (see Table 3.3). However, a straightforward extension to multivariate mixtures is not possible as only axis-aligned Gaussian components can be utilized for representing the reduced mixture.

3.5 Summary

Three algorithms supporting the generic Gaussian mixture filtering as sketched in Algorithm 3 have been proposed in this chapter, namely semi-analytic Gaussian mixture filter (SAGMF), adaptive Gaussian mixture filter (AGMF), and superficial Gaussian mixture reduction (SGMR). These algorithms aggregate following contributions:

- *Component-wise Rao-Blackwellization for nonlinear-nonlinear decomposition:* The decomposition of conditionally integrable functions already proposed for Gaussian filters can be applied straightforwardly in a mixture setup.
- *General linearization error measure for LRKFs:* The proposed linearization error measure exploits the error covariance that is provided as a side product of statistical linear regression. Thus, the computational overhead for quantifying the linearization error is low and the measure can be utilized by any LRKF.

- *Splitting along the strongest nonlinearity:* New components are introduced at the strongest nonlinearity in order to reduce linearization errors most effectively.
- *Scalable and moment-preserving splitting:* Splitting is performed carefully as components are only added where they are really required. This facilitates scaling with high state dimensions. In addition the mean and covariance of the Gaussian mixture are preserved.
- *Mixture reduction via weight-optimization:* To lower the computational demand of global mixture reduction but at the same to benefit from a low reduction errors, components are removed by considering their weights only. For this purpose, a novel QP compromising reduction error and roughness has been proposed.

As with the contributions made for Gaussian filtering, the above contributions and algorithms can and should be used in combination. The nonlinear sub-state that cannot be treated analytically by the SAGMF, can be processed by means of the AGMF, where the linearization error can be kept at a minimum. The newly introduced components due to splitting have to be reduced from time to time in order to keep the computational demand bounded. For this purpose, the SGMR can be employed.

4 Gaussian Process Filtering

So far it was assumed that the nonlinear relation between the current state $\underline{\boldsymbol{x}}_k$ and the next state $\underline{\boldsymbol{x}}_{k+1}$ or the measurement $\underline{\boldsymbol{z}}_k$ are known. In many applications, however, this assumption is no longer valid. Instead of a functional representation of these relations, only labeled data sets are given. Hence, at first the functional representation has to be learned from the data before the actual filtering can be performed. In this chapter, *Gaussian process* (GP) regression for learning the nonlinear model from data is introduced[1]. A GP defines a Gaussian prior density over functions, which can be turned into a posterior density over functions using Bayesian inference when data is present. Evaluating the resulting posterior model at a finite number of inputs yields a Gaussian distribution of functions values.

A GP model is *non-parametric*[2], which has the benefit that one has not to worry about selecting the wrong model to fit the data. Key element of a GP is the

1 In geostatistics GP regression is known as *kriging* [56].

2 The term non-parameteric is a bit misleading. Non-parametric methods like GPs still have parameters, but in contrast to parametric approaches, the learned model has no characteristic structure and parameters. The structure/appearance of the model is derived from training data [32].

so-called *covariance function* or kernel function, which specifies the correlation between the data points. In contrast to other kernel methods like support vector machines (SVMs, [164]) or kernel least squares [61] a GP in addition provides confidence intervals thanks to the density representation of the regression result. Section 4.2 gives a short introduction to the most commonly used covariance functions and how the parameters of these functions can be learned from data. In Section 4.3, the major drawback of the non-parametric nature of GPs is addressed: the high computational demand given large-scale data sets. The state-of-the-art of Bayesian filtering given GP models is surveyed in Section 4.4. The contributions to GP filtering and efficient learning made in Papers H–K are summarized in Section 4.5.

4.1 Gaussian Processes

For GP regression, it is assumed that a set $\mathcal{D} = \{(\underline{x}_1, y_1), \ldots, (\underline{x}_n, y_n)\}$ of *training* data is drawn from the nonlinear mapping

$$\boldsymbol{y} = g(\underline{x}) + \boldsymbol{w}\,, \quad \text{with } \boldsymbol{w} \sim \mathcal{N}(0, \sigma^2) \tag{4.1}$$

where $\underline{x}_i \in \mathbb{R}^{n_x}$ are the inputs and $y_i \in \mathbb{R}$ are the observations or outputs, for $i = 1 \ldots n$. For brevity reasons, $\mathbf{X}_\mathcal{D} \triangleq [\underline{x}_1 \ \cdots \ \underline{x}_n]$ are all inputs and $\underline{y}_\mathcal{D}^\mathrm{T} \triangleq [y_1 \ \ldots \ y_n]$ are the corresponding observations in the following.

A GP is used to infer the latent function $g(.)$ from the data $\mathcal{D}$. It is completely defined by a *mean function* $\mu(\underline{x}) \triangleq \mathrm{E}\{\boldsymbol{g}(\underline{x})\}$, which specifies the expected output value, and a positive semi-definite *covariance function* $\kappa(\underline{x}, \underline{x}') \triangleq \mathrm{cov}\{\boldsymbol{g}(\underline{x}), \boldsymbol{g}(\underline{x}')\}$, which specifies the covariance between pairs of inputs. In the following, the zero mean function $\mu(\underline{x}) = 0$ is considered, which is a reasonable assumption and not a limitation in general [53, 138]. For a detailed discussion on covariance functions see Section 4.2.

A GP forms a Gaussian distribution over functions and thus, one can write $\boldsymbol{g} \sim \mathcal{GP}\left(\mu(\underline{x}), \kappa(\underline{x}, \underline{x}')\right)$. Accordingly, for a finite data set $\mathcal{D}$ the resulting density function of the outputs is a multivariate Gaussian

$$f(\underline{g}|\mathbf{X}_\mathcal{D}) = \mathcal{N}(\underline{0}, \mathbf{K}) \tag{4.2}$$

with $\underline{g}^{\mathrm{T}} \triangleq \left[g(\underline{x}_1) \; \ldots \; g(\underline{x}_n)\right]$ being the vector of latent function values at the training data inputs. This density is the so-called GP prior with *kernel* matrix $\mathbf{K} \triangleq \kappa(\mathbf{X}_{\mathcal{D}}, \mathbf{X}_{\mathcal{D}})$ comprising the elements $(\mathbf{K})_{ij} = \kappa(\underline{x}_i, \underline{x}_j), \forall \underline{x}_i, \underline{x}_j \in \mathcal{D}$. In Figure 4.1a on the next page an exemplary GP prior is depicted.

The *posterior* density of $g(.)$ for an arbitrary input $\underline{x}$ results from marginalizing the latent function over the training set according to

$$f(g(\underline{x}) \mid \underline{x}, \mathcal{D}) = \int f\left(g(\underline{x}), \underline{g} \,\middle|\, \underline{x}, \mathcal{D}\right) \mathrm{d}\underline{g} \,. \tag{4.3}$$

The integrand in (4.3) results from solving the Bayesian inference

$$f\left(g(\underline{x}), \underline{g} \,\middle|\, \underline{x}, \mathcal{D}\right) = c \cdot f\left(\underline{y}_{\mathcal{D}} \,\middle|\, \underline{g}\right) \cdot f\left(g(\underline{x}), \underline{g} \,\middle|\, \underline{x}, \mathbf{X}_{\mathcal{D}}\right) , \tag{4.4}$$

where $f\left(g(\underline{x}), \underline{g} \,\middle|\, \underline{x}, \mathbf{X}_{\mathcal{D}}\right)$ is the GP prior as in (4.2), but extended with the input $\underline{x}$. The likelihood $f\left(\underline{y}_{\mathcal{D}} \,\middle|\, \underline{g}\right)$ is given by

$$f\left(\underline{y}_{\mathcal{D}} \,\middle|\, \underline{g}\right) = \prod_{i=1}^{n} \mathcal{N}\left(y_i; g(\underline{x}_i), \sigma^2\right) \tag{4.5}$$

according to the model (4.1). Hence, all involved density functions are Gaussian and thus, the inference in (4.4) as well as the marginalization in (4.3) can be solved analytically. The desired posterior density in (4.3) is Gaussian with mean and variance

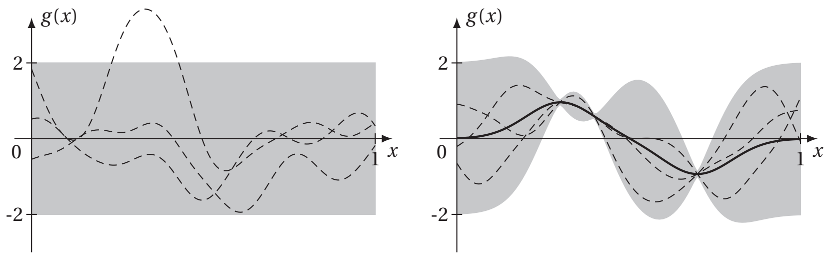
$$\mu_g(\underline{x}) = \mathrm{E}\{\boldsymbol{g}(\underline{x})\} = \underline{k}^{\mathrm{T}} \cdot \mathbf{K}_y^{-1} \cdot \underline{y}_{\mathcal{D}} \,, \tag{4.6}$$

$$\sigma_g^2(\underline{x}) = \mathrm{var}\{\boldsymbol{g}(\underline{x})\} = \kappa(\underline{x}, \underline{x}) - \underline{k}^{\mathrm{T}} \cdot \mathbf{K}_y^{-1} \cdot \underline{k} \,, \tag{4.7}$$

respectively, with $\underline{k}^{\mathrm{T}} \triangleq \kappa(\underline{x}, \mathbf{X}_{\mathcal{D}}) = [\kappa(\underline{x}_1, \underline{x}) \; \ldots \kappa(\underline{x}_n, \underline{x})]$ and $\mathbf{K}_y \triangleq \mathbf{K} + \sigma^2 \cdot \mathbf{I}_n$.

Example 17: GP Posterior

The latent function is chosen to be $g(x) = \sin(2\pi x)$ and the noise variance is $\sigma^2 = 0.02^2$. As covariance function the squared exponential function in (4.8) on page 109 is considered with hyperparameters $\alpha = 1$ and $\lambda = 0.1$. The training data comprises the inputs $\mathbf{X}_{\mathcal{D}} = [0.3 \; 0.4 \; 0.7]$ and outputs $\underline{y}_{\mathcal{D}}^{\mathrm{T}} = [0.96 \; 0.59 \; -0.93]$. The corresponding posterior GP is depicted in



(a) Three sample functions drawn from a GP prior.

(b) The posterior GP after observing three data points. The solid line corresponds to the posterior mean (4.6).

Figure 4.1: Gaussian process prior and posterior. The gray areas indicate 2-sigma confidence regions.

Figure 4.1b. To obtain this plot, the GP is evaluated for 100 values of $x \in [0\ 1]$, where the 100 values are chosen to be equidistant.

It can be seen that near a data point the posterior mean (4.6) is almost identical with the latent function and the posterior variance (4.7) is reduced significantly. Appart from data, the mean goes back to zero and the variance grows towards the value of the prior variance in Figure 4.1a. As a GP defines a distribution over functions, there is an infinite number of functions that can explain the data. Three of them are depicted in Figure 4.1b.

Based on (4.3), it is also possible to determine the density $f\big(y|x,\mathcal{D}\big)$ of the output $\boldsymbol{y}$. This density is also Gaussian with mean as in (4.6) and with variance

$$\sigma_y^2 = \sigma_g^2 + \sigma^2 \,.$$

If instead of the zero mean function an arbitrary mean function $\mu(\underline{x})$ is used, the posterior mean (4.6) changes to

$$\mu_g(\underline{x}) = \mu(\underline{x}) + \underline{k}^{\mathrm{T}} \cdot \mathbf{K}_y^{-1} \cdot (\underline{y}_{\mathcal{D}} - \underline{m})$$

with $\underline{m}^{\mathrm{T}} \triangleq \big[\mu(\underline{x}_1)\ \ldots\ \mu(\underline{x}_n)\big]$, while the posterior variance remains the same. For further reading on GP regression please be referred to [143].

4.2 Covariance Functions

So far it was assumed, that the covariance function is known. In real-world applications however, this assumption is typically not valid. Selecting an appropriate covariance function and determining its parameters is a crucial task. For a GP, the covariance function is of same importance as the kernel function in a SVM. It determines the relation between data points, characterizes the smoothness of the (latent) function, and specifies the impact of a data point on the overall function.

Not every function mapping two vectors to the real line can be considered as a covariance function. It has to be positive semi-definite and symmetric, i.e., the resulting kernel matrix has to be positive semi-definite[3] and symmetric matrix. This constraint is important as the kernel matrix acts as the covariance matrix of the Gaussian prior density (4.2).

4.2.1 Examples

In the following, some examplary covariance functions used throughout this thesis are introduced. Other commonly employed functions are discussed in [143].

Stationary Covariance Functions

Basically, covariance functions can be separated in two classes: stationary and non-stationary functions. A covariance function is called *stationary*[4] if it can be written as a function of the deviation $\Delta\underline{x} \triangleq \underline{x} - \underline{x}'$, i.e., $\kappa(\underline{x}, \underline{x}') = \kappa(\Delta\underline{x})$, for two arbitrary inputs $\underline{x}$ and $\underline{x}'$.

One of the most widely used covariance functions for GPs, but also in many other kernel-based machine learning algorithms like SVMs or radial-basis networks, is the *squared exponential* (SE)

$$\kappa(\Delta\underline{x}) = \alpha^2 \cdot \exp\left(-\tfrac{1}{2} \cdot \Delta\underline{x}^{\mathrm{T}} \Lambda^{-1} \Delta\underline{x}\right) . \tag{4.8}$$

Here, $\Lambda = \mathrm{diag}\left(\left[\lambda_1 \ \ldots \lambda_{n_x}\right]\right)$ is a diagonal matrix of the *characteristic length-scales* λ_i for each input dimension i and α^2 is the variance of the latent function $g(.)$.

3 All eigenvalues of the matrix are non-negative.

4 The term stationary stems from stochastic process theory, where a process is called (weakly) stationary if it possesses a constant mean function and a translation-invariant covariance function [99].

Such parameters of the covariance function are called the *hyperparameters* of the GP.

Non-Stationary Covariance Functions

Covariance functions that directly depend on the two inputs $\underline{x}$ and $\underline{x}'$ are called *non-stationary*. A widely used non-stationary kernel is the *polynomial* covariance function

$$\kappa\left(\underline{x}, \underline{x}'\right) = \alpha^2 \cdot \left(\underline{x}^{\mathrm{T}} \cdot \underline{x}' + c\right)^p \ , \tag{4.9}$$

with degree $p \in \mathbb{N}$, bias c, and variance α^2. Another popular covariance function is the *neural network* (NN) kernel

$$\kappa\left(\underline{x}, \underline{x}'\right) = \alpha^2 \cdot \sin^{-1}\left(\frac{\underline{x}^{\mathrm{T}} \Lambda^{-2} \underline{x}'}{\sqrt{\phi(\underline{x})\phi(\underline{x}')}}\right) , \tag{4.10}$$

with $\phi\left(\underline{x}\right) \triangleq 1 + \underline{x}^{\mathrm{T}} \Lambda^{-2} \underline{x}$, variance α^2, and the diagonal matrix Λ of characteristic length-scales.

Constructing Covariance Functions

Covariance functions can be constructed from existing ones like the above mentioned. Let $\kappa_1\left(\underline{x}, \underline{x}'\right)$ and $\kappa_2\left(\underline{x}, \underline{x}'\right)$ be two covariance functions, the

- *Sum* of both covariance functions $\kappa_1\left(\underline{x}, \underline{x}'\right) + \kappa_2\left(\underline{x}, \underline{x}'\right)$,
- *Product* of both covariance functions $\kappa_1\left(\underline{x}, \underline{x}'\right) \cdot \kappa_2\left(\underline{x}, \underline{x}'\right)$, and
- *Scaling* $\phi\left(\underline{x}\right) \cdot \kappa_1\left(\underline{x}, \underline{x}'\right) \cdot \phi\left(\underline{x}'\right)$ with a deterministic function $\phi\left(\underline{x}\right)$

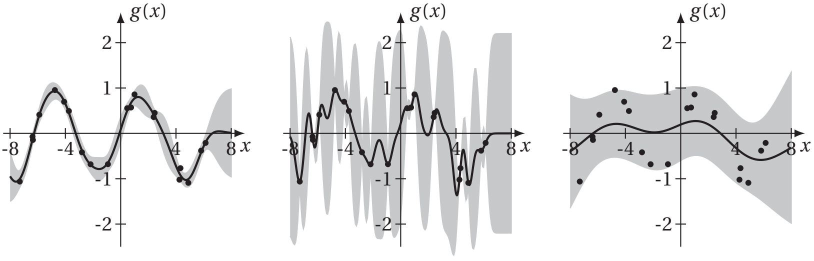
lead again to valid covariance functions.

Example 18: SE plus Noise

In many applications, the SE kernel (4.8) is augmented by the noise variance of the model (4.1) according to

$$\kappa\left(\Delta\underline{x}\right) = \alpha^2 \cdot \exp\left(-\tfrac{1}{2} \cdot \Delta\underline{x}^{\mathrm{T}} \Lambda^{-1} \Delta\underline{x}\right) + \sigma^2 \cdot \delta\left(\Delta\underline{x}\right) . \tag{4.11}$$

This allows treating the noise variance σ^2 as an additional hyperparameter to be learned from data (see next section).



(a) $\underline{\theta}^{\mathrm{T}} = [1\ 0.5\ 0.1]$ **(b)** $\underline{\theta}^{\mathrm{T}} = [0.3\ 1.1\ 0.01]$ **(c)** $\underline{\theta}^{\mathrm{T}} = [3\ 1.2\ 0.9]$

Figure 4.2: Posterior GPs for different hyperparameters $\underline{\theta}^{\mathrm{T}} = [\alpha\ \lambda\ \sigma]$. The data points are indicated by the black circles.

4.2.2 Hyperparameter Learning

Each covariance function possesses several parameters, e.g., the variance α^2 and the length-scale matrix Λ in case of the SE function (4.8). As mentioned above, these so-called hyperparameters need to be adjusted such that the covariance function fits to the given application[5]. In case of GPs, the hyperparameters can be determined or *learned* from the given training data $\mathcal{D}$. In the following, $\underline{\theta}$ comprises all hyperparameters of a given covariance function.

Example 19: Varying Hyperparameters

The latent function is chosen to be $g(x) = \sin(x)$. 20 data points are drawn from this model. Based on this training data, three different GPs with covariance function (4.11) are learned. The resulting GPs are depicted in Figure 4.2. The hyperparameters chosen for the GP in Figure 4.2a are close to optimal. Accordingly, the GP approximates the latent function very well and the posterior variances are adequate. In Figure 4.2b and Figure 4.2c, however, the hyperparameters are not chosen appropriately. While the GP in Figure 4.2b is clearly overfitting, the GP in Figure 4.2c provides an oversmoothed result.

5 Hyperparameter learning is part of the *model selection* problem (recall Sections 3.3.2 and 3.4.3).

To determine appropriate hyperparameters, the standard approach proposed in [117] is *evidence maximization*, which essentially tries to maximize the (positive) log-likelihood (see Appendix B.4)

$$\begin{aligned} \underline{\theta}^* &= \arg\max_{\underline{\theta}} \left\{ \log f\left(\underline{y}_{\mathcal{D}} \middle| \underline{g}, \underline{\theta}\right) \right\} \\ &= \arg\max_{\underline{\theta}} \Big\{ \underbrace{-\tfrac{1}{2} \cdot \underline{y}_{\mathcal{D}}^{\mathrm{T}} \mathbf{K}_y^{-1} \underline{y}_{\mathcal{D}}}_{\text{data-fit}} \underbrace{- \tfrac{1}{2} \log |2\pi \mathbf{K}_y|}_{\text{complexity}} \Big\} , \end{aligned} \tag{4.12}$$

where the likelihood $f\left(\underline{y}_{\mathcal{D}} \middle| \underline{g}, \underline{\theta}\right)$ corresponds to (4.5) and $\mathbf{K}_y$ depends on $\underline{\theta}$. The maximization in (4.12) yields a trade-off between fitting the data as good as possible and obtaining a model of low complexity. Hence, this maximization is a realization of the famous *Occam's razor*[6], which aims for the simplest model possible that is consistent with the data.

Evidence maximization is a non-convex optimization problem, for which no closed-form solution exists and which highly depends on the given training data. However, practice showed that even though the global optimum might not be found, the resulting GP based on suboptimal hyperparameters still explains the data well enough [51]. An alternative to evidence maximization is cross-validation, which is typically employed in SVM training. Cross-validation however is computationally demanding, requires MC simulation, and exploits only a fraction of the entire training data.

4.3 Large Data Sets

The computational complexity for calculating the posterior mean (4.6) and variance (4.7) is mainly dominated by the inversion of the matrix $\mathbf{K}_y$ and evaluating matrix-vector products. Inverting $\mathbf{K}_y$ or the more numerically robust alternative of computing its Cholesky decomposition is in $\mathcal{O}(n^3)$. Thus, for a large number of data points, this calculation becomes very demanding. Fortunately, the Cholesky decomposition of $\mathbf{K}_y$ can be determined off-line if the entire data set is given a priori. This, however is not the case for *streaming data*, where the data

6 "Pluralitas non est ponenda sine necessitate" (plurality should not be posited without necessity) stated by William of Ockham (also spelt Occam), an English Franciscan friar and philosopher, c. 1287–1347.

becomes available on-line during run-time. In this case, the matrix and thus its Cholesky decomposition has to be updated. In order to avoid a complete recalculation of the Cholesky decomposition, so-called *rank-1 Cholesky updates* can be performed for introducing new data points [141].

Given the inverse or Cholesky decomposition of $\mathbf{K}_y$, the complexity of the posterior mean is in $\mathcal{O}(n)$ and of the posterior variance is in $\mathcal{O}(n^2)$. For small data sets, these computations are affordable, but for large problems—tens of thousands data points—both storing $\mathbf{K}_y$ or its decomposition as well as solving the matrix-vector products in (4.6) and (4.7) is prohibitive. To overcome this problem, many approximate GP regression methods have been proposed, where some of the most popular are briefly reviewed in the following.

4.3.1 Active Set Approaches

In [140] a unifying framework for so-called *active set* approaches has been derived. Here, instead of processing the entire training data set, only a subset of the data points—the active set with $m \ll n$ data points—is used. The matrix $\mathbf{K}_y$ is replaced by a matrix that is based on the active set, which reduces the computational complexity to $\mathcal{O}(m^2 \cdot n)$ for matrix computations, $\mathcal{O}(m)$ for calculating the posterior mean, and $\mathcal{O}(m^2)$ for the posterior variance.

Several active set approaches have been proposed independently, but by means of the unifying framework in [140] it can be shown that all these approaches merely differ on modifications of the original GP prior (4.2). The *subset of regressors* approach [175, 204] is probably the simplest realization of an active set method, where the active set is either chosen randomly from the training set or by means of a greedy selection algorithm. The *sparse on-line GP* (SOGP) regression proposed in [47] in addition allows adding and removing data points to the active set at runtime, which is desirable when the training data set is not given entirely a priori. The *sparse pseudo-input GP* (SPGP, [176]) is generally regarded as one of the most effective active set approaches. In contrast to other methods, the active set has not to be a subset of the training data: elements of the active set can be located anywhere in the input domain. For this purpose, the active set is optimized by means of evidence maximization (4.12).

4.3.2 Local Approaches

To speed up GP regression, *local* or partitioning approaches split the training data into p data sets, where for each data set a separate (local) GP is learned. For calculating the posterior mean and variance, the individual estimates of the separate GPs are combined. In doing so, the complexity is $\mathcal{O}(p \cdot m^3)$ for matrix calculation, $\mathcal{O}(p \cdot m)$ for mean calculation, and $\mathcal{O}(p \cdot m^2)$ for calculating the variance, where $m = {}^{p}/_{n}$ is the number of data points per set. In contrast to active set methods, local approaches make use of the entire training data.

Two instances of a local approach are the *Bayesian committee machine* proposed in [198] as well as the *product of GP experts* proposed in [**?**]. In both methods partitioning is performed off-line. In contrast, the *local GP* [131] is an on-line algorithm, where incoming data is assigned to the nearest data set. To update the local kernel matrix, the above mentioned rank-1 updating is utilized.

4.3.3 Algebraic Tricks

Besides the above approximate regression approaches, there exist several approximations that tackle the algebraic operations necessary to compute the posterior mean and variances. This "algebraic tricks" can be used in combination with the above methods in order to further speed up GP regression. *Skilling's method* [174] for instance is an iterative procedure to solve matrix-vector operations of the kind $\mathbf{K}^{-1} \cdot \underline{x}$, which appear in both (4.6) and (4.7). If this method is terminated after k iterations, the complexity is $\mathcal{O}(k \cdot n^2)$ instead of $\mathcal{O}(n^3)$.

If the covariance function possesses as special structure, sparse versions of the Matrix $\mathbf{K}_y$ can be obtained allowing for efficient matrix-operations. This holds for covariance functions with compact support like the Matérn kernel. In case of a compact support, the complexity of the aforementioned rank-1 update reduces from $\mathcal{O}(n^2)$ down to $\mathcal{O}(n)$ [141]. In case of stationary covariance functions, the function values merely depend on the distance between pairs of inputs. By defining a distance threshold, only those pairs of training data points are considered that are within the threshold [202]. In addition, the distance dependency allows storing the training set in a *kd-tree*, which provides a rapid access to the data [72, 202].

4.3.4 Open Issues

Most of the proposed approximations for large data sets are not suitable for streaming data, which requires performing GP regression on-line at runtime. Among the on-line approaches, those utilizing rank-1 updating or partitioning merely alleviate the complexity problem. They still suffer from an increasing amount of data points, i.e., the computational and memory demand grows over time.

4.4 Nonlinear Filtering

Given a GP model of the latent function $g(.)$, it is so far possible to calculate the mean and variance for a deterministic input $\underline{x}$ according to (4.6) and (4.7), respectively. In order to perform Gaussian filtering, however, the input—more precisely the state—has to be a random vector. Accordingly, one has to solve the moment integral (2.8) for $\boldsymbol{g} \sim \mathcal{GP}\left(\mu, \kappa\right)$. As $\boldsymbol{g}$ is a random vector, it is in addition necessary to marginalize out the uncertainty over $g(.)$ in order to obtain the desired moments. Thus, for Gaussian filtering with GP models, the moment integrals (2.8) become

$$
\begin{aligned}
\mu_y &= \iint g(\underline{x}) \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_x, \mathbf{C}_x\right) \cdot \mathcal{N}\left(g; \mu_g(\underline{x}), \sigma_g^2(\underline{x})\right) \mathrm{d}g\,\mathrm{d}\underline{x}\,, \\
\sigma_y^2 &= \iint \left(g(\underline{x}) - \mu_y\right)\left(g(\underline{x}) - \mu_y\right)^{\mathrm{T}} \cdot \\
&\qquad\qquad \mathcal{N}\left(\underline{x}; \underline{\mu}_x, \mathbf{C}_x\right) \cdot \mathcal{N}\left(g; \mu_g(\underline{x}), \sigma_g^2(\underline{x})\right) \mathrm{d}g\,\mathrm{d}\underline{x} + \sigma^2\,, \\
\sigma_{xy} &= \iint \left(\underline{x} - \underline{\mu}_x\right)\left(g(\underline{x}) - \mu_y\right)^{\mathrm{T}} \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_x, \mathbf{C}_x\right) \cdot \mathcal{N}\left(g; \mu_g(\underline{x}), \sigma_g^2(\underline{x})\right) \mathrm{d}g\,\mathrm{d}\underline{x}\,,
\end{aligned}
\tag{4.13}
$$

containing the additional integration over the latent function/GP $g(.)$. Here, the Gaussian density $\mathcal{N}\left(g; \mu_g(\underline{x}), \sigma_g^2(\underline{x})\right)$ is extracted from the GP, where the mean and variance are according to (4.6) and (4.7), respectively.

The number of approaches for Gaussian filtering with GP models proposed so far is limited. For arbitrary covariance functions, [68] proposed a first-order and second-order Taylor-series expansion of the posterior mean (4.6) and variance (4.7), respectively. Similarly, a first-order Taylor-series expansion of (4.6) was used in [103], which yields the so-called *GP extended Kalman filter* (GP-EKF). In [104], the unscented transform (see Section 2.2.5) is employed instead. The sample points are propagated through (4.6) and the desired moments (4.13) are then calculated by means of sample mean and variance, respectively. Accordingly, the Gaussian filter is named *GP unscented Kalman filter* (GP-UKF). In [67] it is shown, that for the SE covariance function, the mean μ_y and the variance σ_y^2 can be determined analytically. But in contrast to [103, 104], no full Gaussian filter that also involves a measurement update was provided. Furthermore, only a scalar output $\boldsymbol{y}$ is supported.

4.5 Contributions

In Section 4.5.1 and Section 4.5.2, a novel GP filtering and smoothing algorithm is proposed that is based on analytic moment matching, i.e., the moment integrals are solved exactly given a GP representation of the latent function $g(.)$. These results are based on Paper H and Paper I. An approximate on-line GP regression algorithm with on-line hyperparameter learning that was proposed in Paper J and Paper K is summarized in Section 4.5.3 and Section 4.5.4, respectively.

4.5.1 Gaussian Process Filtering

So far, only one-dimensional outputs $\boldsymbol{y}$ have been considered. For real-world applications however, the output is typically of higher dimension. Thus, in the following the focus is on a nonlinear transformation as in (2.6) with multivariate output $\underline{\boldsymbol{y}} \in \mathbb{R}^{n_y}$ and multivariate noise $\underline{\boldsymbol{w}} \sim \mathcal{N}(\underline{0}, \mathbf{C}_w)$ with $\mathbf{C}_w = \operatorname{diag}([\sigma_1^2 \ldots \sigma_{n_y}^2])$. It is assumed that a separate GP $\mathcal{GP}(\mu_a, \kappa_a)$ is trained for each dimension $a = 1 \ldots n_y$. For training of the ath GP, the training data set $\mathcal{D}_a = \{\mathbf{X}_\mathcal{D}, \underline{y}_a\}$ consisting of n data points $(\underline{x}_i, y_{a,i})$, $i = 1 \ldots n$ is used.

Mean Vector $\underline{\mu}_y$

To obtain a Gaussian filter in this setup, the moment integrals (4.13) have to be solved. Due to the representation of the each dimension by means of a separate GP, the calculation of the mean vector $\underline{\mu}_y$ is also performed dimension-wise. The ath element $\mu_{y,a}$ of $\underline{\mu}_y$ is given by

$$\mu_{y,a} = \int \mu_{g,a}(\underline{x}) \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_x, \mathbf{C}_x\right) \mathrm{d}\underline{x} = \sum_{i=1}^{n} \beta_{a,i} \cdot \int \kappa_a(\underline{x}, \underline{x}_i) \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_x, \mathbf{C}_x\right) \mathrm{d}\underline{x} \quad (4.14)$$

with $\mu_{g,a}(.)$ being the ath element of the posterior mean (4.6) and $\beta_{a,i}$ being the ith element of the vector $\underline{\beta}_a \triangleq \left(\mathbf{K}_a + \sigma_a^2 \cdot \mathbf{I}_n\right)^{-1} \cdot \underline{y}_a$ with kernel matrix $\mathbf{K}_a$ consisting of the elements $\left(\mathbf{K}_a\right)_{e,f} = \kappa_a\left(\underline{x}_e, \underline{x}_f\right)$ for all $\underline{x}_e$, $\underline{x}_f \in \mathbf{X}_{\mathcal{D}}$. The second equality in (4.14) follows from writing the posterior mean function $\mu_{g,a}(.)$ as a finite sum over the covariance functions [143].

In general the integral in (4.14) cannot be solved in closed form due to the nonlinear covariance function, but if the covariance function belongs to one of the function types listed in Section 2.5.2, an analytical solution is possible. This holds for instance for the SE function (4.8) or the polynomial function (4.9). For the SE covariance function with signal variance α_a^2 and matrix of characteristic length-scales Λ_a, (4.14) can be simplified to

$$\mu_{y,a} = \underline{\beta}_a^{\mathrm{T}} \cdot \underline{q}_a \tag{4.15}$$

with $\underline{q}_a$ comprising the elements

$$q_{a,i} \triangleq \alpha_a^2 \cdot \left|\mathbf{C}_x \Lambda_a^{-1} + \mathbf{I}\right|^{-\frac{1}{2}} \cdot \exp\left(-\tfrac{1}{2}\left(\underline{x}_i - \underline{\mu}_x\right)^{\mathrm{T}} \left(\mathbf{C}_x + \Lambda_a\right)^{-1} \left(\underline{x}_i - \underline{\mu}_x\right)\right), \tag{4.16}$$

for $i = 1 \ldots n$. A similar solution can be obtained for the polynomial covariance function (4.9).

Covariance Matrix $\mathbf{C}_y$

For calculating the covariance $\mathbf{C}_y$, again each element of the matrix is determined individually. At first, the off-diagonal elements $a, b = 1 \ldots n_y$ with $a \neq b$ of $\mathbf{C}_y$, i.e., the cross-covariances, are considered. These elements are given by

$$\begin{aligned}\sigma_{y,ab}^2 &= \int \mu_{g,a}(\underline{x}) \cdot \mu_{g,b}(\underline{x}) \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_x, \mathbf{C}_x\right) \mathrm{d}\underline{x} - \mu_{y,a} \cdot \mu_{y,b} \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{a,i} \cdot \beta_{b,j} \cdot \int \kappa_a\left(\underline{x}, \underline{x}_i\right) \cdot \kappa_b\left(\underline{x}, \underline{x}_j\right) \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_x, \mathbf{C}_x\right) \mathrm{d}\underline{x} - \mu_{y,a} \cdot \mu_{y,b},\end{aligned}$$

with $\mu_{y,a}$ and $\mu_{y,b}$ according to (4.15). For SE covariance functions, the cross-covariance can be further simplified to

$$\sigma_{y,ab}^2 = \underline{\beta}_a^{\mathrm{T}} \cdot \mathbf{Q} \cdot \underline{\beta}_b - \mu_{y,a} \cdot \mu_{y,b} \tag{4.17}$$

with elements of $\mathbf{Q} \in \mathbb{R}^{n \times n}$ according to

$$Q_{ij} = \frac{\exp\left(n_{ij}^2\right)}{\sqrt{|\mathbf{R}|}} \,, \tag{4.18}$$
$$n_{ij}^2 = \log\left(\alpha_a^2\right) + \log\left(\alpha_b^2\right) - \tfrac{1}{2}\left(\underline{\zeta}_i^{\mathrm{T}} \Lambda_a^{-1} \underline{\zeta}_i + \underline{\zeta}_j^{\mathrm{T}} \Lambda_b^{-1} \underline{\zeta}_j - \underline{z}_{ij}^{\mathrm{T}} \mathbf{R}^{-1} \mathbf{C}_x \underline{z}_{ij}\right) ,$$

where $\mathbf{R} \triangleq \mathbf{C}_x\left(\Lambda_a^{-1} + \Lambda_b^{-1}\right) + \mathbf{I}$, $\underline{\zeta}_i \triangleq \underline{x}_i - \underline{\mu}_x$, and $\underline{z}_{ij} \triangleq \Lambda_a^{-1} \underline{\zeta}_i + \Lambda_b^{-1} \underline{\zeta}_j$.

The diagonal elements ($a = b$) of $\mathbf{C}_y$ comprise in addition to (4.17) a term reflecting the noise variance σ_a^2 and the uncertainty of the GP

$$\alpha_a^2 - \operatorname{Tr}\left(\left(\mathbf{K}_a + \sigma_a^2 \mathbf{I}\right)^{-1} \mathbf{Q}\right) + \sigma_a^2 \,. \tag{4.19}$$

Hence, the desired elements of covariance matrix $\mathbf{C}_y$ are given by

$$\sigma_{y,ab}^2 = \begin{cases} \text{Eq. (4.17)} + \text{Eq. (4.19)} & \text{if } a = b \\ \text{Eq. (4.17)} & \text{otherwise} \end{cases} . \tag{4.20}$$

Cross-Covariance Matrix $\mathbf{C}_{xy}$

It remains to compute the cross-covariance $\mathbf{C}_{xy}$ to fully determine a Gaussian filter for GP models. Integrating out $\underline{g}$, the cross-covariance can be simplified to

$$\mathbf{C}_{xy} = \int \underline{x} \cdot \left(\underline{\mu}_g(\underline{x})\right)^{\mathrm{T}} \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_x, \mathbf{C}_x\right) \mathrm{d}\underline{x} - \underline{\mu}_x \cdot \underline{\mu}_y^{\mathrm{T}} \,.$$

By writing $\underline{\mu}_g(\underline{x})$ as a finite sum over covariance functions, the ath column of $\mathbf{C}_{xy}$, $a = 1 \ldots n_y$ can be written as

$$\sum_{i=1}^{n} \beta_{a,i} \cdot \int \underline{x} \cdot \kappa_a\left(\underline{x}, \underline{x}_i\right) \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_x, \mathbf{C}_x\right) \mathrm{d}\underline{x} - \underline{\mu}_x \cdot \mu_{y,a} \,. \tag{4.21}$$

With a SE covariance function, this term can be solved analytically to

$$\sum_{i=1}^{n} \beta_{a,i} \cdot q_{a,i} \cdot \mathbf{C}_x \cdot \left(\mathbf{C}_x + \Lambda_a\right)^{-1} \cdot \left(\underline{x}_i - \underline{\mu}_x\right) , \tag{4.22}$$

where the analytical solution (4.15) for $\underline{\mu}_{y,a}$ has been substituted and $q_{a,i}$ coincides with (4.16).

The Gaussian filter for GP models based on the expressions (4.15), (4.20), and (4.22) for calculating the moments in (4.13) is named *GP assumed density filter* (GP-ADF) in the following.

Example 20: One Step Filtering

The nonlinear dynamic system

$$\boldsymbol{x}_k = \frac{\boldsymbol{x}_{k-1}}{2} + \frac{25\boldsymbol{x}_{k-1}}{1+\boldsymbol{x}_{k-1}^2} + \boldsymbol{w}_k\,, \qquad \text{with } \boldsymbol{w}_k \sim \mathcal{N}\big(0, \sigma_w^2 = 0.2^2\big)\,, \tag{4.23}$$

$$\boldsymbol{z}_k = 5 \cdot \sin(\boldsymbol{x}_k) + \boldsymbol{v}_k\,, \qquad \text{with } \boldsymbol{v}_k \sim \mathcal{N}\big(0, \sigma_v^2 = 0.2^2\big)\,, \tag{4.24}$$

is considered, which is a modified version of the model used in [59, 101]. The standard deviation of the initial state is set to be $\sigma_0^x = 0.5$, i.e., the initial uncertainty is fairly high. The system and measurement noises are relatively small considering the amplitudes of the system function and the measurement function. For the numerical analysis, the mean values $\mu_{0,i}^x$, $i = 1\ldots100$, are placed equidistantly on the interval $[-3,3]$. Then, a single (initial) state $x_{0,i}$ is sampled from $\mathcal{N}\left(\mu_{0,i}^x, (\sigma_0^x)^2\right)$, $i = 1\ldots100$.

For the dynamic system in (4.23)–(4.24), the performance of a single prediction and measurement update of the EKF, the UKF, the CKF, the GP-UKF, and the GP-ADF is compared against the ground truth, which is approximated by means of a near-optimal sampling-based Gaussian filter (denoted as *Gibbs-filter*, [54]) as well as a PF with 200 particles. Compared to the evaluation of longer trajectories, evaluating a single filtering step makes it easier to analyze the estimates of individual filtering algorithms.

Table 4.1 summarizes the performances (rmse, mae, nll) of all filters for estimating the latent state $\boldsymbol{x}$. The results in the table are based on averages over 1,000 test runs and 100 randomly sampled initial states per test run. The table also reports the 95% standard error of the expected performances.

Table 4.1 indicates that the GP-ADF is the most robust filter and statistically significantly outperforms all filters but the Gibbs-filter and the PF. Amongst all other filters the GP-ADF is the closest Gaussian filter to the computationally expensive Gibbs-filter [54]. Note that the PF is not a Gaussian filter and is able to express multi-modality in densities. Therefore, its performance

Table 4.1: Average performances (rmse, mae, nll) with standard errors (95% confidence interval) and p-values testing the hypothesis that the other filters are better than the GP-ADF using a one-sided t-test. The green color highlights the near-optimal results of the Gibbs-filter and the PF.

	rmse		mae		nll	
	average	p-value	average	p-value	average	p-value
EKF	3.62 ± 0.212	4.1×10^{-2}	2.36 ± 0.176	0.38	$3.05 \times 10^{3} \pm 3.02 \times 10^{2}$	$< 10^{-4}$
UKF	10.5 ± 1.08	$< 10^{-4}$	8.58 ± 0.915	$< 10^{-4}$	25.6 ± 3.39	$< 10^{-4}$
CKF	9.24 ± 1.13	2.8×10^{-4}	7.31 ± 0.941	4.2×10^{-4}	$2.22 \times 10^{2} \pm 17.5$	$< 10^{-4}$
GP-UKF	5.36 ± 0.461	7.9×10^{-4}	3.84 ± 0.352	3.3×10^{-3}	6.02 ± 0.497	$< 10^{-4}$
GP-ADF	$\mathbf{2.85 \pm 0.174}$	—	$\mathbf{2.17 \pm 0.151}$	—	$\mathbf{1.97 \pm 6.55 \times 10^{-2}}$	—
Gibbs	$\mathbf{2.82 \pm 0.171}$	0.54	$\mathbf{2.12 \pm 0.148}$	0.56	$\mathbf{1.96 \pm 6.62 \times 10^{-2}}$	0.55
PF	$\mathbf{1.57 \pm 7.66 \times 10^{-2}}$	1.0	$\mathbf{0.36 \pm 2.28 \times 10^{-2}}$	1.0	$\mathbf{1.03 \pm 7.30 \times 10^{-2}}$	1.0

is typically better than the one of Gaussian filters. The difference between the PF and a near-optimal Gaussian filter, the Gibbs-filter, is expressed in Table 4.1. The performance difference essentially depicts how much is lost by using a Gaussian filter instead of a particle filter.

The poor performance of the EKF is due to linearization errors. The filters based on small sample approximations of densities (UKF, GP-UKF, CKF) suffer from the degeneracy of these approximations, which is illustrated in Figure 4.3 on the next page. Note that the CKF uses a smaller set of sample points than the UKF (recall Section 2.2.5), which makes the CKF statistically even less robust than the UKF.

4.5.2 Gaussian Process Smoothing

For an RTSS given a GP representation of the system and measurement function, most of the ingredients are already derived by means of the above GP-ADF. The only missing component is the cross-covariance matrix $\mathbf{C}_{k|k+1}$ in (2.33). For its calculation, one can follow the above derivation of the cross-covariance matrix $\mathbf{C}_{xy}$. According to (4.21), the ath column of $\mathbf{C}_{k|k+1}$ can be written as

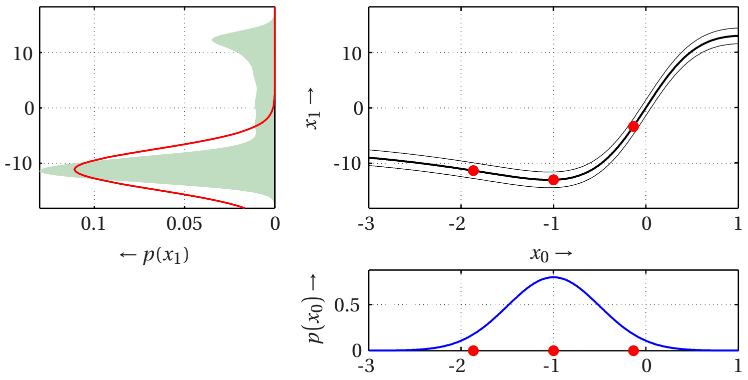
$$\sum_{i=1}^{n} \beta_{a,i} \cdot \int \underline{x} \cdot \kappa_a\big(\underline{x}, \underline{x}_i\big) \cdot \mathcal{N}\left(\underline{x}; \underline{\mu}_k^e, \mathbf{C}_k^e\right) \mathrm{d}\underline{x} - \underline{\mu}_k^e \cdot \mu_{k,a}^p \,, \tag{4.25}$$

with $\mathcal{N}\left(\underline{x}; \underline{\mu}_k^e, \mathbf{C}_k^e\right)$ being the posterior state density and $\underline{\mu}_{k,a}^p$ being the ath element of the predicted mean $\underline{\mu}_k^p$. For an SE covariance function, a closed-form expression of (4.25) can be obtain given by

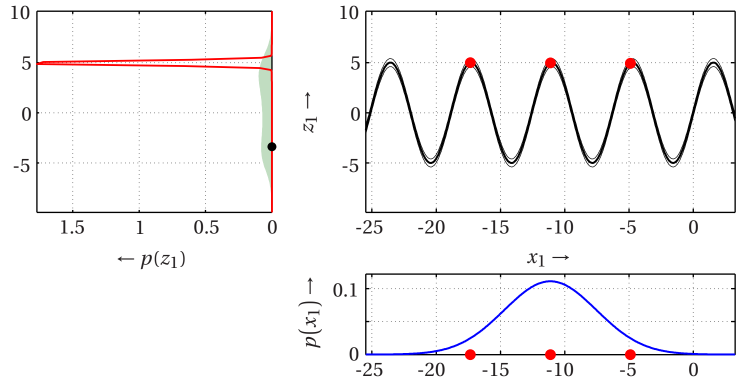
$$\sum_{i=1}^{n} \beta_{a,i} \cdot q_{a,i} \cdot \mathbf{C}_k^e \cdot \left(\mathbf{C}_k^e + \Lambda_a\right)^{-1} \cdot \left(\underline{x}_i - \underline{\mu}_k^e\right) \,, \tag{4.26}$$

with $q_{a,i}$ as in (4.16), but with $\mathbf{C}_x$ and $\underline{\mu}_x$ being substituted with $\mathbf{C}_k^e$ and $\underline{\mu}_k^e$, respectively.

Altogether, the *GP Rauch-Tung-Striebel smoother* (GP-RTSS) utilizes the GP-ADF of the previous section for the forward sweep (prediction and measurement update). For the actual smoothing, (4.26) is used for calculating the cross-



(a) UKF time update $p(x_1|\emptyset)$, which misses out substantial probability mass of the true predictive distribution.



(b) UKF determines $p(z_1|\emptyset)$, which is too sensitive and cannot explain the actual measurement z_1 (black dot, left sub-figure).

Figure 4.3: Degeneracy of the unscented transformation (UT) underlying the UKF. Input distributions to the UT are the Gaussians in the sub-figures at the bottom in each panel. The functions the UT is applied to are shown in the top right sub-figures, i.e, the transition mapping (4.23) in (a) and the measurement mapping (4.24) in (b). Sigma points are marked by red dots. The predictive distributions are shown in the left sub-figures of each panel. The true predictive distributions are the shaded areas; the UT predictive distributions are the solid Gaussians. The predictive distribution of the time update in (a) equals the input distribution at the bottom of (b).

covariance $\mathbf{C}_{k|k+1}$, which is required for determining the smoothed Gaussian state according to (2.34).

The computational complexity of prediction, measurement update, and smoothing (after training the GPs) is in $\mathcal{O}\big(K \cdot n^2 \cdot \big(n_x^3 + n_z^3\big)\big)$ due to matrix inversions, matrix multiplications, and the computation of the **Q**-matrix (4.18). For comparison, Kalman filter and RTSS scale with $\mathcal{O}\big(K \cdot \big(n_x^3 + n_z^3\big)\big)$.

Example 21: Pendulum Tracking

The pendulum tracking example taken from [52] is considered for comparing the performances of four filters and smoothers: the EKF/EKS, the UKF/URTSS, the GP-UKF/GP-URTSS, the CKF/CKS, the Gibbs-filter/smoother, and the GP-ADF/GP-RTSS. The pendulum has mass $m = 1\,\mathrm{kg}$ and length $l = 1\,\mathrm{m}$. The state $\underline{\boldsymbol{x}} = [\dot{\boldsymbol{\varphi}}\ \boldsymbol{\varphi}]^{\mathrm{T}}$ of the pendulum is given by the angle $\boldsymbol{\varphi}$ (measured anti-clockwise from hanging down) and the angular velocity $\dot{\boldsymbol{\varphi}}$. The pendulum can exert a constrained torque $u \in [-5,5]\,\mathrm{Nm}$. A frictionless system is assumed such that the system function $\underline{a}(.)$ is

$$\underline{a}\big(\underline{x}_k, u_k\big) = \int_k^{k+\Delta_k} \begin{bmatrix} \frac{u(\tau) - 0.5\, mlg \sin\left(\varphi(\tau)\right)}{0.25\, ml^2 + I} \\ \dot{\varphi}(\tau) \end{bmatrix} \mathrm{d}\tau \,, \tag{4.27}$$

where I is the moment of inertia and g the acceleration of gravity. Then, the successor state

$$\underline{\boldsymbol{x}}_{k+1} = \underline{\boldsymbol{x}}_{k+\Delta_k} = \underline{a}\big(\underline{\boldsymbol{x}}_k, u_k\big) + \underline{\boldsymbol{w}}_k \,, \quad \text{with } \underline{\boldsymbol{w}}_k \sim \mathcal{N}\big(\underline{0}, \operatorname{diag}\big(0.5^2, 0.1^2\big)\big)$$

is computed using an ODE solver for (4.27) with a zero-order hold control signal $u(\tau)$. The torque is sampled randomly according to $u \sim \mathcal{U}[-5,5]\,\mathrm{Nm}$ and implemented using a zero-order-hold controller. Every time increment $\Delta_k = 0.2\,\mathrm{s}$, the state is observed according to

$$\boldsymbol{z}_k = \arctan\left(\frac{-1 - l \cdot \sin(\boldsymbol{\varphi}_k)}{0.5 - l \cdot \cos(\boldsymbol{\varphi}_k)}\right) + \boldsymbol{v}_k \,, \quad \text{with } \boldsymbol{v}_k \sim \mathcal{N}\big(0, 0.05^2\big) \,. \tag{4.28}$$

Trajectories of length $K = 6\,\mathrm{s} = 30$ time steps are started from a state sampled from $\mathcal{N}\big(\underline{\mu}_0^x, \mathbf{C}_0^x\big)$ with mean vector $\underline{\mu}_0^x = [0\ 0]^{\mathrm{T}}$ and covariance matrix $\mathbf{C}_0^x = \operatorname{diag}\big(0.01^2, (\pi/16)^2\big)$. For each trajectory, GP models $\mathcal{GP}_a$ (system function) and $\mathcal{GP}_h$ (measurement function) were learned based on randomly generated data using either 250 or 20 data points.

Table 4.2: Averaged filtering and smoothing performances with 95% confidence intervals.

Filters	nll	Smoothers	nll
EKF	$1.6 \times 10^2 \pm 29.1$	EKS [120]	$\mathbf{3.3 \times 10^2 \pm 60.5}$
UKF	6.0 ± 3.02	URTSS [158]	$\mathbf{17.2 \pm 10.0}$
CKF	28.5 ± 9.83	CKS [54]	$\mathbf{72.0 \pm 25.1}$
GP-UKF$_{250}$	4.4 ± 1.32	GP-URTSS$_{250}$ [53]	$\mathbf{10.3 \pm 3.85}$
GP-ADF$_{250}$	$\mathbf{1.44 \pm 0.117}$	GP-RTSS$_{250}$	$\mathbf{1.04 \pm 0.204}$
GP-ADF$_{20}$	6.63 ± 0.149	GP-RTSS$_{20}$	6.57 ± 0.148

Table 4.2 reports the values of the nllmeasure for the EKF/EKS, the UKF/URTSS, the GP-UKF/GP-URTSS, the GP-ADF/GP-RTSS, and the CKF/CKS, averaged over 1,000 MC runs. The GP-RTSS is the only method that consistently reduces the nll value compared to the corresponding filtering algorithm. Increased nll values (red color in Table 4.2) occur when the state density cannot explain the state/measurement. A detailed example of this can be found in Paper I. Even with only 20 training points, the GP-ADF/GP-RTSS outperforms the EKF/EKS, UKF/URTSS, CKF/CKS.

4.5.3 Recursive Gaussian Process Regression

As mentioned in Section 4.3, the complexity of GP regression scales cubically with the number of training data points. This complexity can be reduced by means of sparse approximations. Most of these approximations however only work in an off-line mode, i.e., all training data has to be known a priori and is processed in a batch. This is not suitable for streaming data. The *recursive Gaussian Process* (RGP) regression approach introduced next allows for both a sparse representation *and* on-line processing. For this purpose, the latent function is represented by means of a finite set of so-called *basis vectors*.

Let $\mathbf{X} \triangleq [\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_m]$ be the matrix of locations of the basis vectors, where the number of basis vectors m is significantly lower than the size n of $\mathcal{D}$, i.e., $m \ll n$. Furthermore, $\underline{g} \triangleq g(\mathbf{X})$ are the (unkown) values of the latent function at the locations $\mathbf{X}$. The basis vectors can be considered an active set allowing a sparse GP representation. In contrast to most other active set approaches, the basis vectors are updated *on-line* with new observations $\underline{y}_k$ at inputs $\mathbf{X}_k \triangleq \left[\underline{x}_{k,1}, \underline{x}_{k,2}, \ldots, \underline{x}_{k,n_k}\right]$

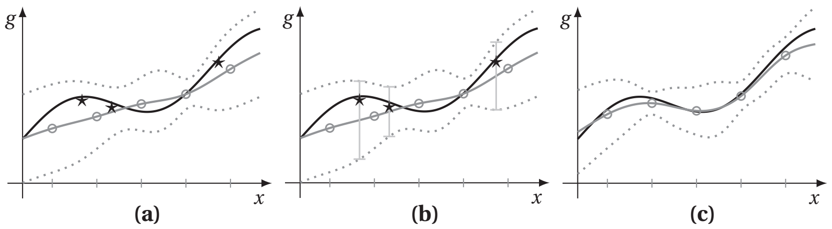


Figure 4.4: (a) The black line indicates the latent function g, while the gray solid and dotted lines represent the mean and variance of recursive GP. The circles indicate the location of the basis vectors $\mathbf{X}$ (x-axis) and their mean values $\underline{\mu}^g$ (y-axis). The stars indicate new observations. (b) Inferring the mean and covariance of g at the locations of the new observations from the current recursive GP estimate. (c) Updating the GP with the new observations gives an improved estimate of the latent function.

and time step $k = 0, 1, \ldots$, which facilitates to process streaming data. Hence, $\underline{y}_k$ and $\mathbf{X}_k$ can be considered a subset of $\underline{y}_{\mathcal{D}}$ and $\mathbf{X}_{\mathcal{D}}$, respectively, but where $\underline{y}_{\mathcal{D}}$ and $\mathbf{X}_{\mathcal{D}}$ are not known completely a priori. Also off-line processing is possible by presenting $\underline{y}_{\mathcal{D}}$ and $\mathbf{X}_{\mathcal{D}}$ in batches to the algorithm.

For all steps $k = 0, 1, \ldots$ it is assumed that the basis vectors are fixed in number and location. Since $g(\underline{x})$ is assumed to be a GP, the initial distribution $f_0(\underline{g}) = \mathcal{N}\left(\underline{g}; \underline{\mu}_0^g, \mathbf{C}_0^g\right)$ of $\underline{\mathbf{g}}$ for $k = 0$ is Gaussian with mean vector $\underline{\mu}_0^g \triangleq \mu(\mathbf{X})$ and covariance matrix $\mathbf{C}_0^g \triangleq \kappa(\mathbf{X},\mathbf{X})$ according to (4.2).

The goal is now to calculate the posterior distribution $f\left(\underline{g}\middle|\underline{y}_{1:k}\right)$ recursively by updating the prior distribution of $\underline{g}$ from the previous step $k-1$

$$f_{k-1} \triangleq f_{k-1}\left(\underline{g}\middle|\underline{y}_{1:k-1}\right) = \mathcal{N}\left(\underline{g}; \underline{\mu}_{k-1}^g, \mathbf{C}_{k-1}^g\right) \tag{4.29}$$

with the new observations $\underline{y}_k$. For this purpose, the desired posterior distribution is expanded according to

$$f_k = \int \underbrace{c_k \cdot f\left(\underline{y}_k\middle|\underline{g}, \underline{g}_k\right) \cdot f\left(\underline{g}, \underline{g}_k\middle|\underline{y}_{1:k-1}\right)}_{= f\left(\underline{g}, \underline{g}_k \middle| \underline{y}_{1:k}\right)} \mathrm{d}\underline{g}_k \tag{4.30}$$

by applying Bayes' law and by integrating out $\underline{g}_k \triangleq g(\mathbf{X}_k)$ from the joint posterior $f\left(\underline{g}, \underline{g}_k \middle| \underline{y}_{1:k}\right)$. Here, c_k is a normalization constant. Based on (4.30), calculating the posterior distribution can be performed in two steps: *Inference*, i.e., calculating the joint prior $f\left(\underline{g}, \underline{g}_k \middle| \underline{y}_{1:k-1}\right)$ given the prior f_{k-1} in (4.29). *Update*, i.e., updating the joint prior with the observations $\underline{y}_k$ and integrating out $\underline{g}_k$. The interaction between both steps is depicted in Figure 4.4.

Inference

In order to determine the joint prior $f\left(\underline{g}, \underline{g}_k \middle| \underline{y}_{1:k-1}\right)$, it is important to emphasize that the joint distribution $f\left(\underline{g}, \underline{g}_k\right)$ is Gaussian with mean and covariance

$$\underline{\mu} = \begin{bmatrix} \mu(\mathbf{X}) \\ \mu(\mathbf{X}_k) \end{bmatrix} \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} \kappa(\mathbf{X},\mathbf{X}) & \kappa(\mathbf{X},\mathbf{X}_k) \\ \kappa(\mathbf{X}_k,\mathbf{X}) & \kappa(\mathbf{X}_k,\mathbf{X}_k) \end{bmatrix} , \tag{4.31}$$

respectively. This follows from the fact that $g(.)$ is a GP and any finite representation of this GP yields a Gaussian distribution. Thus, the joint prior can be written as

$$f\left(\underline{g}, \underline{g}_k \middle| \underline{y}_{1:k-1}\right) \approx f\left(\underline{g}_k \middle| \underline{g}\right) \cdot f_{k-1} = \mathcal{N}\left(\underline{g}_k; \underline{\mu}_k^p, \mathbf{B}\right) \cdot \mathcal{N}\left(\underline{g}; \underline{\mu}_{k-1}^g, \mathbf{C}_{k-1}^g\right) , \tag{4.32}$$

with

$$\underline{\mu}_k^p = \mu(\mathbf{X}_k) + \mathbf{J}_k \cdot \left(\underline{\mu}_{k-1}^g - \mu(\mathbf{X})\right) , \tag{4.33}$$

$$\mathbf{B} = \kappa(\mathbf{X}_k,\mathbf{X}_k) - \mathbf{J}_k \cdot \kappa(\mathbf{X},\mathbf{X}_k) , \tag{4.34}$$

$$\mathbf{J}_k = \kappa(\mathbf{X}_k,\mathbf{X}) \cdot \kappa(\mathbf{X},\mathbf{X})^{-1} . \tag{4.35}$$

The first equality in (4.32) follows from assuming that $\underline{\mathbf{g}}_k$ is conditionally independent of the past observations $\underline{y}_{1:k-1}$ given $\underline{\mathbf{g}}$.[7] Hence, the conditional distribution $f\left(\underline{g}_k \middle| \underline{g}\right)$ is Gaussian and results from the joint distribution $f\left(\underline{g}, \underline{g}_k\right)$ in (4.31) by conditioning on $\underline{\mathbf{g}}$ (see (2.9)), which results in the second equality.

7 This is true if all inputs $\mathbf{X}_{1:k-1}$ of the past observations are a subset of the basis vectors $\mathbf{X}$, otherwise it is an approximation.

After some algebraic transformations, where some basic properties of Gaussian distributions and the Woodbury formula are utilized, the product in (4.32) yields the joint Gaussian $f\left(\underline{g},\underline{g}_k\middle|\underline{y}_{1:k-1}\right) = \mathcal{N}\left(\underline{q};\mathbf{Q}\right)$ of $\underline{\boldsymbol{g}}$ and $\underline{\boldsymbol{g}}_k$ with mean and covariance

$$\underline{q} \triangleq \begin{bmatrix} \underline{\mu}^g_{k-1} \\ \underline{\mu}^p_k \end{bmatrix} \quad \text{and} \quad \mathbf{Q} \triangleq \begin{bmatrix} \mathbf{C}^g_{k-1} & \mathbf{C}^g_{k-1}\mathbf{J}^{\mathrm{T}}_k \\ \mathbf{J}_k\mathbf{C}^g_{k-1} & \mathbf{C}^p_k \end{bmatrix} , \tag{4.36}$$

respectively, and with covariance $\mathbf{C}^p_k \triangleq \mathbf{B} + \mathbf{J}_k\mathbf{C}^g_{k-1}\mathbf{J}^{\mathrm{T}}_k$. For a detailed derivation see Paper J. A close inspection of the second row in (4.36) shows that it has the same structure as an RTSS (see Section 2.3) and it coincides with the augmented Kalman Smoother proposed in [148], but there no update step for basis vectors as introduced next is derived.

Update

Given the result of the previous section that the joint prior in (4.32) is a Gaussian $\mathcal{N}(\underline{q},\mathbf{Q})$, the next step is to perform the update and marginalization in (4.30). For this purpose, (4.30) is rearranged to

$$f_k = \int \underbrace{c_k \cdot f\left(\underline{y}_k\middle|\underline{g}_k\right) \cdot \overbrace{f\left(\underline{g}_k\middle|\underline{y}_{1:k-1}\right)}^{}}_{=f\left(\underline{g}_k\middle|\underline{y}_{1:k}\right)\ \text{(Kalman filter)}} \overbrace{\cdot\, f\left(\underline{g}\middle|\underline{g}_k,\underline{y}_{1:k-1}\right)}^{=f\left(\underline{g},\underline{g}_k\middle|\underline{y}_{1:k-1}\right)} \mathrm{d}\underline{g}_k \tag{4.37}$$

under consideration that $\underline{\boldsymbol{g}}$ is not observed and thus, $f\left(\underline{y}_k\middle|\underline{g}_k\right)$ is independent of $\underline{\boldsymbol{g}}$. Since $f\left(\underline{y}_k\middle|\underline{g}_k\right) = \mathcal{N}\left(\underline{y}_k;\underline{g}_k,\sigma^2\mathbf{I}\right)$ according to (4.5) and $f\left(\underline{g}_k\middle|\underline{y}_{1:k-1}\right) = \mathcal{N}\left(\underline{g}_k;\underline{\mu}^p_k,\mathbf{C}^p_k\right)$ are both Gaussian, $\underline{\boldsymbol{g}}_k$ can be updated easily via a *Kalman filter* update step. Updating $\underline{\boldsymbol{g}}$ and integrating out $\underline{g}_k$ is then performed simultaneously, which yields the desired posterior $f_k = \mathcal{N}\left(\underline{g};\underline{\mu}^g_k,\mathbf{C}^g_k\right)$ with

$$\underline{\mu}^g_k = \underline{\mu}^g_{k-1} + \mathbf{G}_k \cdot \left(\underline{y}_k - \underline{\mu}^p_k\right) , \tag{4.38}$$

$$\mathbf{C}^g_k = \mathbf{C}^g_{k-1} - \mathbf{G}_k\mathbf{J}_k\mathbf{C}^g_{k-1} , \tag{4.39}$$

$$\mathbf{G}_k = \mathbf{C}^g_{k-1}\mathbf{J}^{\mathrm{T}}_k \cdot \left(\mathbf{C}^p_k + \sigma^2\mathbf{I}\right)^{-1} . \tag{4.40}$$

Algorithm 4 Recursive Gaussian Process (RGP)

```
    ▷ Inference
 1: Calculate gain matrix $\mathbf{J}_k$ according to (4.35)
 2: Calculate mean $\underline{\mu}_k^p$ via (4.33) and covariance matrix $\mathbf{C}_k^p$ via (4.36)
    ▷ Update
 3: Calculate gain matrix $\mathbf{G}_k$ according to (4.40)
 4: Calculate mean $\underline{\mu}_k^g$ via (4.38) and covariance matrix $\mathbf{C}_k^g$ via (4.39)
```

Putting all together, at steps $k = 1, 2, \ldots$ the proposed RGP recursively processes observations $\underline{y}_k$ at the inputs $\mathbf{X}_k$ as listed in Algorithm 4. This recursion commences from the initial mean $\underline{\mu}_0^g = \mu(\mathbf{X})$ and covariance $\mathbf{C}_0^g = \kappa(\mathbf{X}, \mathbf{X})$.

Discussion

So far, it was assumed that the set of basis vectors is fixed. The inference step, however, can also be utilized for introducing new basis vectors $\mathbf{X}'$. This might be of interest in locations where the current estimate of the latent function is inaccurate. By replacing $\mathbf{X}_k$ with $[\mathbf{X}, \mathbf{X}']$, the inference step provides the initial mean and covariance as well as the cross-covariance between the new basis vectors and the old ones.

The computations of the inference step scale with $\mathcal{O}(m^2 \cdot n_k)$ due to calculating $\mathbf{J}_k$ in (4.35), where n_k is the number of observations at step k. Here, the inversion of the kernel matrix $\kappa(\mathbf{X}, \mathbf{X})$ is computationally unproblematic, as it has to be calculated only once at step $k = 0$. Once the gain matrix $\mathbf{J}_k$ is calculated, predictions for a single test input are in $\mathcal{O}(m)$ (mean) and $\mathcal{O}(m^2)$ (covariance). Assuming that all observations are processed at once, predictions of the RGP are as complex as predictions of active set GP approaches. In contrast to most sparse GP approaches, the proposed method can process new observations on-line.

The update step scales with $\mathcal{O}(n_k \cdot m^2)$, where the complexity results from matrix multiplications for which more efficient algorithms exist, e.g., Strassen's algorithm [186]. The inversion in (4.40) again is not critical as the affected matrix is of size $n_k \times n_k$, where typically $n_k \ll m$.

4.5.4 On-line Hyperparameter Learning

In the following, the previous assumption of a-priori known hyperparameters is relaxed. Instead, the goal is now to learn the hyperparameters $\underline{\theta} \in \mathbb{R}^{n_\theta}$ simultaneously with estimating the values of the latent function $g(.)$ at the basis vectors. This is achieved by formulating the learning part as a recursive parameter estimation problem, which can be performed together with the function value estimation. Similar to Section 4.5.3, this boils down to calculating a joint posterior $f_k \triangleq f\left(\underline{\xi}_k \middle| \underline{y}_{1:k}\right) = \mathcal{N}\left(\underline{\xi}_k; \underline{\mu}_k^\xi, \mathbf{C}_k^\xi\right)$, where $\underline{\boldsymbol{\xi}}_k^{\mathrm{T}} \triangleq \left[\underline{\boldsymbol{g}}^{\mathrm{T}}\ \underline{\boldsymbol{\theta}}_k^{\mathrm{T}}\right]$ is the joint hidden state with mean and covariance

$$\underline{\mu}_k^\xi \triangleq \begin{bmatrix} \underline{\mu}_k^g \\ \underline{\mu}_k^\eta \end{bmatrix} , \quad \mathbf{C}_k^\xi \triangleq \begin{bmatrix} \mathbf{C}_k^g & \mathbf{C}_k^{g\eta} \\ \mathbf{C}_k^{\eta g} & \mathbf{C}_k^\eta \end{bmatrix} .$$

Starting point for this calculation is a joint prior $f\left(\underline{\xi}_{k-1} \middle| \underline{y}_{1:k-1}\right)$ at step $k-1$, which is updated with the new observations $\underline{y}_k$. This requires the following two operations: *Inference*, i.e., calculating a joint density $f\left(\underline{\xi}_{k-1}, \underline{g}_k \middle| \underline{y}_{1:k-1}\right)$ by exploiting the results of Section 4.5.3, and *Update*, i.e., incorporation of the new observations $\underline{y}_k$ and marginalization to obtain f_k.

Inference

To incorporate the new inputs $\mathbf{X}_k$, it is necessary to infer the latent function $g(.)$ at $\mathbf{X}_k$. For this purpose, the intermediate result (4.36) is exploited. The part of the mean $\underline{q}$ and the covariance $\mathbf{Q}$ regarding $\underline{g}_k$ can alternatively be calculated by employing a Kalman predictor on the linear state-space model

$$\underline{\boldsymbol{g}}_k = \mathbf{J}_k \cdot \underline{\boldsymbol{g}} + \underline{\boldsymbol{w}}_k \quad , \quad \underline{\boldsymbol{w}}_k \sim \mathcal{N}(\underline{b}, \mathbf{B}) , \tag{4.41}$$

where $\underline{b} \triangleq m(\mathbf{X}_k) - \mathbf{J}_k \cdot m(\mathbf{X})$ and $\mathbf{B}$ is according to (4.34). In order to also correlate $\underline{\boldsymbol{g}}_k$ with the hyperparameters, the model in (4.41) is extended to a state-space model given by

$$\begin{bmatrix} \underline{\boldsymbol{\xi}}_{k-1} \\ \underline{\boldsymbol{g}}_k \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \\ \mathbf{J}_k\left(\underline{\boldsymbol{\theta}}_{k-1}\right) & \mathbf{0} \end{bmatrix} \cdot \underbrace{\begin{bmatrix} \underline{\boldsymbol{g}} \\ \underline{\boldsymbol{\theta}}_{k-1} \end{bmatrix}}_{\underline{\boldsymbol{\xi}}_{k-1}} + \underline{\boldsymbol{w}}_k , \tag{4.42}$$

where the noise $\underline{\boldsymbol{w}}_k \sim \mathcal{N}\left(\underline{\mu}_k^w, \mathbf{C}_k^w\right)$ is Gaussian with mean and covariance

$$\underline{\mu}_k^w \triangleq \begin{bmatrix} \underline{0} \\ \underline{0} \\ \underline{b}\left(\underline{\boldsymbol{\theta}}_{k-1}\right) \end{bmatrix}, \mathbf{C}_k^w \triangleq \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}\left(\underline{\boldsymbol{\theta}}_{k-1}\right) \end{bmatrix}, \tag{4.43}$$

respectively. Here, the dependence on the hyperparameters has been made explicit. The first two rows in (4.42) and (4.43) are merely an identity mapping of the given joint state $\underline{\boldsymbol{\xi}}_{k-1}$, while the last row corresponds to (4.41).

Based on model (4.42), performing a prediction would yield the desired joint distribution $f\left(\underline{\xi}_{k-1}, \underline{g}_k \middle| \underline{y}_{1:k-1}\right)$. As the model is nonlinear with respect to the hyperparameters $\underline{\boldsymbol{\theta}}_{k-1}$, the prediction cannot be performed exactly in closed form. Fortunately, the model is conditionally linear. Thus, the prediction can be approximated efficiently and accurately by means of the decomposition technique proposed in Section 2.5.2, where the nonlinear part—the hyperparameters—are sampled by means of an LRKF while the prediction of $\underline{\boldsymbol{g}}$ can be performed exactly via the Kalman predictor.

Update

In order to incorporate the new observations $\underline{y}_k$ in a very computationally efficient manner, the update is performed by means of the observed-unobserved decomposition proposed in Section 2.5.1. The observed state $\underline{\boldsymbol{\xi}}_k^o$ comprises the noise standard deviation $\boldsymbol{\sigma}$ as well as $\underline{\boldsymbol{g}}_k$, while the unobserved state $\underline{\boldsymbol{\xi}}_k^u$ comprises $\underline{\boldsymbol{g}}$ and $\underline{\boldsymbol{\theta}}_k^-$ being the vector of all hyperparameters excluding $\boldsymbol{\sigma}$.

Updating the observed state can be performed in closed form by means of reformulating the nonlinear mapping (4.1) to

$$\boldsymbol{y}_i = g(x_i) + \boldsymbol{\sigma} \cdot \boldsymbol{v} \quad , \quad \boldsymbol{v} \sim \mathcal{N}(0,1) \, , \tag{4.44}$$

with $\boldsymbol{v}$ being uncorrelated with $\boldsymbol{\sigma}$. For a deterministic noise standard deviation σ, the model (4.44) is equivalent to (4.1) since $\boldsymbol{w} = \sigma \cdot \boldsymbol{v}$ with identical mean and variance. Here, the standard deviation $\boldsymbol{\sigma}$ of the observation noise is made explicitly accessible. This simplifies the update, as the mean $\underline{\mu}_k^y$ and covariance $\mathbf{C}_k^y$ of the observations as well as the cross-covariance $\mathbf{C}_k^{oy}$ between observed state and observations can be calculated exactly in closed form as shown in Paper K.

Assuming that the observed state $\underline{\boldsymbol{\xi}}_k^o$ and the observations $\underline{\boldsymbol{y}}_k$ are jointly Gaussian distributed, updating the observed state can be performed according to (2.9), which yields the desired conditional density $f\left(\underline{\xi}_k^o \middle| \underline{y}_{1:k}\right) \approx \mathcal{N}\left(\underline{\mu}_k^e, \mathbf{C}_k^e\right)$ with mean $\underline{\mu}_k^e$ and covariance $\mathbf{C}_k^e$ according to (2.9).

By means of the updated observed state it is now possible to update the unobserved part resulting in the Gaussian density $f\left(\underline{\xi}_k^u | \underline{\xi}_k^o\right) = \mathcal{N}\left(\underline{\mu}_k^u, \mathbf{C}_k^u\right)$ with mean $\underline{\mu}_k^u$ and covariance $\mathbf{C}_k^u$ according to (2.43) and (2.44), respectively.

To finalize the update step and thus, to obtain the desired joint posterior $f_k = \mathcal{N}\left(\underline{\xi}_k; \underline{\mu}_k^\xi, \mathbf{C}_k^\xi\right)$ with updated basis vectors and hyperparameters, the above results are combined according to

$$\underline{\mu}_k^\xi = \begin{bmatrix} \underline{\mu}_k^u \\ \underline{h}^{\mathrm{T}} \cdot \underline{\mu}_k^e \end{bmatrix} \quad , \quad \mathbf{C}_k^\xi = \begin{bmatrix} \mathbf{C}_k^u & \mathbf{L}_k \mathbf{C}_k^e \underline{h} \\ \underline{h}^{\mathrm{T}} \mathbf{C}_k^e \mathbf{L}_k^{\mathrm{T}} & \underline{h}^{\mathrm{T}} \mathbf{C}_k^e \underline{h} \end{bmatrix} \tag{4.45}$$

with $\mathbf{L}_k \triangleq \mathbf{C}_k^{uo} \left(\mathbf{C}_k^o\right)^{-1}$ and $\underline{h}^{\mathrm{T}} \triangleq [1,0,0,\ldots,0]$. The first row in (4.45) corresponds to marginalizing out $\underline{\boldsymbol{g}}_k$.

Example 22: Synthetic Data

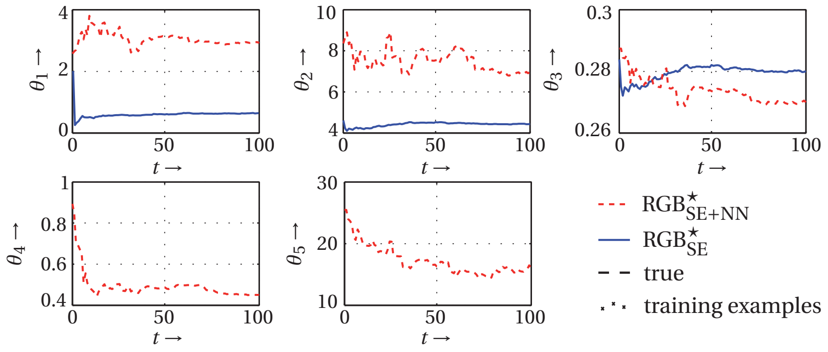
The performance of the RGP and the simultaneous regression and hyperparameter learning approach (denoted as RGP* in the following) is compared with a full GP as well as with the sparse GP methods SOGP and SPGP. For this purpose, data generated by means of two different synthetic functions are considered. The first function

$$\boldsymbol{y} = \frac{\boldsymbol{x}}{2} + \frac{25 \cdot \boldsymbol{x}}{1+\boldsymbol{x}^2} \cdot \cos(\boldsymbol{x}) + \boldsymbol{w} \quad , \quad \boldsymbol{w} \sim \mathcal{N}(0, 0.1) \tag{4.46}$$

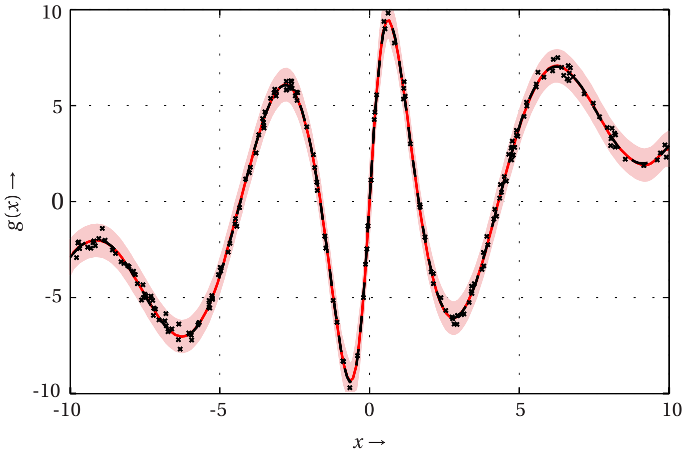
is smooth but non-stationary. It is similar to the system model in (4.23). At each step k, 40 input-observation pairs are selected randomly from the interval $[-10, 10]$. In total 100 steps are performed. The active sets (SOGP, SPGP) and basis vectors (RGP, RGP*) comprise 50 elements, which are placed equidistant on the interval $[-10, 10]$. As second function

$$\boldsymbol{y} = \mathcal{N}(0.6, 0.04) + \mathcal{N}(0.15, 0.0015) + 4 \cdot H(0.3) + \boldsymbol{w} \ , \tag{4.47}$$

is considered, where $H(.)$ is the Heaviside step function with $H(a) = 0$ if $x \leq a$ and $H(a) = 1$ if $x > a$. This function has a discontinuity at $x = 0.3$



(a) Evolution of the hyperparameters of RGP* with SE (blue, solid) and SE+NN covariance function (red, dashed), where $\theta_1 = l_1$ (length-scale), $\theta_2 = \alpha_1$ (signal standard deviation), $\theta_3 = \sigma$ (noise standard deviation) are the parameters of the SE kernel and $\theta_4 = l_2$ (length-scale), $\theta_5 = \beta$ (signal standard deviation) are the parameters of the NN kernel.



(b) True function (black, dashed line), regression result by RGP* with SE+NN covariance function together with 99% confidence area, and the training examples of step $t = 100$ (black crosses).

Figure 4.5: Exemplary regression result of proposed approach for function (4.46).

and was considered as a benchmark in [212]. The noise $\boldsymbol{w}$ has variance $\sigma^2 = 0.16$. A total of 70 steps are performed, with 50 data points per step drawn from $[-2, 2]$. On this interval, 30 active set elements and basis vectors are placed equidistant.

The mean function of all GPs is zero and as covariance function two different ones are employed: the SE function (4.8) as well as the sum of SE and NN function (see (4.10)), denoted as SE+NN in the following. The hyperparameters for the full GP are optimized via evidence maximization (4.12). These optimized hyperparameters are also used for the RGP.

In Figure 4.5a on the previous page, an exemplary regression result of RGP* with SE+NN covariance function is depicted. The true function is accurately reconstructed. As shown in Figure 4.5b, the hyperparameters are adjusted over time and converge. This leads to improved regression results compared to the other hyperparameter learning approaches SOGP and SPGP as can be seen in Table 4.3 on the next page. This holds for both covariance functions, whereas SE+NN yields better results as it is possible to capture the non-stationarity thanks to the non-stationary NN kernel. Compared to a full GP, RGP* is slightly inferior. The off-line hyperparameter optimization (4.12) provides optimal results and RGP* cannot improve further. With the optimal hyperparameters however, RGP performs close to a full GP but with significantly lower runtime.

The results in Table 4.4 indicate that off-line hyperparameter optimization (4.12) is not always optimal. Here, the hyperparameters learned by RGP* result in better estimates compared to all other algorithms with at the same time lower computational load. It is worth mentioning that RGP* is the only sparse approach that really exploits the properties of the SE+NN kernel resulting in an improved regression compared to the SE kernel.

Discussion

Directly modeling some of the hyperparameters by means of a Gaussian may not be appropriate in some cases. For instance, the length-scale hyperparameters of the SE covariance function in (4.8) have to be positive. To account for such constraints, a standard trick in GP regression is to transform the hyperparameters first and then to train the transformed parameters. After training, the inverse

Table 4.3: Average rmse, nll, and runtime for function (4.46).

	SE			SE+NN		
	rmse	nll	time in s	rmse	nll	time in s
Full GP	0.31 ± 0.02	0.25 ± 0.05	0.82	0.30 ± 0.02	0.24 ± 0.06	1.46
RGP	0.31 ± 0.02	0.26 ± 0.06	0.16	0.31 ± 0.03	0.24 ± 0.06	0.11
RGP*	0.37 ± 0.02	0.41 ± 0.14	0.65	0.35 ± 0.05	0.34 ± 0.12	1.81
SOGP	1.18 ± 0.03	7.49 ± 0.4	0.93	0.44 ± 0.03	0.80 ± 0.13	2.58
SPGP	0.54 ± 0.02	1.15 ± 0.11	13.05	0.39 ± 0.02	0.51 ± 0.09	24.40

Table 4.4: Average rmse, nll, and runtime for function (4.47).

	SE			SE+NN		
	rmse	nll	time in s	rmse	nll	time in s
Full GP	1.38 ± 0.41	1.70 ± 0.33	0.92	1.04 ± 0.43	1.41 ± 0.35	2.57
RGP	1.40 ± 0.38	1.98 ± 0.40	0.45	1.12 ± 0.35	1.86 ± 0.19	0.48
RGP*	0.98 ± 0.11	1.48 ± 0.23	0.38	0.88 ± 0.10	1.39 ± 0.15	1.14
SOGP	1.68 ± 0.07	2.89 ± 0.17	0.8	1.68 ± 0.07	2.88 ± 0.17	2.21
SPGP	1.63 ± 0.10	1.91 ± 0.06	5.6	1.65 ± 0.07	1.93 ± 0.05	10.85

transformation is applied in order to obtain the original hyperparameters. In case of positive hyperparameters, the logarithm for transforming and the exponential function as inverse transformation are common. RGP* can directly be used to also train/estimate transformed hyperparameters.

Assuming Gaussian noise $\boldsymbol{w}$ in (4.1) is not reasonable for every application. Capturing a non-Gaussian distribution by the proposed methods can for instance be achieved via warping as proposed in [177]. Alternatively, f_k could be represented by means of a Gaussian mixture allowing for the application of techniques proposed in Section 3.4.

The computation and memory costs of RGP* for a single time step k scale with $\mathcal{O}\left(s \cdot n_k \cdot (m + n_\theta)^2 + n_k^3\right)$ and with $\mathcal{O}\left((m + n_\theta)^2\right)$, respectively, where s is the number of samples of the employed LRKF, n_k is the number of observations at step k, m is the number of basis vectors, and n_θ is the dimension of $\underline{\theta}$. If at each step k the same number of observations is processed, than the computational and memory costs are constant for each step for both RGP and RGP*. Furthermore and in contrast to a full GP the computational and memory costs do not increase over time, i.e., when more and more observations become available.

4.6 Summary

Assuming that no analytical system and measurement models are available, but GP representations of these models exist, the contributions made in this chapter are concerned with performing analytic filtering and smoothing as well as on-line learning of GP models:

- *Analytic filtering and smoothing for GP models:* For particular covariance functions, Gaussian filtering and smoothing for GP system and measurement models can be performed in closed-form, without the need of sampling or linearization. Given a sufficient amount of training data, filtering and smoothing are superior compared to many other Gaussian filters operating on the analytical models.
- *Recursive GP regression:* Especially for streaming data, there is a lack of sparse GP regression approaches in the state-of-the-art. The proposed RGP allows regression with constant computational and memory demand. This approach makes no restriction on the used mean and covariance functions.

- *On-line hyperparameter learning:* RGP can be extended in such a way that on-line hyperparameter learning for streaming data is possible. Here, updating the basis vectors and hyperparameters with new data is treated as a joint Gaussian filtering problem, which leads to a computationally efficient and accurate GP regression approximation.

Generally, there are many machine learning techniques for learning system and measurement models from data. GP regression however forms a Bayesian approach of this task, with close relationship to Gaussian filtering. This relationship is the main purpose for allowing the above contributions, where it is possible to benefit from the rich theoretical and algorithmic foundation of Gaussian filtering introduced and extended in Chapter 2.

5

Applications

For every major distribution and filtering group introduced in the previous three chapters a dedicated real-world application is studied in this chapter. These applications are:

- *Range-based localization:* Estimating the position and orientation of a moving object via Gaussian filtering (summarizes Papers L and M).
- *Gas dispersion source estimation:* Determining the location and strength of a gas release into atmosphere using Gaussian mixture filtering (Paper N).
- *Active Object recognition:* Effectively utilizing a movable camera for fast object recognition based on Gaussian process regression (Paper O).

This list already indicates that Bayesian filtering in general and the proposed solutions in particular are applicable in a broad range of real-world estimation problems.

5.1 Range-based Localization

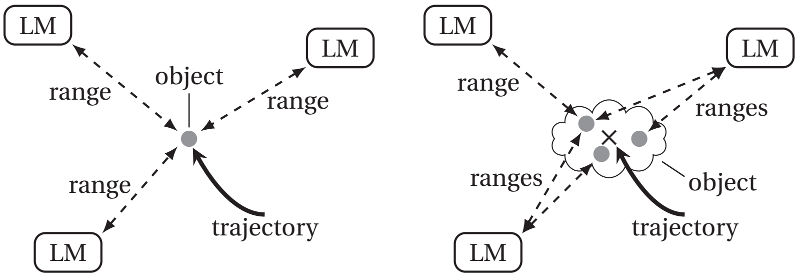
In applications such as car navigation, mobile robot navigation, or telepresence, the position of a moving object is often localized based on range/ distance measurements between the object and known landmarks. These ranges can for example be measured by times of arrival or field strengths [168].

Existing range-based localization algorithms can be divided into two classes. Approaches of the first class assume exact (or almost exact) range measurements. As long as this assumption is satisfied, closed-form localization approaches as those in [15, 34, 42, 77, 118, 195], gradient descent algorithms [152], or methods based on linearization via Taylor-series expansion [65] perform very well. However, these approaches merely allow for a *static localization*, i.e., at every time step an independent location estimation is performed. Furthermore, accurate range measurements require specialized and expensive hardware.

Dealing with inaccurate measurements that may arise for example from signal strength information or ultrasonic range finders requires range-based localization approaches from the second class. Based on probabilistic models that capture measurement uncertainties—for instance arising from measurement noise or modeling errors—the object's position and velocity can be estimated by means of a Bayesian filter in a recursive fashion. This allows for *dynamic localization*, i.e., the combination of dead reckoning and static localization, for a smoother and more robust localization. The maybe most prominent range-based localization algorithm based on Bayesian filtering is used in GPS.

Example 23: GPS

The *global positioning system* (GPS) is the most widely used satellite-based navigation system. The localization principle employed in GPS is based on *multilateration*, i.e., measuring the distance between the object's position and several reference points or *landmarks* in 3D. In GPS the necessary distances are determined by using *time of arrival*. Here, the satellites send a signal to the receiver (the object) that includes information about the exact time of its broadcast. If the time of the receiver is synchronized with the system time of the satellites, which is the same for all satellites, the receiver is now capable of calculating the duration of signal transmission. By multiplying the duration of the transmission with the speed of light the required distance is obtained. The receiver typically uses an EKF for estimating its position and velocity based on the calculated distances and the dynamics of the object [98].



(a) Position estimation. **(b)** Pose estimation.

Figure 5.1: Examples for range-based localization problems. Based on the measured ranges between landmarks (LMs) at known positions and the unknown position of the object, the object's trajectory has to be estimated. In case of pose estimation (b), the object possesses itself several landmarks (gray circles) to which range measurements can be performed. The center of mass is indicated by the cross.

When assuming that the object can be considered as a point in space, the quantities of interest are merely position, velocity and sometimes acceleration. GPS for instance makes this assumption. In applications like telepresence however, where the extent of the object is important, also the orientation and corresponding angular velocities in 3D have to be estimated. For both problems—position *and* pose estimation—dynamic range-based localization algorithms utilizing Gaussian filtering are introduced in the following.

5.1.1 Position Estimation

At first it is assumed that the object can be considered a point as depicted in Figure 5.1a. In this case, the state $\underline{\boldsymbol{x}}^{\mathrm{T}} \triangleq \left[\underline{\boldsymbol{t}}^{\mathrm{T}}\ \underline{\dot{\boldsymbol{t}}}^{\mathrm{T}}\right]$ of the object consists of its position $\underline{\boldsymbol{t}} = [\boldsymbol{x}\ \boldsymbol{y}\ \boldsymbol{z}]^{\mathrm{T}}$ and velocity $\underline{\dot{\boldsymbol{t}}} = [\dot{\boldsymbol{x}}\ \dot{\boldsymbol{y}}\ \dot{\boldsymbol{z}}]^{\mathrm{T}}$.

Dynamic and Measurement Model

The dynamic behavior—the motion—of the object is described by means of the linear discrete-time dynamic system

$$\underline{\boldsymbol{x}}_{k+1} = \mathbf{A} \cdot \underline{\boldsymbol{x}}_k + \underline{\boldsymbol{w}}_k\ , \tag{5.1}$$

where the noise $\underline{\boldsymbol{w}}_k$ is assumed to be zero-mean white Gaussian. For a *position velocity model* [206], the matrix $\mathbf{A}$ and the covariance of the process noise $\mathbf{C}^w$ are given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & T \cdot \mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} , \quad \mathbf{C}^w = \begin{bmatrix} \frac{T^3}{3}\mathbf{C}_c^w & \frac{T^2}{2}\mathbf{C}_c^w \\ \frac{T^2}{2}\mathbf{C}_c^w & T \cdot \mathbf{C}_c^w \end{bmatrix} , \tag{5.2}$$

respectively, where T is the sampling time. $\mathbf{C}_c^w \triangleq \operatorname{diag}\left(\begin{bmatrix}\sigma_{c,x}^2 & \sigma_{c,y}^2 & \sigma_{c,z}^2\end{bmatrix}\right)$ corresponds to the process noise covariance of the continuous time system model with $\sigma_{c,\xi}^2$ being the variances of dimension $\xi \in \{x, y, z\}$.

Range measurements to N landmarks at the known positions $\underline{S}_i \in \mathbb{R}^3$ with $i = 1, \ldots, N$ are incorporated. The nonlinear relation between the object position and the landmark position is given by

$$\boldsymbol{\rho}_{k,i} = \left\| \underline{S}_i - \underline{\boldsymbol{t}}_k \right\|_2 , \tag{5.3}$$

where $\boldsymbol{\rho}_{k,i}$ is the Euclidean distance between object and landmark.

In a real scenario, the ranges cannot be measured exactly, i.e., measurement uncertainty has to be considered, which is usually done by incorporating a noise process into (5.3). Two possibilities arise for incorporation. In the first case, which is the *standard model*

$$\boldsymbol{\rho}_{k,i} = \left\| \underline{S}_i - \underline{\boldsymbol{t}}_k \right\|_2 + \underline{\boldsymbol{v}}_{k,i} , \tag{5.4}$$

the noise process $\boldsymbol{v}_{k,i}$ directly affects the range $\boldsymbol{r}_{k,i}$. In the second case

$$\boldsymbol{\rho}_{k,i} = \left\| \underline{S}_i - \underline{\boldsymbol{t}}_k - \underline{\boldsymbol{v}}_{k,i} \right\|_2 , \tag{5.5}$$

which is called *noise before non-linearity* [43], the noise process affects the difference between object and landmark position. This measurement model can be interpreted such that the positions of the landmarks are uncertain. In both measurement models, the noise process is assumed to be zero-mean white Gaussian. In the following, the second model (5.5) is considered for mainly two reasons: First, the standard model (5.4) is only appropriate in situations where the distance $\boldsymbol{\rho}_{k,i}$ is large compared to the variance of the noise $\boldsymbol{v}_{k,i}$. Otherwise, negative ranges are possible, which is not true in reality. This problem cannot occur in the second measurement model. Second, the model in (5.5) allows analytic moment matching.

Analytic Moment Matching

Thanks to the linearity of the dynamic model (5.2), the standard Kalman filter prediction step (2.12) can be employed for propagating the state from time step k to time step $k+1$. To obtain also analytic expressions for the moment integrals (2.8) in case of the measurement update, the measurement equation (5.5) is squared, which yields

$$\boldsymbol{d}_i \triangleq (\boldsymbol{\rho}_i)^2 = (\underline{S}_i - \underline{\boldsymbol{t}})^{\mathrm{T}} \cdot (\underline{S}_i - \underline{\boldsymbol{t}}) - 2 \cdot (\underline{S}_i - \underline{\boldsymbol{t}})^{\mathrm{T}} \cdot \underline{\boldsymbol{v}}_i + \underline{\boldsymbol{v}}_i^{\mathrm{T}} \cdot \underline{\boldsymbol{v}}_i \, , \tag{5.6}$$

where $\boldsymbol{d}_i$ is a squared range assumed to be calculated by $\hat{d}_i = \hat{\rho}_i^2$. Thus, the modified measurement equation (5.6) can be described in short term via

$$\boldsymbol{d}_i = h_i(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}_i) \tag{5.7}$$

for a single measurement to landmark i and via

$$\underline{\boldsymbol{d}} = \underline{h}(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}) \tag{5.8}$$

for measurements to all landmarks, where d_i and $h_i(.,.)$ are the ith element of $\underline{\boldsymbol{d}}$ and $\underline{h}(.,.)$, respectively. Hence, the vector of squared ranges $\hat{\underline{d}}$ is calculated according to $\hat{\underline{d}} = \hat{\underline{r}} \odot \hat{\underline{r}}$. The measurement noise $\underline{\boldsymbol{v}}$ in (5.8) is zero-mean with covariance matrix

$$\mathbf{C}^v = \begin{bmatrix} \mathbf{C}_1^v & \cdots & \mathbf{C}_{1,j}^v & \cdots & \mathbf{C}_{1,N}^v \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{C}_{i,1}^v & \cdots & \mathbf{C}_{i,j}^v & \cdots & \mathbf{C}_{i,N}^v \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{C}_{N,1}^v & \cdots & \mathbf{C}_{N,j}^v & \cdots & \mathbf{C}_N^v \end{bmatrix} ,$$

where $\mathbf{C}_{i,j}^v$ is the 3×3 covariance matrix between the ith and jth landmark. By assuming correlations between landmarks, an algorithm valid for many real-world application can be derived. The case of uncorrelated landmarks is a special case of the algorithm.

Due to the consideration of squared ranges, the measurement model in (5.6) is a polynomial of order two allowing closed-form calculations of the moment integrals (2.8). Hence, according to (2.9) the mean vector and covariance matrix of the posterior state estimate $\underline{\boldsymbol{x}}_k \sim \mathcal{N}\left(\underline{\mu}_k^e, \mathbf{C}_k^e\right)$ are calculated via

$$\begin{aligned} \underline{\mu}_k^e &= \underline{\mu}_k^p + \mathbf{C}_k^{xd} \cdot \left(\mathbf{C}_k^d\right)^{-1} \cdot \left(\underline{\hat{d}}_k - \underline{\mu}_k^d\right) , \\ \mathbf{C}_k^e &= \mathbf{C}_k^p - \mathbf{C}_k^{xd} \cdot \left(\mathbf{C}_k^d\right)^{-1} \cdot \left(\mathbf{C}_k^{xd}\right)^{\mathrm{T}} , \end{aligned} \tag{5.9}$$

respectively, where $\underline{\mu}_k^d$ (squared measurement mean), $\mathbf{C}_k^d$ (squared measurement covariance matrix), and $\mathbf{C}_k^{xd}$ (cross-covariance between state and squared measurement) are given by[1]

$$\begin{aligned} \underline{\mu}_k^d &= (\mathbf{V} \odot \mathbf{V})^{\mathrm{T}} \cdot \underline{1}_M + \underline{1}_N \cdot \mathrm{Tr}\left(\mathbf{P} \cdot \mathbf{C}_k^p \cdot \mathbf{P}^{\mathrm{T}}\right) + \mathbf{O}^{\mathrm{T}} \cdot \mathrm{diag}\left(\mathbf{C}^v\right) , \\ \mathbf{C}_k^d &= \mathbf{O}^{\mathrm{T}} \cdot \left(4 \cdot \left(\mathrm{vec}(\mathbf{V}) \cdot \mathrm{vec}(\mathbf{V})^{\mathrm{T}}\right) \odot \mathbf{T} + 2 \cdot \mathbf{T} \odot \mathbf{T}\right) \cdot \mathbf{O} , \\ \mathbf{C}_k^{xd} &= -2 \cdot \mathbf{C}_k^p \cdot \mathbf{P}^{\mathrm{T}} \cdot \mathbf{V} , \end{aligned} \tag{5.10}$$

respectively, with

$$\begin{aligned} \mathbf{S} &\triangleq \begin{bmatrix} \underline{S}_1 & \dots & \underline{S}_N \end{bmatrix} , \\ \mathbf{P} &\triangleq \begin{bmatrix} \mathbf{I}_M & \mathbf{0}_M \end{bmatrix} , \\ \mathbf{V} &\triangleq \mathbf{S} - \left(\underline{1}_N\right)^{\mathrm{T}} \otimes \left(\mathbf{P} \cdot \underline{\mu}_k^p\right) , \\ \mathbf{O} &\triangleq \mathbf{I}_N \otimes \underline{1}_M , \\ \mathbf{T} &\triangleq \mathbf{C}^v + \mathbf{1}_N \otimes \left(\mathbf{P} \cdot \mathbf{C}^p \cdot \mathbf{P}^{\mathrm{T}}\right) , \end{aligned}$$

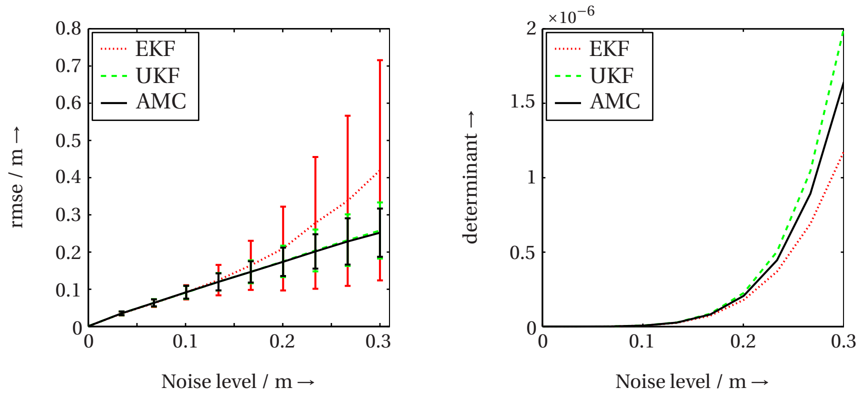
where vec(**V**) is the vectorized version of the matrix **V**, $\underline{1}_N$ is a vector of ones of dimension N, and $\mathbf{1}_N$ is a one matrix. The variable $M = 3$ stands for the three-dimensional space.

Example 24: Four Landmarks

The proposed analytical moment calculation (AMC) is compared against the EKF and UKF. For this purpose four landmarks with positions

$$\mathbf{S} = \begin{bmatrix} \underline{S}_1 & \dots & \underline{S}_4 \end{bmatrix} = \begin{bmatrix} -2 & -2 & 2 & 2 \\ -2 & 2 & -2 & 2 \\ 0 & 0 & 0 & 2 \end{bmatrix} \mathrm{m}$$

1 A detailed derivation can be found in Paper L.



(a) The average rmse and its standard deviation.

(b) Mean of the determinant of the position covariance $\mathbf{C}^e_{k,t}$.

Figure 5.2: Result of the three estimators AMC, UKF, and EKF for different noise levels.

are considered. The measurement noise covariance matrix of each landmark i is assumed to be $\mathbf{C}^v_i = \mathbf{I} \cdot \sigma^2_n$ with $\sigma_n = {}^{(n-1)}/_3\,\mathrm{m}$ where $n = 1 \ldots 10$, i.e., ten different noise levels are investigated. For each noise level 1000 MC runs are simulated, where each run consists of 100 measurement steps.

The initial state at time step $k = 0$ has zero mean and covariance $\mathbf{C}_0 = 10 \cdot \mathbf{I}_6$. The sampling time is $T = 0.1\mathrm{s}$. The process noise covariance matrix comprises the elements $\sigma^2_{c,x} = \sigma^2_{c,y} = 0.01$, and $\sigma^2_{c,z} = 0.0001$.

In Figure 5.2a, the average rmse is depicted. For small noise, all three filters perform similar. If the noise increases, the rmse of the EKF increases much stronger compared to the other two filters. For a high noise level, the UKF and the proposed approach present comparable results, where the average rmses and the standard deviations of the AMC are slightly smaller.

The average determinant of the covariance matrix of the position estimate $\underline{t}_k$ of all test runs is shown in Figure 5.2b. Due to the linearization based on first-order Taylor-series expansions, the determinant of the EKF is too small and thus the EKF is too certain about its estimate. Hence, the estimation results are inconsistent, which is often a problem when using an EKF. On the other hand, LRKFs or analytic approaches as the AMC overcome this problem. The determinant of the AMC is smaller compared to the determinant of the UKF. Furthermore, as described before, the rmse of the AMC is smaller as well. All together, the AMC is more informative compared to the UKF.

The computational complexity of the above closed-form measurement update is in $\mathcal{O}(M^3 + M^2 \cdot N)$ for the mean $\underline{\mu}_k^d$, in $\mathcal{O}(M^2 \cdot N^3)$ for the covariance $\mathbf{C}_k^d$, and $\mathcal{O}(M^2 \cdot N)$ for the cross-covariance $\mathbf{C}_k^{xd}$, where typically $M \ll N$. For calculating the required moments in (5.10), only vector-matrix products and no additional matrix inversions or roots are required. For comparison, already the computational complexity of calculating the matrix square root required for an LRKF is in $\mathcal{O}(N^3 \cdot M^3)$.

5.1.2 Position and Orientation Estimation

In the following, the previous localization problem is generalized in order to allow the estimation of the pose—position and orientation—of an extended object in 3D. Hence, the object state is given by

$$\underline{\boldsymbol{x}} \triangleq \begin{bmatrix} \underline{\boldsymbol{t}}^{\mathrm{T}} & \underline{\dot{\boldsymbol{t}}}^{\mathrm{T}} & \underline{\boldsymbol{r}}^{\mathrm{T}} & \underline{\boldsymbol{\omega}}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$$

where $\underline{\boldsymbol{t}}$ and $\underline{\dot{\boldsymbol{t}}}$ are the position and velocity in 3D, respectively, and $\underline{\boldsymbol{\omega}}$ is the angular velocity in 3D . The orientation vector $\underline{\boldsymbol{r}} \in \mathbb{R}^3$ is a so-called *rotation vector* with norm $\|\underline{r}\| = \pi$. Alternatives for describing the orientation are quaternions [107] or Euler angles [129]. Quaternions however are not minimal as they consist of four elements and the unit quaternions constraint can lead to inaccurate state estimates. Euler angles suffer from singularities and are not intuitive in usage. Rotation vectors instead are a minimal state representation. A further advantage of rotation vectors is that the dynamic behavior can be described by means of a nonlinear differential equation [22].

Dynamic Model

As the state now also comprises orientation related quantities, the dynamic model consists of two separate motion models: one for the translation and the second for the rotation. The translation model coincides with (5.1) and (5.2), respectively. The temporal evolution of the rotation vector $\underline{\boldsymbol{r}}_k$ is described by means of a nonlinear equation [17, 22]

$$\underline{\boldsymbol{r}}_{k+1} = \underline{\boldsymbol{r}}_k + \underbrace{T \cdot \left(\mathbf{I} + 0.5 \cdot \mathbf{R}(\underline{\boldsymbol{r}}_k) + a(\underline{\boldsymbol{r}}_k) \cdot \mathbf{R}(\underline{\boldsymbol{r}}_k) \cdot \mathbf{R}(\underline{\boldsymbol{r}}_k)\right)}_{\triangleq \Lambda(\underline{\boldsymbol{r}}_k)} \cdot \underline{\boldsymbol{\omega}}_k \,, \tag{5.11}$$

with

$$a(\underline{\boldsymbol{r}}_k) \triangleq \tfrac{1 - 0.5 \cdot \|\underline{\boldsymbol{r}}_k\|}{\|\underline{\boldsymbol{r}}_k\|^2} \cdot \cot\left(\tfrac{\|\underline{\boldsymbol{r}}_k\|}{2}\right) .$$

The system model for the angular velocity $\underline{\boldsymbol{\omega}}_k$ is assumed to be

$$\underline{\boldsymbol{\omega}}_{k+1} = \underline{\boldsymbol{\omega}}_k + \underline{\boldsymbol{w}}_k^{\omega} \,, \tag{5.12}$$

where $\underline{\boldsymbol{w}}_k^{w}$ is the process noise that affects the angular velocity. The process noise has zero mean and is Gaussian distributed with covariance $\mathbf{C}^{\omega}$.

By combining (5.2), (5.11) and (5.12), the system model for the pose estimation scenario can be written as

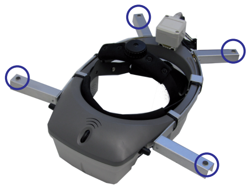
$$\underline{\boldsymbol{x}}_{k+1} = \begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \Lambda(\underline{\boldsymbol{r}}_k) \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \underline{\boldsymbol{x}}_k + \underline{\boldsymbol{w}}_k \,, \tag{5.13}$$

where the covariance of the process noise $\underline{\boldsymbol{w}}_k$ comprises the covariances of the process noises from the translation model $\mathbf{C}^t$ and the rotation model $\mathbf{C}^{\omega}$ according to

$$\mathbf{C}^w = \begin{bmatrix} \mathbf{C}^t & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}^{\omega} \end{bmatrix} .$$

Measurement Model

As depicted in Figure 5.1b, in range-based pose estimation the measured ranges depend on known landmarks located on the extended object and on known landmarks in a global coordinate system. Thus, the ranges depend on both the unknown translation *and* rotation of the object with respect to the global coordinate system.


(a) Sensors at a head-mounted display.

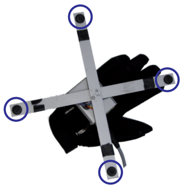

(b) Sensors mounted at a glove.

Figure 5.3: Deployment of sensors (marked by blue circles) for tracking the pose of a user or of body parts in telepresence applications. Images taken from [110].

Example 25: Telepresence

An example application where knowing the pose is of interest is *large-scale telepresence* [152]. Here, the pose of a human has to be tracked for steering a robot. For tracking the pose of the user or the pose of several body parts, several emitters are located at known positions in a global coordinate system. They are emitting signals that are received by several sensors attached to the user, e.g., at a head-mounted display (see Figure 5.3a) or at gloves (see Figure 5.3b). Based on the emitted and received signals, ranges between emitters and sensors can be determined.

The relationship between measured ranges, translation, and rotation is given by

$$\boldsymbol{\rho}_{k,ij} = \left\| \underline{L}_j - \mathbf{D}(\underline{\boldsymbol{r}}_k) \cdot \underline{M}_i - \underline{\boldsymbol{t}}_k - \underline{\boldsymbol{v}}_{k,ij} \right\|_2 , \tag{5.14}$$

which resembles the noise before non-linearity range measurement model as in (5.5). Here, $\underline{L}_j$ is the position of the jth landmark with respect to the global coordinate system and $\underline{M}_i$ is the position of the ith landmark with respect to the object coordinate system. $\underline{\boldsymbol{v}}_{k,ij}$ is the measurement noise between landmark $\underline{L}_j$ and landmark $\underline{M}_i$. The term $\boldsymbol{\rho}_{k,ij}$ is the measured range between these two landmarks, while $\underline{\boldsymbol{d}}_k$ comprises all possible measurements between global and object landmarks. The term $\mathbf{D}(\cdot)$ is the rotation matrix parametrized by the rotation vector $\underline{\boldsymbol{r}}_k$

$$\mathbf{D}(\underline{\boldsymbol{r}}_k) = \mathbf{I} + \frac{\sin(\|\underline{\boldsymbol{r}}_k\|)}{\|\underline{\boldsymbol{r}}_k\|} \cdot \mathbf{R}(\underline{\boldsymbol{r}}_k) + \frac{1-\cos(\|\underline{\boldsymbol{r}}_k\|)}{\|\underline{\boldsymbol{r}}_k\|^2} \cdot \mathbf{R}(\underline{\boldsymbol{r}}_k) \cdot \mathbf{R}(\underline{\boldsymbol{r}}_k) \,,$$

known as *Rodrigues formula*, with

$$\mathbf{R}(\underline{\boldsymbol{r}}_k) = \begin{bmatrix} 0 & -\boldsymbol{r}_z & \boldsymbol{r}_y \\ \boldsymbol{r}_z & 0 & -\boldsymbol{r}_x \\ -\boldsymbol{r}_y & \boldsymbol{r}_x & 0 \end{bmatrix}$$

being a skew-symmetric matrix.

Semi-Analytic Moment Matching

The system model (5.13) is conditionally linear, i.e., if the rotation vector is set to a fixed value, the system model becomes linear and the prediction for each value can be performed by using the well-known Kalman predictor equation.

To facilitate an analytical solution of the measurement update in closed from, the measurement equation (5.14) is squared as in Section 5.1.1, which yields

$$\underline{\boldsymbol{d}}_{k,ij} \triangleq \left(\boldsymbol{\rho}_{k,ij}\right)^2 = \left(\underline{g}_{ij}(\underline{\boldsymbol{r}}_k) - \underline{\boldsymbol{t}}_k - \underline{\boldsymbol{v}}_{k,ij}\right)^{\mathrm{T}} \cdot \left(\underline{g}_{ij}(\underline{\boldsymbol{r}}_k) - \underline{\boldsymbol{t}}_k - \underline{\boldsymbol{v}}_{k,ij}\right) \,, \qquad (5.15)$$

with

$$\underline{g}_{ij}(\underline{\boldsymbol{r}}_k) \triangleq \underline{L}_j - \mathbf{D}(\underline{\boldsymbol{r}}_k) \cdot \underline{M}_i \,. \qquad (5.16)$$

But in contrast to the previous section, squaring the measurement model alone is not sufficient due to the nonlinear term (5.16). By conditioning on $\underline{\boldsymbol{r}}$, however, (5.16) becomes affine and the entire measurement model becomes quadratic. Thus, the measurement model is conditionally integrable and the nonlinear-nonlinear decomposition proposed in Section 2.5.2 can be applied. The analytically integrable system state comprises $\underline{\boldsymbol{x}}_a^{\mathrm{T}} \triangleq \left[\underline{\boldsymbol{t}}^{\mathrm{T}} \quad \underline{\dot{\boldsymbol{t}}}^{\mathrm{T}} \quad \underline{\boldsymbol{w}}^{\mathrm{T}}\right]$, while the sampled state is $\underline{\boldsymbol{x}}_s \triangleq \underline{\boldsymbol{r}}$. For the latter, sampling via LRKFs can be employed. In doing so, for every fixed sample value of the rotation vector, the closed-form solutions in (5.10) for the moment integrals can be used with some minor modifications: the state dimension is now nine instead of six and the number of measured ranges is significantly higher as pair-wise measurements between object landmarks and global landmarks are incorporated.

Example 26: Two-dimensional Localization

A two-dimensional coordinate system is considered containing four sensors (global landmarks) and four emitters (object landmarks) with positions

$$\begin{bmatrix} \underline{M}_1^{\mathrm{T}} \\ \underline{M}_2^{\mathrm{T}} \\ \underline{M}_3^{\mathrm{T}} \\ \underline{M}_4^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} -0.2 & -0.2 \\ -0.2 & 0.2 \\ 0.2 & -0.2 \\ 0.2 & 0.2 \end{bmatrix} \mathrm{m}\,, \quad \begin{bmatrix} \underline{L}_1^{\mathrm{T}} \\ \underline{L}_2^{\mathrm{T}} \\ \underline{L}_3^{\mathrm{T}} \\ \underline{L}_4^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} -2 & -2 \\ -2 & 2 \\ 2 & -2 \\ 2 & 2 \end{bmatrix} \mathrm{m}$$

with respect to the object coordinate system and the global coordinate system, respectively. At different noise levels ranging from $[0.000001,\ldots,0.3]$ m, 1000 random trajectories are generated, where the sampling time was $T = 0.1\,\mathrm{s}$. The noise process is assumed as isotropic.

The proposed SAGF is compared to the UKF. For the UKF a decomposition into directly observed states $\underline{\boldsymbol{t}}$, $\underline{\boldsymbol{r}}$ and indirectly observed states $\underline{\dot{\boldsymbol{t}}}$, $\underline{\boldsymbol{\omega}}$ as proposed in Section 2.5.1 is used. Furthermore, due to the fact that the measurement noise is mapped through the nonlinear transformation, it has to be approximated with samples as well. In total 71 sample points are required for the UKF. On the other hand, the proposed approach only has to approximate the rotation by sample points.

For the 2D case, the system equation (5.13) becomes linear and thus, the Kalman filter prediction can be using directly. The entries of the continuous process noise covariance are set to be $\mathbf{C}_c^t = \mathrm{diag}([0.1\ 0.1])$ and $C_c^{\omega} = 0.1$. The initial state has zero mean and its covariance matrix comprises $\mathbf{C}_0^t = \mathrm{diag}([10\ 10])$, $\mathbf{C}_0^{\dot{t}} = \mathrm{diag}([10\ 10])$, $\sigma_{r,0}^2 = 0.001$, and $\sigma_{\omega,0}^2 = 0.0001$.

The estimation performance in terms of the rmse of both estimators is almost identical. Regarding the computational effort, the proposed approach only has to determine sample points for one dimension, which can be implemented very efficiently. On the other hand, the UKF calculates a matrix square root of the covariance of the noise and the directly observed state. This operations is computationally involved considering the size of the combined covariance matrix, which is 35×35. In the simulation, the proposed approach is *three times faster* than the standard approach.

5.2 Gas Dispersion Source Estimation

If a hazardous gas has been released—either accidentally or deliberately—into atmosphere, it is of paramount importance to gain knowledge of this event at an early stage in order to increase the effectiveness of counter measures for protecting the public and for mitigating the harmful effects. By means of so-called *atmospheric dispersion models* (ADMs), it is possible to predict the concentration spread of the released gas in space and time. These models, however, merely provide reliable predictions, if the characteristics of the gas source are known precisely. To determine or estimate the source characteristics it necessary to solve an *inverse problem*, where one has to infer the location and strength of the gas release from concentration measurements of spatially distributed sensors.

In general, solution methods of the source estimation problem can be classified into *forward* and *backward* methods [142]. Forward methods employ an forward-running ADM multiple times in order to find an estimate of the source that best describes the given concentration measurements. Here, the mostly used techniques are based on Bayesian inference in combination with Monte Carlo sampling. Sequential Monte Carlo methods as described in [178, 216] employ a set of samples or particles that forms the posterior probability distribution of the source parameters. This distribution is updated by means of Bayes' rule whenever new concentration measurements from sensors are available. In contrast to this online procedure, Markov chain Monte Carlo (MCMC) methods process all acquired concentration measurements in a batch in order to determine the posterior distribution. For this purpose, samples are drawn from the posterior distribution by simulating a Markov chain that has the desired posterior distribution as its stationary distribution. Given a properly constructed Markov chain it can be shown that MCMC reaches the stationary distribution after a typically large number of sampling steps, which is known as the burn-in phase. Application of MCMC to source estimation can be found for instance in [23, 79, 169].

Backward methods instead perform only one model run in the reverse direction from the sensors to the source. Commonly used techniques are backtracking, where an inverse version of an ADM is utilized (see e.g. [81]), and variational methods, where a cost function between model predictions and concentration measurements is optimized (see e.g. [154, 184]). The backward approach is preferred over forward methods, when the number of sources is larger than the number of sensors [142].

Most of the above state-of-the-art methods allow merely an off-line batch source estimation. The AGMF proposed in Section 3.4.2, however, facilitates an on-line estimation, where concentration measurements are processed continually. In doing so, timely information about the source location and strength can be provided allowing fast responses. For the AGMF, the so-called *Gaussian plume dispersion model* is utilized as a forward model, which facilitates predicting the gas dispersion in closed-form with low computational overhead.

5.2.1 Atmospheric Dispersion Models

In the following, $c(\underline{x}, t)$ is the concentration of the substance at position $\underline{x} = [x\ y\ z]^{\mathrm{T}} \in \mathbb{R}^3$ and at time $t \geq 0$. The concentration follows the *advection-diffusion equation*

$$\frac{\partial c(\underline{x}, t)}{\partial t} = \nabla \cdot \left(\mathbf{D} \cdot \nabla c(\underline{x}, t) - \underline{v} \cdot c(\underline{x}, t)\right) + s(\underline{x}, t) \tag{5.17}$$

with $\nabla \triangleq [\partial/\partial x\ \partial/\partial y\ \partial/\partial z]^{\mathrm{T}}$ (see e.g. [80]). The term $\mathbf{D} \cdot \nabla c(\underline{x}, t)$ describes the diffusion according to Fick's law with diffusion matrix $\mathbf{D}(\underline{x}, t)$ and the term $\underline{v} \cdot c(\underline{x}, t)$ represents linear advection due to wind with velocity $\underline{v}(\underline{x}, t)$. Finally, $s(\underline{x}, t)$ is a source or sink term.

Analytical solutions of (5.17), i.e., functions $c(\underline{x}, t)$ satisfying the equation, exist merely for some special cases. One such special case employed for source estimation in the following is the Gaussian plume dispersion model. In order to obtain a closed-form solution, the Gaussian plume model requires several assumptions:

1. The substance is emitted at a constant *rate* $q > 0$ from a single point source at location $\underline{x}_s \triangleq [0\ 0\ z_s]^{\mathrm{T}}$. Thus, the source term $s(\underline{x}, t)$ in (5.17) becomes
$$s(\underline{x}, t) = q \cdot \delta(x) \cdot \delta(y) \cdot \delta(z - z_s)\ .$$
2. The wind is constant with velocity $v \geq 0$ and the wind direction is along the x-axis. Hence, the velocity in (5.17) becomes $\underline{v} = [v\ 0\ 0]^{\mathrm{T}}$.
3. The diffusion is a function of the downwind distance (positive x-axis) only. Furthermore, it is assumed that the advection dominates the diffusion in

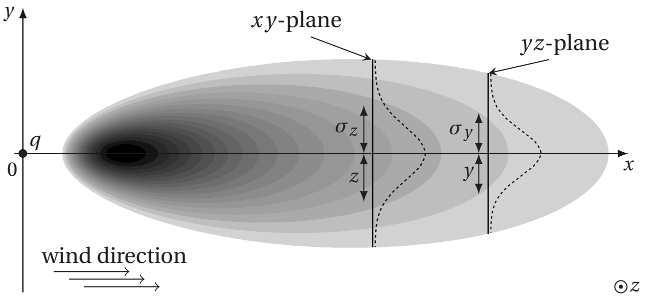


Figure 5.4: Ground level concentrations according to the Gaussian plume model, where black indicates the highest concentration level. The dotted lines indicate the profile of the plume when cutting through the plume in parallel with the xy-plane and yz-plane.

wind direction. Thus, the diffusion along the x-axis can be neglected and $\mathbf{D} = \operatorname{diag}\left(\left[0\ K_y(x)\ K_y(x)\right]\right)$ with eddy diffusion coefficients K_y, K_z.

4. The terrain is flat and the ground cannot be penetrated by the substance.
5. The solution is steady state, i.e., time independent.

Based on these assumptions and additional boundary conditions that force vanishing concentrations at infinite distance from the source and at upwind distances, (5.17) has the time-invariant solution

$$c(\underline{x}) = \frac{q}{2\pi \cdot v \cdot \sigma_y \sigma_z} \cdot \exp\left(-\frac{y^2}{2\sigma_y^2}\right) \cdot \left[\exp\left(-\frac{(z-z_s)^2}{2\sigma_z^2}\right) + \exp\left(-\frac{(z+z_s)^2}{2\sigma_z^2}\right)\right], \tag{5.18}$$

which is the well-known *Gaussian plume dispersion model* (for a detailed derivation see [184]). Here, σ_y and σ_z are the so-called standard deviations of the Gaussian concentration distribution. They depend on the stability of the atmosphere and are both functions of x. They can be obtained via integrating K_y and K_z in downwind direction or—more practically—can be determined via the Pasquill-Gifford stability classification scheme [37, 137].

Example 27: Gaussian Gas Plume

Consider a source emitting a gas contaminant from a height of $z_s = 4\,\text{m}$. The wind velocity is $v = 2\,\text{m/s}$. The atmospheric stability is of class D, which is "neutral" according to the Pasquill-Gifford classification. The corresponding ground level concentration is depicted in Figure 5.4. The gas plume spreads along the x-axis (downwind) and its shape is that of a Gaussian in planes normal to the x-axis.

The Gaussian plume model (5.18) is employed as it is widely used and suitable for describing short range substance releases. Furthermore, being an analytical model, it allows for an on-line and computationally light-weight estimation of the unknown parameters. It can be extended for arbitrary wind directions ϕ and arbitrary source location $\underline{x}_s = [x_s\ y_s\ z_s]^{\mathrm{T}}$ by means of straightforward translation and coordinate rotation, which yields the model

$$\begin{aligned} c(\underline{x}) = \tfrac{q}{2\pi \cdot v \cdot \sigma_y \sigma_z} \cdot \exp\Big(-\tfrac{(1+2\sin(\phi)\cos(\phi))\cdot(y-y_s)^2}{2\sigma_y^2}\Big) \cdot \\ \Big[\exp\Big(-\tfrac{(z-z_s)^2}{2\sigma_z^2}\Big) + \exp\Big(-\tfrac{(z+z_s)^2}{2\sigma_z^2}\Big)\Big] , \end{aligned} \tag{5.19}$$

where σ_y and σ_z are now functions of x_s, x, y, and ϕ. This model is employed in the following.

5.2.2 Parameter Estimation

Estimating the source rate q and source location $\underline{x}_s$ is a so-called *parameter estimation* problem. That is, the quantities of interest are time-invariant; there is no dynamical model driving the parameters over time. Instead, only data in form of concentration measurements from a set of spatially distributed sensors is available to determine the parameters. It is assumed that the measurements become available sequentially over time, i.e., batch or off-line estimation is impractical. Additional parameters like wind speed or direction are assumed to be known, as they can be provided reliably from external sources like weather stations.

In order to apply the AGMF, an appropriate measurement model is required that relates the concentration measurements to the unknown source parameters $\underline{\boldsymbol{\theta}}^{\mathrm{T}} \triangleq [\underline{\boldsymbol{x}}_s^{\mathrm{T}}\ \boldsymbol{q}]$. The Gaussian plume model reflects this relations. Thus, suppose that the measurement $\hat{z}_k$ is acquired by a sensor at location $\underline{x}_r \triangleq \left[x_r \; y_r \; z_r\right]^\mathrm{T}$ at time step k, the resulting measurement models is given by

$$\hat{z}_k = c\left(\underline{x}_r; \underline{\boldsymbol{\theta}}\right) + \boldsymbol{v}_k \,, \quad \boldsymbol{v}_k \sim \mathcal{N}\left(0, \sigma_v^2\right) \,, \tag{5.20}$$

where $c\left(\underline{x}_r; \underline{\boldsymbol{\theta}}\right)$ is the true concentration value according to (5.19) and $\boldsymbol{v}_k$ is the sensor's noise, which is assumed to be independent in time and space. It is worth mentioning that in case of multiple sources, the concentration measurement can be written as

$$\hat{z}_k = \sum_{i=1}^{L} c_i\left(\underline{x}_r; \underline{\boldsymbol{\theta}}_i\right) + \boldsymbol{v}_k \,, \tag{5.21}$$

where the superposition of the concentration values $c_i\left(\underline{x}_r; \underline{\boldsymbol{\theta}}_i\right)$ of the sources $i = 1 \ldots L$ is exploited (see [184]).

Example 28: Indianapolis Field Study

To demonstrate the performance of the AGMF in estimating the source parameters $\underline{\boldsymbol{\theta}}$, the data acquired during the EPRI Indianapolis field study is considered, where SF_6 tracer gas was released from a $z_s = 83.8\,\mathrm{m}$ stack at a power plant in Indianapolis, Indiana, USA. Data was recorded by 160 ground-level sensors over 19 days in September and October 1985 for 8 to 9 hours every day. Details about the field study and the data can be found in [78].

In Figure 5.5, the locations of the sensors and the sensors' concentration measurements of the 19th September 1985 are depicted. Even though all sensor measure at a hourly rate, the measurements are processed sequentially to demonstrate the on-line estimation capability of the AGMF.

The source is located at the origin and the emission rate of the tracer gas is $q = 0.0041\,\mathrm{g/s}$. Information about wind speed, wind direction, and atmospheric stability was made available by meteorological observations. The initial estimate of the source at time step $k = 0$ is given by a single Gaussian with mean vector and covariance matrix

$$\hat{\underline{\theta}}_0 = [2000\;3000\;102\;0.033]^\mathrm{T} \,, \quad \mathbf{C}_0 = \mathrm{diag}\left(\left[10^6\;10^6\;500\;0.001\right]\right)$$

respectively. Figure 5.5 and Figure 5.6 show the convergence of the source estimate towards the true source location over time and with increasing number of concentration measurements, respectively. It is important to

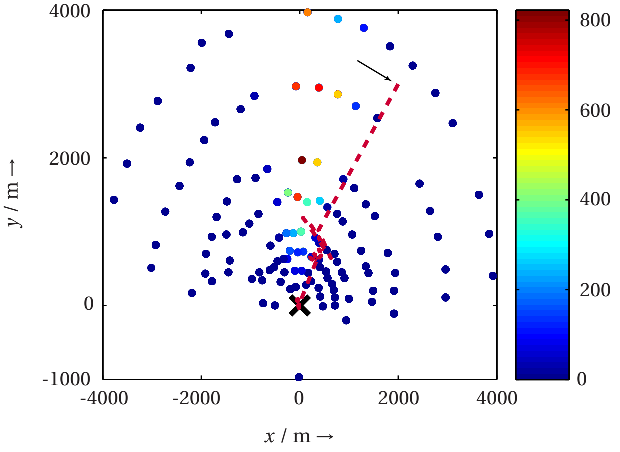


Figure 5.5: Trajectory of the estimated source location (red, dashed) and the true location of the source (black cross). Circular markers denote the sensor locations colored with the measured concentration in parts per trillion (ppt).

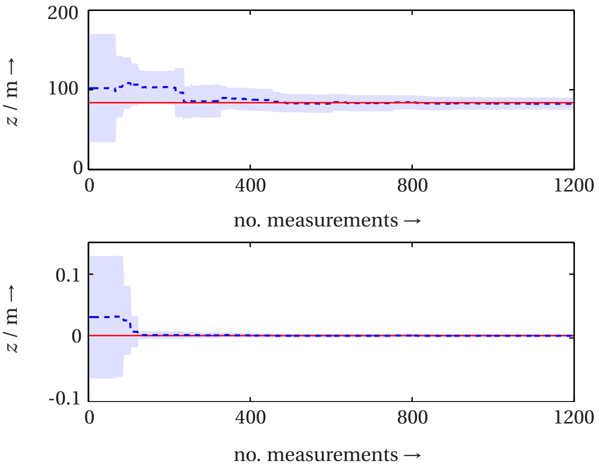


Figure 5.6: Estimate of source height z and emission rate q with increasing number of measurements. The shaded area denotes the 3-sigma confidence region and the red line indicates the true value.

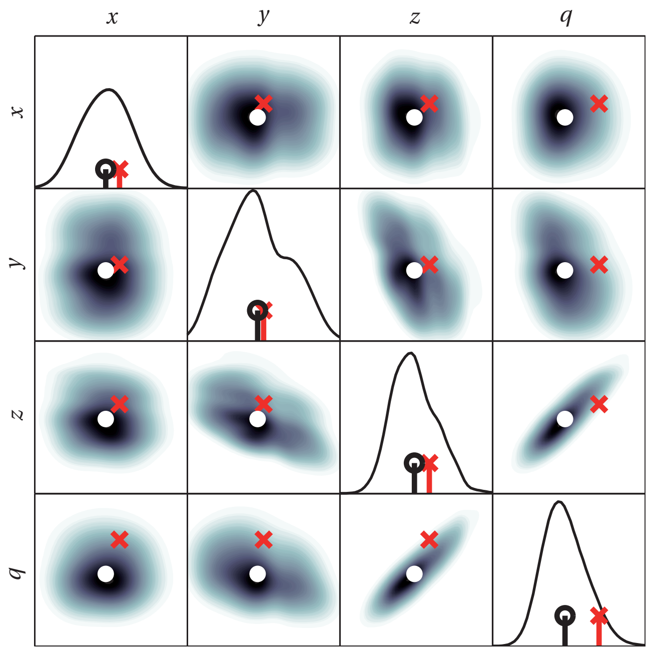


Figure 5.7: Bivariate posterior densities of the AGMF source estimate. The diagonal plots are the univariate marginal densities. Red crosses indicate the true value, while white and black circles, respectively, denote the mean of the respective density.

note that many sensor measurements (typically 60%-70%) provide a concentration measurement of almost zero as most of the sensor are outside the gas plume, as can be seen in Figure 5.5. This explains the step-wise convergence of the estimate and reduction of the variance in Figure 5.6.

The posterior density $f_k^e(\underline{\theta})$ after all $k = 1200$ measurements is depicted in Figure 5.7. It can be seen that the mean of the estimate is close to the true source parameters. Slight deviation from the ground truth is only observed for the emission rate, but still the true parameters are within the high confidence region of the estimate. Thus, the AGMF is not overconfident.

5.3 Active Object Recognition

Research on computer vision mostly focuses on the object or scene observed by the camera system. It is assumed that the parameters of the camera (e.g., position, illumination, or focus) are given or determined off-line in a time-consuming trial-and-error process involving human interaction. Particular operations are then applied on the acquired images in order to solve the considered vision task like recognizing an object. In such *passive* vision systems, the camera parameters are not adapted on-line. This is in contrast to an *active* vision system, where the next camera observation is carefully planned based on the previously acquired images and prior information about the considered scene.

While various approaches for passive object recognition exist (see e.g. [189] and references therein), active object recognition still is in its early stages. One of the first approaches to active object recognition can be found in [21], where the object models are learned via the eigenspace approach introduced in [126]. The planning algorithm greedily chooses the view that leads to the maximum entropy reduction of the object hypotheses. In [55], from a finite set of views the one maximizing the mutual information between observations and classes is selected. The approach is designed for arbitrary features, but requires approximate mutual information calculation via Monte Carlo sampling, which prevents a direct extension to continuous views. An upper bound of the Jeffrey divergence is employed in [111]. Again, merely a finite set of viewpoints is considered. Reinforcement learning approaches for active object recognition are proposed in [50, 134]. Here, learning the object models and planning is performed simultaneously. A comparison of some of the aforementioned approaches can be found in [49].

The active object recognition method proposed in this thesis consists of two parts as depicted in Figure 5.8. In the off-line *learning* part for each object a so-called object model is created. For varying camera parameters, e.g., focus or position, 2D images of each 3D object are generated. GP regression is then applied on the sample images to learn the object models.

In the on-line *recognition* part, planning the next-best camera view and Bayesian state estimation are performed alternately. For planning, mutual information is maximized with respect to the camera parameters. Based on the chosen parameter, the object estimate is updated via Bayesian estimation under consideration of the learned object models.

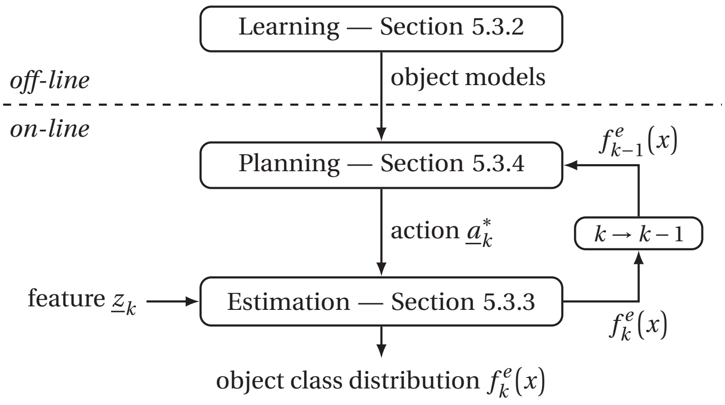

Learning — Section 5.3.2
object models
off-line
on-line
Planning — Section 5.3.4
$f^e_{k-1}(x)$
action $\underline{a}^*_k$
$k \to k-1$
feature $\underline{z}_k$
Estimation — Section 5.3.3
$f^e_k(x)$
object class distribution $f^e_k(x)$

Figure 5.8: Flow chart of the active object recognition system.

In contrast to prior art, the proposed method is very general as it is not restricted to specific image features. Furthermore, camera parameters can be arbitrary and continuous valued. All derivations hold for arbitrary GP covariance functions.

5.3.1 Object Classification

Object recognition can be considered a classification task where the state corresponds to the object class $x \in \mathcal{X} \triangleq \{x_1, x_2, \dots x_N\} \subset \mathbb{N}$, with N being the finite number of possible object classes. For classification purposes, the state is represented by means of discrete random variable $\boldsymbol{x} \in \mathcal{X}$. Based on a feature vector $\underline{z}_k \in \mathcal{Z} \subseteq \mathbb{R}^{n_z}$ acquired from images at time/stage $k = 0,1,\dots$, the goal is to estimate the true latent object class. The measurement model

$$\underline{\boldsymbol{z}}_k = \underline{h}(\boldsymbol{x}, \underline{a}_k) + \underline{\boldsymbol{v}}_k \tag{5.22}$$

relates the feature vector with the object class. The quantity $\underline{a}_k \in \mathcal{A} \subseteq \mathbb{R}^{n_a}$ is the *camera parameter* that allows actively driving the classification process. Potential camera parameters are position, orientation, focal length, or exposure time, just to name a few. For active object recognition, an appropriate camera parameter has to be chosen at every stage k in order to improve the recognition performance and speed.

5.3.2 Learning

An analytical expression of the measurement model (5.22) is not available in general as it describes a complex transformation of a potentially high-dimensional feature vector to an abstract object class. To overcome this issue, a GP model is learned to represent (5.22). As the feature is typically multi-dimensional, for each dimension $e = 1 \dots n_z$ of $\underline{\boldsymbol{z}}_k$ a separate GP is learned independently using the same training inputs $\mathbf{X}$ but different training outputs $\underline{\hat{z}}^e \triangleq \left[\hat{z}_1^e \; \dots \; \hat{z}_n^e\right]^{\mathrm{T}}$. An alternative to this procedure are so-called *multi-output* GPs [25].

Furthermore, learning the GPs for each feature vector dimension has to be performed independently for each object class x_i, $i = 1 \dots N$. This results in N multivariate GPs $\underline{\boldsymbol{h}}_i(.) \sim \mathcal{GP}$ of dimension n_z named *object models* in the following. To learn an object model $\underline{\boldsymbol{h}}_i$, samples $\underline{a}_l$, $l = 1 \dots n$ of the parameter space $\mathcal{A}$ are used as training inputs $\mathbf{X}$. For each input sample $\underline{a}_l$, an object of the class $x_i \in \mathcal{X}$ is observed by the camera resulting in the feature vector $\underline{\hat{z}}_l = \left[\hat{z}_l^1 \; \hat{z}_l^2 \; \dots \; \hat{z}_l^{n_z}\right]^{\mathrm{T}}$ acting as training output. In total, for n_z output dimensions and N object classes, $n_z \times N$ GPs are learned. Since learning these measurement models is an offline task (see Figure 5.8), the required computation time is independent of the computation time for object recognition. Furthermore, for high-dimensional features, which may be obtained for instance by means of the scale-invariant feature transform (SIFT, [115]), dimensionality reduction techniques like principal component analysis [2] or GP latent variable models [199] can be employed in order to reduce the number of GPs to be learned.

5.3.3 Estimation

To estimate the object class for a given camera parameter $\underline{a}_k$ and feature vector $\underline{\hat{z}}_k$ the Bayesian measurement update step (1.7) is employed[2], where the density $f_k^e(x)$ in the given object recognition task corresponds to a discrete distribution modeled as a mixture of Kronecker deltas according to

$$f_k^e(x) = \sum_{i=1}^{N} \omega_{k,i} \cdot \delta_{x,i} \; . \tag{5.23}$$

2 Due to the implicit assumption that the feature vectors $\underline{\boldsymbol{z}}_k$ for $k = 0,1,\dots$ are conditionally independent given $\boldsymbol{x}$, this form of Bayesian classification/object recognition is known as *naive Bayes classifier* [60]. Even though this assumptions might not be true, naive Bayes classifiers showed a good classification performance in practice [57].

As the state is static—the object does not change its class over time—no prediction is performed and thus it holds that $f_k^p(x) \equiv f_{k-1}^e(x)$. The weight $\omega_{k,i}$ represents the probability that object x belongs to class i. The weights are non-negative and sum up to one. The measurement update boils down to updating the weights whenever a new feature vector $\underline{\hat{z}}_k$ is available. Before providing the weight update equation, it is first necessary to investigate the structure of the likelihood.

Likelihood

In contrast to (1.7), the likelihood in the considered recognition task also depends on the camera parameter. In case of a given object class $x = i$, the likelihood $f\left(\underline{\hat{z}}_k \middle| x = i, \underline{a}_k\right)$ corresponds to the GP $\underline{\boldsymbol{h}}_i$. If in addition the camera parameter $\underline{a}_k$ is given, the likelihood becomes a Gaussian density $\mathcal{N}\left(\underline{\hat{z}}_k; \underline{\mu}_{k,i}^z, \mathbf{C}_{k,i}^z\right)$ with mean vector and covariance matrix according to

$$\begin{aligned} \underline{\mu}_{k,i}^z &= \left[\mu_{k,i}^1 \; \mu_{k,i}^2 \; \dots \; \mu_{k,i}^{n_z}\right]^{\mathrm{T}} , \\ \mathbf{C}_{k,i}^z &= \operatorname{diag}\!\left(\left(\sigma_{k,i}^1\right)^2, \left(\sigma_{k,i}^2\right)^2, \dots, \left(\sigma_{k,i}^{n_z}\right)^2\right) , \end{aligned} \tag{5.24}$$

respectively. The elements in (5.24) corresponding to dimension $e = 1 \dots n_z$ are calculated according to (4.6) and (4.7), respectively, with $\underline{a}_k$ acting as test input $\underline{x}$ and $\underline{\hat{z}}^e$ being the training output vector $\underline{y}_{\mathcal{D}}$. Overall, the likelihood for a fixed $\underline{a}_k$ can be characterized by means of the conditional density

$$f\left(\underline{\hat{z}}_k \middle| x, \underline{a}_k\right) = \sum_{i=1}^{N} \delta_{x,i} \cdot \mathcal{N}\left(\underline{\hat{z}}_k; \underline{\mu}_{k,i}^z, \mathbf{C}_{k,i}^z\right) . \tag{5.25}$$

It is important to note that for a fixed feature vector $\underline{\hat{z}}_k$—as required for solving Bayes' equation—the conditional density in (5.25) becomes a weighted sum of Kronecker deltas as in (5.23), because all Gaussian components are evaluated at $\underline{\hat{z}}_k$ and thus, become scalar weighting coefficients.

Posterior Weights

Given the likelihood in (5.25), the measurement update can be evaluated analytically resulting in a weight update at stage k according to

$$\omega_{k,i} = c_k \cdot \omega_{k-1,i} \cdot \mathcal{N}\left(\underline{\hat{z}}_k; \underline{\mu}_{k,i}^z, \mathbf{C}_{k,i}^z\right)$$

for object class $i = 1 \dots l$, where $c_k = \left(\sum_i \omega_{k-1,i} \cdot \mathcal{N}\left(\hat{\underline{z}}_k; \underline{\mu}^z_{k,i}, \mathbf{C}^z_{k,i}\right)\right)^{-1}$ is a normalization constant and $\omega_{k-1,i}$ are the weights of $f^e_{k-1}(x)$.

5.3.4 Planning

So far, the camera parameter $\underline{a}_k$ was assumed to be given. But in active object recognition, an action is chosen automatically by the imaging system itself for acquiring high informative observations. For this purpose, the optimization problem

$$\underline{a}^*_k = \arg\max_{\underline{a}_k} \mathrm{I}\left(\boldsymbol{x}, \underline{\boldsymbol{z}}_k | \underline{a}_k\right) \tag{5.26}$$

is formulated to determine the optimal action $\underline{a}^*_k$ to be applied at stage k. Since solving (5.26) results in the camera parameters to be applied next, it is often referred to as *next-best-view* planning (see e.g. [153]). As objective function in (5.26), the *mutual information*

$$\mathrm{I}\left(\boldsymbol{x}, \underline{\boldsymbol{z}}_k | \underline{a}_k\right) = \iint f\left(x, \underline{z}_k\right) \cdot \log \frac{f\left(x, \underline{z}_k\right)}{f(x) \cdot f\left(\underline{z}_k\right)} \,\mathrm{d}x\,\mathrm{d}\underline{z}_k \tag{5.27}$$

between state and feature vector given a camera parameter is considered. This measure quantifies the amount of information the knowledge of an observation reveals about the state and vice versa. It is closely related to Shannon's entropy and zero only iff both variables are independent [45].

Unfortunately, an analytical calculation of the mutual information is not possible as it requires evaluating the logarithm of a Gaussian mixture representing $\underline{\boldsymbol{z}}_k$. To obtain a computationally cheap and robust approximation, the alternative formulation of the mutual information according to

$$\mathrm{I}\left(\boldsymbol{x}, \underline{\boldsymbol{z}}_k | \underline{a}_k\right) = \mathrm{H}\left(\underline{\boldsymbol{z}}_k | \underline{a}_k\right) - \mathrm{H}\left(\underline{\boldsymbol{z}}_k | \boldsymbol{x}, \underline{a}_k\right) \tag{5.28}$$

is employed, where $\mathrm{H}(\boldsymbol{x})$ is the differential entropy

$$\mathrm{H}(\boldsymbol{x}) = -\int f(x) \cdot \log f(x) \,\mathrm{d}x \;.$$

The second term in (5.28) has an analytical expression, while the first term can be bounded from below as shown in [85].

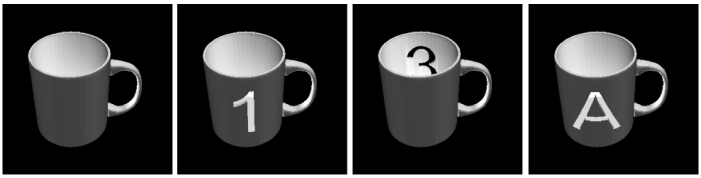


Figure 5.9: Cups with different labels.

The optimization problem in (5.26) neither is convex nor possesses it a closed-form solution. To increase the probability of finding the optimal camera parameter or at least to ensure finding a parameter that is very close to the optimal one, so-called *multi-start optimization* is performed (see e.g. [179]). Here, optimization is repeated from varying initial points. To cover the camera parameter space $\mathcal{A}$ uniformly, the initial points form a regular grid on $\mathcal{A}$.

Example 29: Recognizing Cups

The object to be recognized in this example are synthetically generated cups as depicted in Figure 5.9. Eight different cups exist, all being identical except for a label that is cut through the surface. The labels of six cups are visible from the same perspective, one is visible from the opposite point of view and one cup is not labeled at all.

For learning and recognition, 100×100 pixel normalized grayscale images are generated from the cups, where zero-mean Gaussian noise with variance 14.7 is added. 1D and 2D features are extracted from the images. In the 1D case, the mean gray value is considered. The eigenspace or principal component decomposition approach proposed in [126] is used for extracting 2D features, where the two largest eigenvalues are taken into account.

By means of the camera parameter, the position can be changed in one or two dimensions. In the 1D case, the camera moves on a circle that is parallel to the horizontal plane and centered at the object. In the 2D case, the camera position can be varied on a sphere centered at the object. Here, the actions correspond to the azimuth and elevation angles.

To learn the GPs, each dimension of the action space is sampled regularly in 10 decimal degree steps, i.e., for the one-dimensional circular action space, this leads to 36 sample images.

For comparison, the following active object recognition approaches are considered:

Planner The proposed approach, where 5 and 15 initial points for optimization are exploited for the 1D and 2D action space, respectively.

Grid An approach similar to [55], where at each stage the action maximizing the mutual information is taken from a finite set. Here, this finite set coincides with the set of initial points of the Planner.

Random Actions are selected uniformly at random.

For each combination of feature and action space, 50 MC simulation runs are performed, where the true cup is selected uniformly at random. The initial distribution $f_0^e(x)$ is uniform. A decision about the object type is made if either the probability (weight) of one object class exceeds 0.95 or after eight stages.

For the 2D action space, the mutual information surface for three cups is plotted in Figure 5.10a. Here, the optimal action is indicated by the red circle, which corresponds to an elevation angle of approximately 45^o. For this action, the corresponding views on the three cups are depicted in Figure 5.10b–d. It can be seen that this view facilitates to look inside the cups and thus, allows an easy discrimination of all three cups.

The average values over the 50 simulation runs in terms of recognition rate, number of views, and maximum object probability are listed in Table 5.1. It can be seen that the Planner performs best with respect to almost any performance indicators. In comparison to Random, the number of stages after which a recognition decision is made is significantly lower. Simultaneously, the certainty in this decision is much higher as the average maximum object probability indicates. The performance of the Grid approach is often close to the proposed approach. But the significantly lower number of views of the Planner shows the benefits of performing a continuous optimization for next-best-view planning. In contrast to both Grid and Random, the proposed Planner can take advantage of an increasing feature and action dimension, i.e., with an increasing dimension the recognition rate increases as well and the number of views decreases.

A high object probability not necessarily coincides with the best recognition rate as seen in the case of the 1D action space and 2D feature space. While

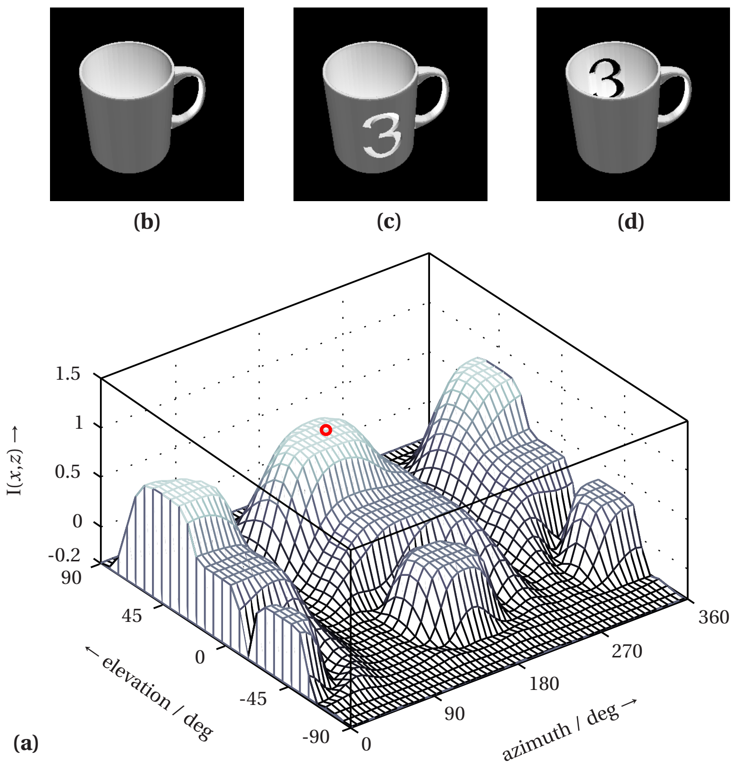


Figure 5.10: (a) Lower bound of mutual information with optimal view/action (red circle). (b)–(d) View of three of the cups corresponding to the optimal action.

Table 5.1: Cup recognition. (a) recognition rate in percent, (b) average number of views, (c) average maximum object probability.

Dim.	Planner			Grid			Random		
$\mathcal{A}$ / $\mathcal{Z}$	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
1 / 1	**66**	**6.06**	**0.74**	62	6.1	0.71	50	7.32	0.53
1 / 2	88	**3.08**	**0.97**	74	4.96	0.89	**94**	6.88	0.81
2 / 1	**92**	**2.5**	**0.99**	62	4.1	0.95	76	6.34	0.70
2 / 2	**100**	**1.88**	**0.99**	88	2.5	0.97	68	6.92	0.74

Random merely relies on the GP object models for estimation, Grid and Planner additionally use the models for planning. Thus, a bootstrapping effect can cause the decision maker to get stuck in a repetitive pattern. The quality of the GP models is essential for the recognition process and thus, under- and over-fitting require special attention.

5.4 Summary

For the applications considered in this chapter, not only the novel algorithms proposed in the previous chapters of this thesis have been turned into practice. Also several additional contributions have been suggested:

- *(Semi-)Analytical measurement updates for position estimation and pose estimation:* By considering a squared range measurement equation a novel computationally efficient closed-form position estimation is derived. In case of pose estimation, this analytical solution can be exploited thanks to the nonlinear-nonlinear decomposition proposed in Section 2.5.2.
- *On-line source estimation based on Gaussian plume model:* The state-of-the-art focuses on off-line MCMC methods for source estimation. By employing the Gaussian plume dispersion model as a measurement model and by modeling the unknown source parameters as state vector with Gaussian mixture density representation facilitates a computationally light-weight but highly accurate source estimation.
- *Gaussian process object models for active object recognition:* Instead of a discretization of the camera parameter space, which is common in the state-of-the-art, the mapping from the latent object class to the feature vector is learned by means of GP regression. This approach can be applied in various recognition scenarios as it is not restricted to specific features, camera parameters, or covariance functions.
- *Efficient next-best-view planning based on lower bound approximation of mutual information:* The GP object models together with a novel lower bound on the mutual information allow optimizing the camera parameters on a fine-grained level and with low computational overhead.

6

Concluding Remarks

6.1 Conclusions

Gaussian filtering lays the foundation to all contributions made in this thesis. In many real-world applications, the Gaussian assumption is valid and thus, sufficient for accurate filtering. In addition, thanks to its algebraical simplicity, Gaussian filtering can be performed in an efficient and scalable manner. The contributions in this thesis exploit closed-form solutions that exist for particular nonlinear models, which leads to a further improvement of the estimation performance and computational efficiency. By means of the proposed decomposition techniques and polynomial approximation, these benefits can even be utilized for problems that are not fully covered by any of the closed-form cases. The experiments and simulations performed for position and pose estimation provide evidence that the proposed Gaussian filtering techniques lead to an improvement compared to state-of-the-art approaches like the EKF or the UKF.

For more complex filtering problems, where the Gaussian assumption no longer holds, the contributions made for Gaussian filtering are not necessarily inapplicable. Quite the contrary, embedding Gaussian filtering into a Gaussian mixture framework turns out to be a very powerful estimation tool. Most of the simplicity of Gaussian filtering remains by this migration. Merely some additional "management" tasks have to be performed for Gaussian mixture filtering. The contributions in this thesis serve these tasks, which are mainly concerned with controlling the number of mixture components. Especially for the proposed adaptive splitting scheme there is experimental evidence that by exploiting the linearization error for introducing components leads to a significant improvement of the estimation performance, while the number of additional components can be kept on a low level. It has been shown that continually adding and removing components is better than filtering with a constantly high number of mixture components. Furthermore, all computations can be performed on-line, which is beneficial over other accurate estimation techniques like MCMC, which only allows off-line or batch processing.

In case of missing mathematical models that describe the system dynamics and sensor characteristics, Gaussian process regression is suggested to learn probabilistic models from data. Given such a GP model, filtering and smoothing can be performed in closed form when restricting to Gaussian distributions. By means of simulations it has been shown that the obtained estimation performance is superior compared to state-of-the-art Gaussian filters, even in cases where these filters utilize the exact model. To bound the computational complexity with a growing data set, a recursive GP regression algorithm has been proposed in addition. Here, the hyperparameters of the GP are learned by means of utilizing the proposed Gaussian filtering techniques, which facilitates on-line learning.

Besides aiming for computationally efficient Bayesian filtering, the contributions made in this thesis lead to a lower user involvement. That is, many of the proposed algorithms can be operated in a black-box fashion. Examples are the automatic Chebyshev series expansion by means of the discrete cosine transform (Section 2.5.3), adaptive Gaussian mixture splitting (Section 3.4.1), automatic model selection in the mixture reduction (Section 3.4.3), or the on-line hyperparameter optimization for GPs (Section 4.5.4). In doing so, the application of these algorithms to a given problem can be simplified, which reduces deployment time and operational costs.

6.2 Future Work

In all three pillars considered in this thesis there is enough room for further improvements and extensions. In the following, an outlook on future work is provided.

Gaussian Filtering

One of the contributions in this thesis that currently requires a significant amount of manual inspection is the nonlinear-nonlinear decomposition in Section 2.5.2. An expert has to investigate manually which parts of the state are analytically integrable and which are not. To facilitate an automatic decomposition, one idea is to utilize Risch's algorithm[1] [149, 150] together with a decomposition exploration algorithm similar to the one proposed in [102].

The moment homotopy for polynomial nonlinearities in Section 2.5.5 is a first step towards removing the joint Gaussian assumption between state and measurement. In general, it is desirable to perform Gaussian filtering without this assumption for arbitrary nonlinearities to improve the robustness and to reduce the estimation error. First approaches in this direction can be found in [74, 105].

The measurement update of the Gaussian filter for polynomial nonlinearities can naturally output a full exponential density representation. Maintaining this representation also over the prediction, however, is more difficult as neither the predicted density nor even the predicted moments can be expressed analytically. Even if the predicted moments were available, determining an exponential density that matches the moments is also not possible in closed form. See [147] for a first step to solve this issue.

The CPKF proposed in Section 2.5.3 is operational for one-dimensional states so far. The next step is to extend it to multiple dimensions. While this is straightforward in terms of the Chebyshev series expansion, closed-form moment calculation for Gaussians mapped through multi-dimensional polynomials is still computationally demanding. The results in [97] already lowered the computations significantly compared to previous approaches, but the moment recursion proposed in Section 2.5.4 might offer a way to a further reduction.

Gaussian Mixture Filtering

Thanks to the individual processing of the Gaussian components considered in this thesis, Gaussian mixture filters directly benefit from any improvement achieved for Gaussian filters. Thus, the outlook on future work for mixture filters is mainly focused on the refinement and reapproximation operations of Algorithm 3.

1 More precisely, a realization of this algorithm in modern computer algebra systems.

The SGMR algorithm proposed in Section 3.4.3 so far is only applicable for one-dimensional and two-dimensional mixtures due to the use of the curvature as roughness penalty. Thus, future work is dedicated to explore and propose curvature measures for higher dimensions.

Splitting a Gaussian component into many as discussed in Section 3.4.2 takes the linearization error into account. This procedure is generally applicable, but can be enhanced by application-specific criteria. For instance, for the source estimation application considered in Section 5.2, it might be beneficial to also take the distance between sensor location and component mean into account. This avoids situations where no component is split due to large distances—the Gaussian plume model is then approximately linear. In this case, the AGMF degenerates to a simple Gaussian mixture filter with a fixed number of components.

The source estimation application gives room for further improvements. Currently it is assumed that the number of sources is known a priori, but actually new sources might appear spontaneously or an existing source might disappear over time. Such birth and death processes being common in multiple target tracking (see e.g. [185]) should be incorporated.

Gaussian Process Filtering

For the GP-ADF and GP-RTSS proposed in Section 4.5.1 and Section 4.5.2, respectively, it is assumed that training data of the state is given, but as this state is hidden, this assumption is impractical for many applications. The GP for the system and the measurement model has to be learned without the need of direct access to the hidden states. This can be achieved by means of Expectation Maximization since both GP-ADF and GP-RTSS allow for gradient-based parameter optimization.

The number and placement of the basis vectors required for the RGP in Section 4.5.3 has not been discussed. The algorithm supports adding and removing basis vectors on-line, which allows correcting an insufficient initial selection of basis vectors. However, a criterion that facilitates a good choice of new basis vectors is left for future work. Techniques used in active learning [109] or sensor planning [82] for instance can be utilized for this purpose.

Rather straightforward to extend is the on-line hyperparameter learning proposed in Section 4.5.4. Instead of restricting to a Gaussian representation of the hyperparameters, also Gaussian mixtures can be used by means of exploiting the techniques proposed in Chapter 3.

A

Particle Filtering

Monte Carlo (MC) methods for solving the integrals appearing in Bayesian filtering became popular from the 1980s on, with the advent of cheap but powerful micro-processors. In contrast to the previously popular Kalman filtering methods and its derivatives, MC methods make no assumptions regarding the models or density functions. In the following, a brief introduction to particle filters is given, which are a popular form of MC approximations to Bayesian filtering.

A.1 Perfect Monte Carlo Sampling

MC methods rely on a non-parametric representation of the density function $f\left(\underline{x}\,|\,\underline{\hat{z}}_{0:k}\right)$ by means of a set of n independent and identically distributed samples $\underline{x}^{(i)}$, $i = 1 \ldots n$. These samples are often named particles, which led to the naming *particle filters.* Given the sample set, the density function can be approximated as a sum of Dirac delta distributions according to

$$f\left(\underline{x}\,|\,\underline{\hat{z}}_{0:k}\right) \approx \frac{1}{n}\sum_{i=1}^{n} \delta\left(\underline{x} - \underline{x}_i\right) . \tag{A.1}$$

For an arbitrary nonlinear function $\underline{g}(\underline{x}) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$, this representation leads to a perfect MC approximation of the expectation calculation

$$\mathrm{E}\left\{\underline{g}(\underline{\boldsymbol{x}})\right\} = \int \underline{g}(\underline{x}) \cdot f\left(\underline{x} \,|\, \hat{\underline{z}}_{0:k}\right) \mathrm{d}\underline{x} \overset{(\mathrm{A.1})}{\approx} \frac{1}{n} \sum_{i=1}^{n} \underline{g}\left(\underline{x}^{(i)}\right) . \tag{A.2}$$

The central limit theorem guarantees that the MC approximation converges with an increasing number of particles, regardless of the dimension of $\underline{\boldsymbol{x}}$. This dimensionless property is unique to MC methods compared to other (deterministic) numerical integration methods, at least from a theoretical point of view [58]. However, practice shows that the number of required particles also grows exponentially with the dimension of $\underline{\boldsymbol{x}}$ (see e.g. [48]).

Despite the nice theoretical properties of MC approximation, sampling from $f\left(\underline{x} \,|\, \hat{\underline{z}}_{0:k}\right)$ is often very difficult as the density typically has a complicated functional form and is only known up to a normalization constant. A solution to this issue is *importance sampling*.

A.2 Importance Sampling

The key idea of importance sampling is to use an approximation density $\pi\left(\underline{x} \,|\, \hat{\underline{z}}_{0:k}\right)$ called *importance function* instead of $f\left(\underline{x} \,|\, \underline{z}_{1:k}\right)$. Samples can be drawn much easier from the importance function [183]. If it holds that $\pi\left(\underline{x} \,|\, \hat{\underline{z}}_{0:k}\right) > 0$ whenever $f\left(\underline{x} \,|\, \hat{\underline{z}}_{0:k}\right) > 0$ then the expectation in (A.2) can be decomposed to

$$\mathrm{E}\left\{\underline{g}(\underline{\boldsymbol{x}})\right\} = \int \left(\underline{g}(\underline{x}) \frac{f\left(\underline{x} \,|\, \hat{\underline{z}}_{0:k}\right)}{\pi\left(\underline{x} \,|\, \hat{\underline{z}}_{0:k}\right)}\right) \pi\left(\underline{x} \,|\, \hat{\underline{z}}_{0:k}\right) \mathrm{d}\underline{x} = \mathrm{E}\left\{\underline{g}(\underline{\boldsymbol{x}}) \cdot \omega(\underline{\boldsymbol{x}})\right\} ,$$

with weight $\omega(\underline{x}) \triangleq \frac{f(\underline{x}|\hat{\underline{z}}_{0:k})}{\pi(\underline{x}|\hat{\underline{z}}_{0:k})}$. By now drawing samples $\underline{x}_i$ from the importance function and not from the density $f\left(\underline{x} \,|\, \hat{\underline{z}}_{0:k}\right)$, the expectation in (A.2) can be approximated as

$$\mathrm{E}\left\{\underline{g}(\underline{\boldsymbol{x}})\right\} = \sum_{i=1}^{n} \omega^{(i)} \cdot \underline{g}\left(\underline{x}^{(i)}\right)$$

with normalized weights

$$\omega^{(i)} = \frac{\omega\left(\underline{x}^{(i)}\right)}{\sum_{i=1}^{n} \omega\left(\underline{x}^{(i)}\right)} \tag{A.3}$$

Thus, the set of particles now comprises the particles itself and the corresponding weights (A.3). Accordingly, the density is approximated by means of a weighted sum of Dirac delta distributions

$$f\left(\underline{x}\,|\,\hat{\underline{z}}_{0:k}\right) \approx \sum_{i=1}^{n} \omega^{(i)} \cdot \delta\left(\underline{x} - \underline{x}^{(i)}\right) .$$

A.2.1 Sequential Importance Sampling

Importance sampling as introduced above does not allow for recursive filtering as required for models of the form

$$\begin{aligned} \underline{\boldsymbol{x}}_k &\sim f\left(\underline{x}_k | \underline{x}_{k-1}\right) , \\ \underline{\boldsymbol{z}}_k &\sim f\left(\underline{z}_k | \underline{x}_k\right) , \end{aligned}$$

where the state $\underline{\boldsymbol{x}}_k$ varies with the time k. Instead, one would have to recalculate the weights whenever new measurements become available, which leads to a growing computational demand with an increasing number of measurements and time, respectively. The main reason of this drawback lays in the definition of the importance function. For recursive processing, the importance function itself has to follow a recursion according to

$$\pi\left(\underline{x}_{0:k} | \hat{\underline{z}}_{0:k}\right) = \pi\left(\underline{x}_k | \underline{x}_{0:k-1}, \hat{\underline{z}}_{0:k}\right) \cdot \pi\left(\underline{x}_{0:k-1} | \hat{\underline{z}}_{0:k-1}\right) ,$$

which leads to a recursive expression for the weights

$$\omega_k^{(i)} \propto \frac{f\left(\hat{\underline{z}}_k | \underline{x}_k^{(i)}\right) \cdot f\left(\underline{x}_k^{(i)} | \underline{x}_{k-1}^{(i)}\right)}{\pi\left(\underline{x}_k^{(i)} | \underline{x}_{0:k-1}^{(i)}, \hat{\underline{z}}_{0:k}\right)} \cdot \underbrace{\frac{f\left(\underline{x}_{0:k-1} | \hat{\underline{z}}_{0:k-1}\right)}{\pi\left(\underline{x}_{0:k-1}^{(i)} | \hat{\underline{z}}_{0:k-1}\right)}}_{\propto\, \omega_{k-1}^{(i)}} . \tag{A.4}$$

This recursion follows from the observation that the samples $\underline{x}_{0:k-1}^{(i)}$ have already been drawn from the importance function $\pi\left(\underline{x}_{0:k-1} | \hat{\underline{z}}_{0:k-1}\right)$ and the weights $\omega_{k-1}^{(i)}$ have already been calculated in the time step $k-1$. Thus, the samples $\underline{x}_{0:k}^{(i)}$ can been obtained from $\pi\left(\underline{x}_{0:k} | \hat{\underline{z}}_{0:k}\right)$ by drawing $\underline{x}_k^{(i)}$ from $\pi\left(\underline{x}_k | \underline{x}_{0:k-1}^{(i)}, \hat{\underline{z}}_{0:k}\right)$. This efficient MC approximation of drawing samples and calculating the weights according to (A.4) is called *sequential importance sampling* (SIS), which leads to the approximation

$$f_k^e(\underline{x}_k) \approx \sum_{i=1}^{n} \omega_k^{(i)} \cdot \delta\left(\underline{x}_k - \underline{x}_k^{(i)}\right) \tag{A.5}$$

of the posterior density of the state $\underline{\boldsymbol{x}}_k$.

A.2.2 Choice of Importance Function

The SIS is still formulated quite generally as the concrete choice of the importance function leaves many degrees of freedom. A common further very practical restriction made is to apply the Markov assumption, which leads to

$$\pi\left(\underline{x}_k | \underline{x}_{0:k-1}, \hat{\underline{z}}_{0:k}\right) = \pi\left(\underline{x}_k | \underline{x}_{k-1}, \hat{\underline{z}}_{0:k}\right) .$$

In doing so, it is no longer necessary to store the whole particle trajectory $\underline{x}_{0:k}^{(i)}$, but only the current particles $\underline{x}_k^{(i)}$. It can be shown, that the *optimal importance function* minimizing the variance of the weights $\omega_k^{(i)}$ is given by

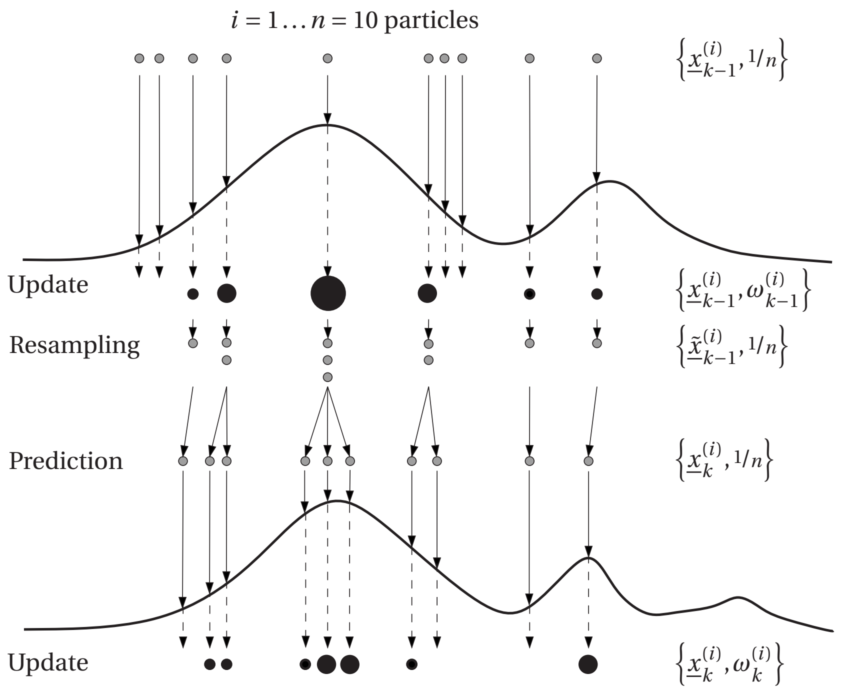
$$\pi\left(\underline{x}_k | \underline{x}_{k-1}, \hat{\underline{z}}_{0:k}\right) = f\left(\underline{x}_k | \underline{x}_{k-1}, \hat{\underline{z}}_k\right) ,$$

but unfortunately the optimal importance function is typically not given in analytic form and drawing samples from it is not possible [59]. One approximation often applied to circumvent this issue is it utilize Gaussian filtering techniques like linearization (see Section 2.2.3) or linear regression (see Section 2.2.5), which leads for instance to the extended particle filter or unscented particle filter [200].

Another variation of SIS is the employ the transition density $f\left(\underline{x}_k | \underline{x}_{k-1}\right)$ as importance function. This choice leads to the *bootstrap filter* [70] allowing a very simple implementation as drawing the particles $\underline{x}_k^{(i)}$ corresponds to evaluating the system function $\underline{a}_k(.)$ on the given particles $\underline{x}_{k-1}^{(i)}$ and samples from the system noise $\underline{\boldsymbol{w}}_k$. The drawback of the bootstrap filter is that no measurement information is used for drawing $\underline{x}_k^{(i)}$, which typically leads to a very large number of particles for accurate estimates.

A.2.3 Resampling

By applying SIS it easily happens that over time almost all particles have a weight of (nearly) zero, which effectively reduces the number of particles. To overcome



$i = 1 \ldots n = 10$ particles
$\left\{\underline{x}_{k-1}^{(i)}, {}^{1}/{}_{n}\right\}$
Update
$\left\{\underline{x}_{k-1}^{(i)}, \omega_{k-1}^{(i)}\right\}$
Resampling
$\left\{\underline{\tilde{x}}_{k-1}^{(i)}, {}^{1}/{}_{n}\right\}$
Prediction
$\left\{\underline{x}_{k}^{(i)}, {}^{1}/{}_{n}\right\}$
Update
$\left\{\underline{x}_{k}^{(i)}, \omega_{k}^{(i)}\right\}$

Figure A.1: SIR starts with a particle set $\left\{\underline{x}_{k-1}^{(i)}, {}^{1}/{}_{n}\right\}$ at time step $k-1$. The weight of each particle is updated given the current measurement value $\underline{\hat{z}}_{k-1}$, which results in the particle set $\left\{\underline{x}_{k-1}^{(i)}, \omega_{k-1}^{(i)}\right\}$ representing the posterior $f_{k-1}^{e}\left(\underline{x}_k\right)$. The resampling step duplicates the particles with high weights. The resulting particle set $\left\{\underline{\tilde{x}}_{k-1}^{(i)}, {}^{1}/{}_{n}\right\}$ still represents the posterior $f_{k-1}^{e}\left(\underline{x}_k\right)$. Drawing the particles $\underline{x}_k^{(i)}$ from the importance function, which corresponds to the prediction step, leads to the particle set $\left\{\underline{x}_{k}^{(i)}, {}^{1}/{}_{n}\right\}$ representing the predicted density $f_k^{p}\left(\underline{x}_k\right)$. (Image adapted from [58])

this *sample degeneration* problem a so-called *resampling* step has to be applied in addition. Here, after performing the weight update according to (A.4), n new particles are drawn from the posterior (A.5) that replace the current particle set. The new particles all have the same weight ${}^{1}/{}_{n}$.

Over the years many resampling methods have been proposed (see e.g. [36, 101, 114]). The key idea of any resampling is to remove particles with small weights and duplicate those with a large weight. MC methods with an additional resampling step are called *sequential importance resampling* (SIR), which are more known under the term *particle filter* (PF). In Figure A.1 the steps of a PF are depicted.

B

Performance Measures

In this chapter, some measures quantifying the estimation performance of Bayesian filtering algorithms are briefly introduced.

B.1 Root Mean Square Error

Generally, the error between the estimated mean $\underline{\mu}_k^x$ and the true state $\underline{x}_k$ is defined as

$$\underline{e}_k \triangleq \underline{x}_k - \underline{\mu}_k^x \,, \tag{B.1}$$

which is a vector of dimension-wise errors $e_{k,i} = x_{k,i} - \mu_{k,i}^x$ for $i = 1 \ldots n_x$. A standard performance measure for Bayesian filters depending on the error (B.1) is the *root mean square error* (rmse) defined by

$$\text{rmse}_i \triangleq \sqrt{\frac{1}{K} \sum_{k=1}^{K} e_{k,i}^2}$$

for dimension $i = 1 \ldots n_x$, where K is the number of time steps.

In this thesis, MC simulations are used besides real data in order to evaluate the performance of a filter. In case of MC simulations, the rmse for each time step over the different simulations can be evaluated. Hence, the rmse for time step k and dimension i is given by

$$\mathrm{rmse}_i^{\mathrm{MC}} \triangleq \sqrt{\frac{1}{n_{\mathrm{MC}}} \sum_{m=1}^{n_{\mathrm{MC}}} \left(e_{k,i}^m\right)^2}$$

where $e_{k,i}^m$ is the estimation error of dimension i for the mth MC simulation run and n_{MC} is the number of MC simulation runs.

B.2 Mean Absolute Error

A performance measure very similar to the rmse is the *mean absolute error* (mae), which is defined as

$$\mathrm{mae}_i \triangleq \frac{1}{K} \sum_{k=1}^{K} \left|e_{k,i}\right| \tag{B.2}$$

for dimension $i = 1 \ldots n_x$.

B.3 Normalized Estimation Error Square

The rmse and mae merely takes the mean of the estimated state into account. The *normalized estimation error square* (nees) considers the uncertainty of the estimation in addition. It is defined as

$$\mathrm{nees}_k \triangleq \left(\underline{x}_k - \underline{\mu}_k^x\right)^{\mathrm{T}} \left(\mathbf{C}_k^x\right)^{-1} \left(\underline{x}_k - \underline{\mu}_k^x\right) ,$$

which corresponds to a weighted Euclidean distance of the state errors at time step k, where the weight is given by the inverse state covariance matrix. Due to incorporating the inverse covariance matrix, a large estimation error has a small contribution to the nees if the covariance is large and conversely, a large estimation error contributes more in case of a small covariance. Thus, this measure allows indicating a consistent filter.

In contrast to the rmse, the nees is a dimensionless quantity. It is χ^2-distributed with n_x degrees of freedom if the state estimate is Gaussian [71]. The nees is also known as *Mahalanobis distance*.

B.4 Negative Log-Likelihood

An alternative to the nees that is often used in machine learning is the *negative log-likelihood* (nll)

$$\begin{aligned} &\text{(filtering)} \quad \mathrm{nll}_k \triangleq -\log f_k^x\big(\underline{x}_k\big) \,, \\ &\text{(learning)} \quad \mathrm{nll} \triangleq -\sum_{i=1}^{n} \log f\big(\underline{\hat{z}}_i \big| \underline{x}\big) \,, \end{aligned} \tag{B.3}$$

where the first term is used for performance measurement in filtering, while the second term is used in a model learning context e.g. via GPs in this thesis. The dependence of the latter definition on the likelihood $f\big(\underline{\hat{z}}\big| \underline{x}\big)$ explains the naming of this performance measure. In general, the nll quantifies how well a realization (true state or measurement value) can be explained by the given (estimated or learned) density function.

In case of a Gaussian state density, the first nll term in (B.3) can be formulated to

$$\mathrm{nll}_k = \log\sqrt{\left|2\pi \mathbf{C}_k^x\right|} + \tfrac{1}{2}\left(\underline{x}_k - \underline{\mu}_k^x\right)^{\mathrm{T}} \left(\mathbf{C}_k^x\right)^{-1} \left(\underline{x}_k - \underline{\mu}_k^x\right) \,.$$

The second term in (B.3) can be resolved similarly. It can be see that the nll consists of the nees and a term that penalizes a too large covariance matrix, which allows discovering of overestimation.

C

Quadratic Programming

Optimization problems with quadratic objective function and affine constraints are called *quadratic programs* (QPs). Hence, a quadratic program is defined as

$$\begin{aligned} \min_{\underline{x}} \quad & \tfrac{1}{2} \cdot \underline{x}^{\mathrm{T}} \mathbf{Q} \underline{x} + \underline{q}^{\mathrm{T}} \cdot \underline{x} + c \\ \text{s.t.} \quad & \mathbf{G} \cdot \underline{x} \preceq \underline{h} \\ & \mathbf{A} \cdot \underline{x} = \underline{b} \,, \end{aligned} \tag{C.1}$$

where **Q** is a symmetric matrix and c is a constant. Solving a QP generally is NP-hard [156]. However, if the matrix **Q** is in addition positive definite, a QP becomes a special case of a convex optimization problem and thus, any locally optimal solution to (C.1) is also globally optimal. This optimal solution can be determined in polynomial time for instance by means of an ellipsoid method. For further reading on QP and convex optimization in general see [24].

Bibliography

[1] Fahed Abdallah, Amadou Gning, and Philippe Bonnifait. Box Particle Filtering for Nonlinear State Estimation using Interval Analysis. *Automatica*, 44(3):807–815, March 2008.

[2] Hervé Abdi and Lynne J. Williams. Principal Component Analysis. In *Wiley Interdisciplinary Reviews: Computational Statistics*, volume 2, pages 433–459. Wiley, New York, July 2010.

[3] Simo Ali-Löytty. Efficient Gaussian Mixture Filter for Hybrid Positioning. In *Proceedings of the IEEE/ION Position, Location and Navigation Symposium*, pages 60–66, Monterey, CA, May 2008.

[4] Simo Ali-Löytty. *Gaussian Mixture Filters in Hybrid Positioning*. PhD thesis, Tampere University of Technology, Tampere, Finland, August 2009.

[5] Simo Ali-Löytty and Niilo Sirola. Gaussian Mixture Filter in Hybrid Navigation. In *Proceedings of the European Navigation Conference*, pages 831–837, May 2007.

[6] Simo Ali-Löytty and Niilo Sirola. Gaussian Mixture Filters for Hybrid Positioning. In *Proceedings of the 20th International Technical Meeting of the Satellite Devision of the Institute of Navigation (ION GNSS)*, pages 562–569, Fort Worth, TX, September 2007.

[7] Daniel L. Alspach and Harold W. Sorenson. Nonlinear Bayesian Estimation using Gaussian Sum Approximation. *IEEE Transactions on Automatic Control*, 17(4):439–448, August 1972.

[8] Brian D. O. Anderson and John B. Moore. *Optimal Filtering*. Dover Publications, 2005.

[9] Christophe Andrieu and Arnaud Doucet. Particle filtering for partially observed Gaussian state space models. *Journal of the Royal Statistical Society: Series B*, 64(4):827–836, 2002.

[10] Ienkaran Arasaratnam. *Cubature Kalman Filtering: Theory & Applications*. PhD thesis, McMaster University, April 2009.

[11] Ienkaran Arasaratnam and Simon Haykin. Cubature Kalman Filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, June 2009.

[12] Ienkaran Arasaratnam and Simon Haykin. Cubature Kalman Smoothers. *Automatica*, 47(20):2245–2250, October 2011.

[13] Ienkaran Arasaratnam, Simon Haykin, and Robert J. Elliott. Discrete-Time Nonlinear Filtering Algorithms Using Gauss–Hermite Quadrature. *Proceedings of the IEEE*, 95(5):953–977, 2007.

[14] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[15] S. Bancroft. An Algebraic Solution of the GPS Equations. *IEEE Transactions on Aerospace and Electronic Systems*, AES-21(1):56–59, January 1985.

[16] Richard E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[17] Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck. Instantaneous Pose Estimation using Rotation Vectors. In *Proceedings of the 34th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3413–3416, Taipei, Taiwan, April 2009.

[18] Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck. Semi-Analytic Stochastic Linearization for Range-Based Pose Tracking. In *Proceedings of the 2010 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 44–49, Salt Lake City, Utah, September 2010.

[19] Jürgen Beyerer. *Verfahren zur quantitativen statistischen Bewertung von Zusatzwissen in der Messtechnik*. VDI Fortschritt-Berichte, Reihe 8, Nummer 783, 1999.

[20] Samuel S. Blackman. *Multiple-Target Tracking with Radar Applications.* Norwood, MA: Artech House, 1986.

[21] Hermann Borotschnig, Lucas Paletta, Manfred Prantl, and Axel Pinz. Appearance-Based Active Object Recognition. *Image and Vision Computing,* 18:715–727, 2000.

[22] John E. Bortz. A New Mathematical Formulation for Strapdown Inertial Navigation. *IEEE Transactions on Aerospace and Electronic Systems,* AES-7(1):61–66, January 1971.

[23] Mieczyslaw Borysiewicz, Anna Wawrzynczak, and Piotr Kopka. Bayesian-Based Methods for the Estimation of the Unknown Model's Parameters in the Case of the Localization of the Atmospheric Contamination Source. *Foundations of Computing and Decision Sciences,* 37(4):253–270, 2012.

[24] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[25] Philipp Boyle and Marcus Frean. Dependent Gaussian Processes. In Lawrence K. Saul, Yair Weiss, and Leon Bottou, editors, *Advances in Neural Information Processing Systems 17,* pages 217–224. MIT Press, 2005.

[26] Kai P. Briechle. *Nichtlineare Filterverfahren mit Anwendung auf Lokalisierungsprobleme.* PhD thesis, Technische Universität München, March 2003.

[27] Damiano Brigo and Francois Le Gland. A Finite Dimensional Filter with Exponential Density. In *Proceedings of the 1997 IEEE Conference on Decision and Control (CDC),* volume 2, pages 1643–1644, San Diego, CA, 1997.

[28] Damiano Brigo, Bernard Hanzon, and Francois Le Gland. A Differential Geometric Approach to Nonlinear Filtering: the Projection Filter. Technical Report 2598, Insitut National De Recherche en Informatique et en Automatique, June 1995.

[29] Damiano Brigo, Bernard Hanzon, and Francois Le Gland. Approximate Nonlinear Filtering by Projection on Exponential Manifold of Densities. *Bernoulli,* 5(3):495–534, June 1999.

[30] Vladimir Britanak, Patrick C. Yip, and Kamisetty R. Rao. *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations.* Academic Press, 2006.

[31] Pierrick Bruneau, Marc Gelgon, and Fabien Picarougne. Parsimonious reduction of Gaussian mixture models with a variational-Bayes approach. *Pattern Recognition*, 43:850–858, 2010.

[32] Lucian Buşoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators.* CRC Press, 2010.

[33] Richard S. Bucy and Kenneth D. Senne. Digital synthesis of non-linear filters. *Automatica*, 7(3):287–298, May 1971.

[34] James J. Caffery, Jr. A New Approach to the Geometry of TOA Location. In *Proceedings of 55th IEEE Vehicular Technology Conference*, pages 1942–1949, 2000.

[35] Manfredo Do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice-Hall, Englewood Cliffs, NJ, 1976.

[36] James Carpenter, Peter Clifford, and Paul Fearnhead. Improved particle filter for nonlinear problems. In *IEE Proceedings Radar, Sonar and Navigation*, volume 146, pages 2–7, February 1999.

[37] M. D. Carrascal, M. Puigcerver, and P. Puig. Sensitivity of Gaussian plume model to dispersion specifications. In *Theoretical and Applied Climatology*, volume 48, pages 147–157. Springer, 1993.

[38] Subhash Challa, Yaakov Bar-Shalom, and Vikram Krishnamurthy. Nonlinear Filtering via Generalized Edgeworth Series and Gauss-Hermite Quadrature. *IEEE Transactions on Signal Processing*, 48(6):1816–1820, June 2000.

[39] K. C. Chang and Wei Sun. Scalable Fusion with Mixture Distributions in Sensor Networks. In *Proceedings of the 11th International Conference Control, Automation, Robotics and Vision*, pages 1252–1256, Singapore, December 2010.

[40] Rong Chen and Jun S. Liu. Mixture Kalman Filters. *Journal of the Royal Statistical Society: Series B*, 62(3):493–508, 2000.

[41] Zhe Chen. Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond. Technical report, Adaptive Systems Laboratory, McMaster University, 2003.

[42] Kwok-Wai Cheung and Hing Cheung So. A Multidimensional Scaling Framework for Mobile Location Using Time-of-Arrival Measurements. *IEEE Transactions on Signal Processing*, 53(2):460–470, February 2005.

[43] Martin Clark and Richard Vinter. A New Class of Moment Matching Filters for Nonlinear Tracking and Estimation Problems. In *2006 IEEE Nonlinear Statistical Signal Processing Workshop*, pages 108–112, September 2006.

[44] C. W. Clenshaw. A note on the summation of Chebyshev series. *Mathematical Tables and other Aids to Computation*, 9(51):118–120, 1955.

[45] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.

[46] David F. Crouse, Peter Willett, Krishna Pattipati, and Lennart Svensson. A Look At Gaussian Mixture Reduction Algorithms. In *Proceedings of the 14th International Conference on Information Fusion*, Chicago, IL, July 2011.

[47] Lehel Csató and Manfred Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, March 2002.

[48] Fred Daum and Jim Huang. Curse of Dimensionality and Particle Filters. In *Proceedings of the 2003 IEEE Aerospace Conference*, volume 4, pages 1979–1993, March 2003.

[49] Guido de Croon, Ida G. Sprinkhuizen-Kuyper, and Eric O. Postma. Comparing Active Vision Models. *Image and Vision Computing*, 27:374–384, March 2009.

[50] Frank Deinzer, Joachim Denzler, and Heinrich Niemann. Viewpoint Selection - Planning Optimal Sequences of Views for Object Recognition. In *In International Conference on Computer Vision*, pages 65–73. Springer, 2003.

[51] Marc P. Deisenroth. *Efficient Reinforcement Learning Using Gaussian Processes*. PhD thesis, Karlsruhe Institute of Technology, 2010.

[52] Marc P. Deisenroth, Marco F. Huber, and Uwe D. Hanebeck. Analytic Moment-based Gaussian Process Filtering. In *26th International Conference on Machine Learning (ICML)*, pages 225–232, Montreal, Canada, June 2009.

[53] Marc P. Deisenroth, Ryan Turner, Marco F. Huber, Uwe D. Hanebeck, and Carl E. Rasmussen. Robust Filtering and Smoothing with Gaussian Processes. *IEEE Transactions on Automatic Control*, 57(7):1865–1871, July 2012.

[54] Marc Peter Deisenroth and Henrik Ohlsson. A General Perspective on Gaussian Filtering and Smoothing: Explaining Current and Deriving New Algorithms. In *Proceedings of the American Control Conference 2011*, pages 1807–1812, San Francisco, CA, June 2011.

[55] Joachim Denzler and Christopher M. Brown. Information Theoretic Sensor Data Selection for Active Object Recognition and State Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):145–157, February 2002.

[56] Peter J. Diggle and Paulo J. Ribeiro, Jr. *Model-Based Geostatistics*. Springer Series in Statistics. Springer, 2007.

[57] Pedro Domingos and Michael Pazzani. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29:103–130, 1997.

[58] Arnaud Doucet, Nando de Freitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice*. Statistics for Engineering and Information Science. New York: Springer-Verlag, 2001.

[59] Arnaud Doucet, Simon J. Godsill, and Christophe Andrieu. On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and Computing*, 10:197–208, 2000.

[60] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley & Sons, 2nd edition, 2000.

[61] Yaakov Engel, Shie Mannor, and Ron Meir. The Kernel Recursive Least-Squares Algorithm. *IEEE Transactions on Signal Processing*, 52(8):2275–2285, August 2004.

[62] Friedrich Faubel and Dietrich Klakow. An Adaptive Level of Detail Approach to Nonlinear Estimation. In *Proceedings of the 2010 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3958–3961, 2010.

[63] Friedrich Faubel, John McDonough, and Dietrich Klakow. The Split and Merge Unscented Gaussian Mixture Filter. *IEEE Signal Processing Letters*, 16(9):786–789, September 2009.

[64] Brian Ferris, Dirk Hähnel, and Dieter Fox. Gaussian Processes for Signal Strength-Based Location Estimation. In *Proceedings of Robotics Science and Systems*, 2006.

[65] Wade Foy. Position-Location Solutions by Taylor-Series Estimation. *IEEE Transactions on Aerospace and Electronic Systems*, AES-12(2):187–194, March 1976.

[66] Arthur Gelb. *Applied Optimal Estimation*. MIT Press, 1974.

[67] Agathe Girard, Carl E. Rasmussen, Joaquin Quiñonero-Candela, and Roderick Murray-Smith. Gaussian Process PriorsWith Uncertain Inputs Application to Multiple-Step Ahead Time Series Forecasting. In *Advances in Neural Information Processing Systems 15*, 2003.

[68] Agathe Girard, Carl Edward Rasmussen, and Roderick Murray-Smith. Gaussian Process priors with Uncertain Inputs: Multiple-Step-Ahead Prediction. Technical Report TR-2002-119, University of Glasgow, Department of Computing Science, October 2002.

[69] Jacob Goldberger and Sam Roweis. Hierarchical Clustering of a Mixture Model. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 505–512, 2005.

[70] Neil J. Gordon, David J. Salmond, and Adrian F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings on Radar and Signal Processing*, 140(2):107–113, April 1993.

[71] Karl Granström. *Extended target tracking using PHD filters*. PhD thesis, Linköping University, 2012.

[72] Alexander G. Gray and Andrew W. Moore. ‘N-Body’ Problems in Statistical Learning. In *Advances in Neural Information Processing Systems 13*, 2001.

[73] Uwe D. Hanebeck. *Nonlinear Methods for State Estimation in Stochastic Dynamical Systems – A Concise Introduction*. Habilitation treatise, Technische Universität München, 2002.

[74] Uwe D. Hanebeck. PGF 42: Progressive Gaussian Filtering with a Twist. In *Proceedings of the 16th International Conference on Information Fusion*, Istanbul, Turkey, July 2013.

[75] Uwe D. Hanebeck, Kai Briechle, and Andreas Rauh. Progressive Bayes: A New Framework for Nonlinear State Estimation. In *Proceedings of SPIE, AeroSense Symposium*, volume 5099, pages 256–267, Orlando, Florida, May 2003.

[76] Uwe D. Hanebeck and Olga Feiermann. Progressive Bayesian Estimation for Nonlinear Discrete-Time Systems: The Filter Step for Scalar Measurements and Multidimensional States. In *Proceedings of the 2003 IEEE Conference on Decision and Control*, pages 5366–5371, Maui, Hawaii, USA, December 2003.

[77] Uwe D. Hanebeck and Günther Schmidt. Closed-Form Elliptic Location with an Arbitrary Array Topology. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3070–3073, 1996.

[78] Steven Hanna, Joseph Chang, and Helge R. Olesen. *Indianapolis Tracer Data and Meteorological Data*, May 1997.

[79] Bill Hirst, Philip Jonathan, Fernando G. del Cueto, David Randell, and Oliver Kosut. Locating and quantifying gas emission sources using remotely obtained concentration data. *Atmospheric Environment*, 74:141–158, August 2013.

[80] Ekkehard Holzbecher. *Environmental Modeling*. Springer, 2nd edition, 2012.

[81] Frédéric Hourdin and Olivier Talagrand. Eulerian backtracking of atmospheric tracers. I: Adjoint derivation and parametrization of subgrid-scale transport. *Quarterly Journal of the Royal Meteorological Society*, 132(615):567–583, January 2006.

[82] Marco Huber. *Probabilistic Framework for Sensor Management*. PhD thesis, Universität Karlsruhe (TH), April 2009.

[83] Marco Huber, Dietrich Brunn, and Uwe D. Hanebeck. Closed-Form Prediction of Nonlinear Dynamic Systems by Means of Gaussian Mixture Approximation of the Transition Density. In *Proceedings of the 2006 IEEE*

International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pages 98–103, Heidelberg, Germany, September 2006.

[84] Marco F. Huber. Recursive Gaussian Process Regression. In *Proceedings of the 38th International Conference on Acoustics, Sound, and Signal Processing (ICASSP)*, pages 3362–3366, Vancouver, BC, Canada, May 2013.

[85] Marco F. Huber, Tim Bailey, Hugh Durrant-Whyte, and Uwe D. Hanebeck. On Entropy Approximation for Gaussian Mixture Random Vectors. In *Proceedings of the 2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 181–188, Seoul, Republic of Korea, August 2008.

[86] Marco F. Huber, Dietrich Brunn, and Uwe D. Hanebeck. Efficient Nonlinear Measurement Updating based on Gaussian Mixture Approximation of Conditional Densities. In *Proceedings of the 2007 American Control Conference (ACC)*, pages 4425–4430, New York, New York, July 2007.

[87] Marco F. Huber, Tobias Dencker, Masoud Roschani, and Jürgen Beyerer. Bayesian Active Object Recognition via Gaussian Process Regression. In *Proceedings of the 15th International Conference on Information Fusion (Fusion)*, July 2012.

[88] Marco F. Huber and Uwe D. Hanebeck. Gaussian Filter based on Deterministic Sampling for High Quality Nonlinear Estimation. In *Proceedings of the 17th IFAC World Congress*, pages 13527–13532, Seoul, Republic of Korea, July 2008.

[89] Marco F. Huber and Uwe D. Hanebeck. Progressive Gaussian Mixture Reduction. In *Proceedings of the 11th International Conference on Information Fusion (Fusion)*, Cologne, Germany, July 2008.

[90] Kazufumi Ito and Kaiqi Xiong. Gaussian Filters for Nonlinear Filtering Problems. *IEEE Transactions on Automatic Control*, 45(5):910–927, May 2000.

[91] Edwin T. Jaynes. *Probability Theory*. Cambridge University Press, 2003.

[92] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*. Dover Publications, Inc., 2007.

[93] Simon Julier, Jeffrey Uhlmann, and Hugh F. Durrant-Whyte. A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, 2000.

[94] Simon J. Julier and Jeffrey K. Uhlmann. A New Extension of the Kalman Filter to Nonlinear Systems. In *International Symposium on Aerospace/Defence Sensing, Simulation and Control*, 1997.

[95] Simon J. Julier and Jeffrey K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[96] Rudolf E. Kalman. A new Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME, Journal of Basic Engineering*, 82 (Series D)(1):35–45, 1960.

[97] Raymond Kan. From Moments of Sum to Moments of Product. *Journal of Multivariate Analysis*, 99(3):542–554, March 2008.

[98] Elliott D. Kaplan and Christopher Hegarty. *Understanding GPS: Principles and Applications*. Artech House, 2nd edition, 2005.

[99] Uwe Kiencke and Ralf Eger. *Meßtechnik*. Springer, 6th edition, 2001.

[100] Kiseon Kim and Georgy Shevlyakov. Why Gaussianity? *IEEE Signal Processing Magazine*, pages 102–113, March 2008.

[101] Genshiro Kitagawa. Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[102] Vesa Klumpp, Frederik Beutler, Uwe D. Hanebeck, and Dietrich Fränken. The Sliced Gaussian Mixture Filter with Adaptive State Decomposition Depending on Linearization Error. In *Proceedings of the 13th International Conference on Information Fusion*, Edinburgh, United Kingdom, July 2010.

[103] Jonathan Ko and Dieter Fox. GP-BayesFilters: Bayesian Filtering using Gaussian Process Prediction and Observation Models. *Autonomous Robots*, 27(1):75–90, 2009.

[104] Jonathan Ko, Daniel J. Klein, Dieter Fox, and Dirk Haehnel. GP-UKF: Unscented Kalman Filters with Gaussian Process Prediction and Observation Models. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1901–1907, San Diego, CA, October-November 2007.

[105] Jayesh H. Kotecha and Petar M. Djurić. Gaussian Particle Filtering. *IEEE Transactions on Signal Processing*, 51(10):2592–2601, 2003.

[106] Jayesh H. Kotecha and Petar M. Djurić. Gaussian Sum Particle Filtering. *IEEE Transactions on Signal Processing*, 51(10):2602–2612, 2003.

[107] Edgar Kraft. A Quaternion-Based Unscented Kalman Filter for Orientation Tracking. In *Proceedings of the Sixth International Conference of Information Fusion*, volume 1, pages 47–54, 2003.

[108] Stuart C. Kramer and Harold W. Sorenson. Recursive Bayesian estimation using piece-wise constant approximations. *Automatica*, 24(6):789–801, November 1988.

[109] Andreas Krause and Carlos Guestrin. Nonmyopic Active Learning of Gaussian Processes: An Exploration-Exploitation Approach. In *Proceedings of the 24 th International Conference on Machine Learning*, pages 449–456, Corvallis, OR, 2007.

[110] Peter Krauthausen. *Learning Dynamic Systems for Intention Recognition in Human-Robot-Cooperation.* PhD thesis, Karlsruhe Institute of Technology (KIT), 2012.

[111] Catherine Laporte and Tal Arbel. Efficient Discriminant Viewpoint Selection for Active Bayesian Recognition. *International Journal of Computer Vision*, 68:267–287, July 2006.

[112] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter. Comments on "A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators". *IEEE Transactions on Automatic Control*, 45(8):1406–1408, 2002.

[113] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter. *Nonlinear Kalman Filtering for Force-Controlled Robot Tasks.* Springer Berlin, 2005.

[114] Jun S. Liu and Rong Chen. Blind Deconvolution via Sequential Imputations. *Journal of the American Statistical Association*, 90(430):567–576, June 1995.

[115] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the 7th International Conference on Computer Vision*, volume 2, pages 1150–1157, Kerkyra, Greece, September 1999.

[116] Mihai Bogdan Luca, Stéphane Azou, Gilles Burel, and Alexandru Serbanescu. On Exact Kalman Filtering of Polynomial Systems. *IEEE Transactions on Circuits and Systems—I: Regular Papers*, 53(6):1329–1340, June 2006.

[117] David J. C. MacKay. Comparison of Approximate Methods for Handling Hyperparameters. *Neural Computation*, 11(5):1035–1068, 1999.

[118] Dimitris E. Manolakis. Efficient Solution and Performance Analysis of 3-D Position Estimation by Trilateration. *IEEE Transactions on Aerospace and Electronic Systems*, 32(4):1239–1248, October 1996.

[119] John C. Mason and David C. Handscomb. *Chebyshev Polynomials*. Chapman & Hall/CRC, 2003.

[120] Peter Maybeck. *Stochastic Models, Estimation, and Control, Volume 2*. Academic Press, 1982.

[121] Vladimir Maz'ya and Gunther Schmidt. On approximate approximations using gaussian kernels. *IMA Journal of Numerical Analysis*, 16:13–29, 1996.

[122] Leonard A. McGee and Stanley F. Schmidt. Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry. Technical memorandum 86847, NASA, 1985.

[123] H. S. Migon and P. J. Harrison. An application of non-linear Bayesian forecasting to television advertising. In J. M. Bernardo, M. H DeGroot, D. V. Lindley, and A. F. M. Smith, editors, *Bayesian Statistics 2*. Valencia University Press, 1985.

[124] Javier G. Monroya, Achim J. Lilienthalc, Jose-Luis Blancob, Javier Gonzalez-Jimeneza, and Marco Trincavelli. Probabilistic Gas Quantification with MOX Sensors in Open Sampling Systems - A Gaussian Process Approach. *Sensors and Actuators B: Chemical*, 188:298–312, November 2013.

[125] Mark R. Morelande and Bill Moran. An Unscented Transformation for Conditionally Linear Models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages III–1417–III–1420, April 2007.

[126] Hiroshi Murase and Shree K. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal Computer Vision*, 14:5–24, January 1995.

[127] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective.* MIT Press, 2012.

[128] Christian Musso, Nadia Oudjane, and Francois Le Gland. Improving Regularised Particle Filters. In A. Doucet, N. de Freitas, and N. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, chapter 12. New York: Springer Verlag, 2001.

[129] A. Nemra and N. Aouf. Robust INS/GPS Sensor Fusion for UAV Localization Using SDRE Nonlinear Filtering. *IEEE Sensors Journal*, 10(4):789–798, April 2010.

[130] Duy Nguyen-Tuong and Jan Peters. Local Gaussian Processes Regression for Real-time Model-based Robot Control. In *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 380–385, 2008.

[131] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. Real-Time Local GP Model Learning. In Olivier Sigaud and Jan Peters, editors, *From Motor Learning to Interaction Learning in Robots*, volume 264 of *Studies in Computational Intelligence*, pages 193–207. Springer-Verlag, 2010.

[132] Daniel Nikovski and Matthew Brand. Non-Linear Stochastic Control in Continuous State Spaces by Exact Integration in Bellman's Equations. In *Proceedings of the 2003 International Conference on Automated Planning and Scheduling*, pages 91–95, 2003.

[133] Magnus Nørgaard, Niels K. Poulsen, and Ole Ravn. New Developments in State Estimation for Nonlinear Systems. *Automatica*, 36(11):1627–1638, November 2000.

[134] Lucas Paletta and Axel Pinz. Active Object Recognition By View Integration and Reinforcement Learning. *Robotics and Autonomous Systems*, 31:71–86, 2000.

[135] Lucy Y. Pao. Multisensor Multitarget Mixture Reduction Algorithms for Tracking. *AIAA Journal of Guidance, Control and Dynamics*, 17:1205–1211, 1994.

[136] Kalyanapuram R. Parthasarathy. *Probability Measures on Metric Spaces.* American Mathematical Society, new edition, 2005.

[137] Frank Pasquill. The estimation of the dispersion of windborne material. *The Meteorological Magazine*, 90(1063):33–49, 1961.

[138] Fernando Pérez-Cruz, Steven Van Vaerenbergh, Juan José Murillo-Fuentes, Miguel Lázao-Gredilla, and Ignacio Santamaría. Gaussian Processes for Nonlinear Signal Processing. *IEEE Signal Processing Magazine*, 30(4):40–50, July 2013.

[139] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*, chapter 17: Integration of Ordinary Differential Equations. Cambridge University Press, 3rd edition, 2007.

[140] Joaquin Quiñonero-Candela and Carl E. Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

[141] Ananth Ranganathan, Ming-Hsuan Yang, and Jeffrey Ho. Online Sparse Gaussian Process Regression and Its Applications. *IEEE Transactions on Image Processing*, 20(2):391–404, February 2011.

[142] K. Shankar Rao. Source estimation methods for atmospheric dispersion. *Atmospheric Environment*, 41:6964–6973, 2007.

[143] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[144] H. E. Rauch, F. Tung, and C. T. Striebel. Maximum Likelihood Estimates of Linear Dynamic Systems. *AIAA Journal*, 3(8):1445–1450, August 1965.

[145] Andreas Rauh, Kai Briechle, and Uwe D. Hanebeck. Nonlinear Measurement Update and Prediction: Prior Density Splitting Mixture Estimator. In *Proceedings of the 2009 IEEE International Conference on Control Applications (CCA)*, July 2009.

[146] Andreas Rauh and Uwe D. Hanebeck. Calculating Moments of Exponential Densities Using Differential Algebraic Equations. *IEEE Signal Processing Letters*, 10(5):144–147, May 2003.

[147] Andreas Rauh and Uwe D. Hanebeck. Moment-Based Prediction Step for Nonlinear Discrete-Time Dynamic Systems Using Exponential Densities. In *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference*, pages 1923–1928, Sevilla, Spain, December 2005.

[148] Steven Reece and Stephen Roberts. An Introduction to Gaussian Processes for the Kalman Filter Expert. In *Proceedings of the 13th International Conference on Information Fusion*, Edinburgh, UK, 2010.

[149] Robert H. Risch. The problem of integration in finite terms. *Transactions of the American Mathematical Society*, 139:167–189, May 1969.

[150] Robert H. Risch. The solution of the problem of integration in finite terms. *Bulletin of the American Mathematical Society*, 76:605–608, 1970.

[151] Theodore Rivlin. *Chebyshev Polynomials*. New York: Wiley, 1990.

[152] Patrick Rößler, Frederik Beutler, Uwe D. Hanebeck, and Norbert Nitzsche. Motion Compression Applied to Guidance of a Mobile Teleoperator. In *Proceedings of the 2005 IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2495–2500, 2005.

[153] Sumantra Dutta Roy, Santanu Chaudhury, and Subhashis Banerjee. Active recognition through next view planning: a survey. *Pattern Recognition*, 37(3):429–446, March 2004.

[154] Alison Rudd, Alan G. Robins, Jason J. Lepley, and Stephen E. Belcher. An Inverse Method for Determining Source Characteristics for Emergency Response Applications. *Boundary-Layer Meteorology*, 144(1):1–20, July 2012.

[155] Andrew R. Runnalls. Kullback-Leibler Approach to Gaussian Mixture Reduction. *IEEE Transactions on Aerospace and Electronic Systems*, 43(3):989–999, July 2007.

[156] Sartaj Sahni. Computationally Related Problems. *SIAM Journal on Computing*, 3(4):262–279, 1974.

[157] David J. Salmond. Mixture reduction algorithms for target tracking in clutter. In *Proceedings of SPIE Signal and Data Processing of Small Targets*, volume 1305, pages 434–445, October 1990.

[158] Simo Särkkä. Unscented rauch-tung-striebel smoother. *IEEE Transactions on Automatic Control*, 53(3):845–849, 2008.

[159] Simo Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[160] Juha Sarmavuori and Simo Särkkä. Fourier-Hermite Kalman Filter. *IEEE Transactions on Automatic Control*, 57(6):1511–1515, June 2012.

[161] Tor S. Schei. A finite difference method for linearizing in nonlinear estimation algorithms. *Automatica*, 33(11):2051–2058, November 1997.

[162] Dennis Schieferdecker and Marco F. Huber. Gaussian Mixture Reduction via Clustering. In *Proceedings of the 12th International Conference on Information Fusion (Fusion)*, pages 1536–1543, Seattle, Washington, July 2009.

[163] Bernhard Schölkopf and Alexander Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive computation and machine learning series. MIT Press, Cambridge, Massachusetts, 2002.

[164] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[165] Thomas Schön, Fredrik Gustafsson, and Per-Johan Nordlund. Marginalized Particle Filters for Mixed Linear/Nonlinear State-Space Models. *IEEE Transactions on Signal Processing*, 53(7):2279–2287, July 2005.

[166] Fred C. Schweppe. *Uncertain Dynamic Systems*. Prentice–Hall, 1973.

[167] David W. Scott and William F. Szewczyk. From Kernels to Mixtures. *Technometrics*, 43(3):323–335, 2001.

[168] Fernando Seco, Antonio R. Jiménez, Carlos Prieto, Javier Roa, and Katerina Koutsou. A Survey of Mathematical Methods for Indoor Localization. In *IEEE International Symposium on Intelligent Signal Processing*, pages 9–14, August 2009.

[169] Inanc Senocak, Nicolas W. Hengartner, Margaret B. Short, and W. Brent Daniel. Stochastic Event Reconstruction of Atmospheric Contaminant Dispersion Using Bayesian Inference. *Atmospheric Environment*, 42(33):7718–7727, October 2008.

[170] Miroslav Simandl and Jindrich Duník. Sigma point gaussian sum filter design using square root unscented filters. In *Proceedings of the 16th IFAC World Congress*, Prague, Czech Republic, July 2005.

[171] Miroslav Simandl and Jindrich Duník. Design of derivative-free smoothers and predictors. In *Proceedings of the 14th IFAC Symposium on System Identification*, pages 1240–1245, Australia, 2006.

[172] Miroslav Simandl, Jakub Královeca, and Torsten Söderström. Advanced point mass method for nonlinear state estimation. *Automatica*, 42(7):1133–1145, July 2006.

[173] Dan Simon. *Optimal State Estimation: Kalman, H-Infinity, and Nonlinear Approaches*. Wiley & Sons, 1st edition, 2006.

[174] John Skilling. Bayesian Numerical Analysis. In W. T. Grandy, Jr. and P. W. Milonni, editors, *Physics and Probability*, pages 207–222. Cambridge University Press, 1993.

[175] Alex J. Smola and Peter Bartlett. Sparse Greedy Gaussian Process Regression. In *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001.

[176] Ed Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1259–1266. The MIT Press, 2006.

[177] Edward Snelson, Carl Edward Rasmussen, and Zoubin Ghahramani. Warped Gaussian Processes. In *Advances in Neural Information Processing Systems 16*. The MIT Press, 2004.

[178] Michael D. Sohn, Pamela Reynolds, Navtej Singh, and Ashok J. Gadgil. Rapidly Locating and Characterizing Pollutant Releases in Buildings. *Journal of the Air & Waste Management Association*, 52(12):1422–1432, 2002.

[179] Francisco J. Solis and Roger J-B. Wets. Minimization by Random Search Techniques. *Mathematics of Operations Research*, 6(1):19–30, February 1981.

[180] Harold W. Sorenson and Daniel L. Alspach. Recursive bayesian estimation using gaussian sums. *Automatic*, 7(4):465–479, July 1971.

[181] Harold W. Sorenson and Allen R. Stubberud. Non-linear filtering by approximation of the a posteriori density. *International Journal of Control*, 8(1):33–51, 1968.

[182] Krishnaswamy Srinivasan. State Estimation by Orthogonal Expansion of Probability Distributions. *IEEE Transactions on Automatic Control*, 15(1):3–10, February 1970.

[183] Rajan Srinivasan. *Importance Sampling: Applications in Communications and Detection.* Springer, 2002.

[184] John M. Stockie. The Mathematics of Atmospheric Dispersion Modelling. *SIAM Review*, 53(2):349–372, 2011.

[185] Lawrence D. Stone, Roy L. Streit, Thomas L. Corwin, and Kristine L. Bell. *Bayesian Multiple Target Tracking.* Artech House, 2nd edition, 2014.

[186] Volker Strassen. Gaussian Elimination is not Optimal. *Numerische Mathematik*, 13(4):354–356, August 1969.

[187] Jürgen Sturm. *Approaches to Probabilistic Model Learning for Mobile Manipulation Robots.* PhD thesis, Albert-Ludwigs-Universität Freiburg im Breisgau, 2011.

[188] Hsi Guang Sung. *Gaussian Mixture Regression and Classification.* PhD thesis, Rice University, May 2004.

[189] Richard Szeliski. *Computer Vision: Algorithms and Applications*, chapter 14 – Recognition. Springer London, 2010.

[190] Nassim N. Taleb. *The Black Swan: The Impact of the Highly Improbable.* Random House Trade, 2nd edition, 2010.

[191] Wing Ip Tam, K.N. Plataniotis, and D. Hatzinakos. An adaptive Gaussian sum algorithm for radar tracking. *Signal Processing*, 77:85–104, 1999.

[192] D. Tenne and T. Singh. The Higher Order Unscented Filter. In *Proceedings of the American Control Conference*, pages 2441–2446, June 2003.

[193] Gabriel Terejanu, Puneet Singla, Tarunraj Singh, and Peter D. Scott. A Novel Gaussian Sum Filter Method for Accurate Solution to Nonlinear Filtering Problem. In *Proceedings of the 11th International Conference on Information Fusion*, 2008.

[194] Gabriel Terejanu, Puneet Singla, Tarunraj Singh, and Peter D. Scott. Uncertainty Propagation for Nonlinear Dynamic Systems Using Gaussian Mixture Models. *Journal of Guidance, Control, and Dynamics*, 31(6):1623–1633, November–December 2008.

[195] Federico Thomas and Lluís Ros. Revisiting trilateration for robot localization. *IEEE Transactions on Robotics*, 21:93–101, 2005.

[196] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[197] Petr Tichavský, Carlos H. Muravchik, and Arye Nehorai. Posterior Cramér-Rao Bounds for Discrete-Time Nonlinear Filtering. *IEEE Transactions on Signal Processing*, 46(5):1386–1396, May 1998.

[198] Volker Tresp. A Bayesian Committee Machine. *Neural Computation*, 12(11):2719–2741, 2000.

[199] Raquel Urtasun and Trevor Darrell. Discriminative Gaussian Process Latent Variable Model for Classification. In *Proceedings ot the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

[200] Rudolph van der Merwe, Arnaud Doucet, Nando de Freitas, and Eric Wan. The Unscented Particle Filter. In *Advances in Neural Information Processing Systems*, volume 12, pages 666–672. MIT Press, 2000.

[201] Nuno Vasconcelos. Image Indexing with Mixture Hierarchies. In *Proceedings of the IEEE Conference in Computer Vision and Pattern Recognition*, 2001.

[202] Shrihari Vasudevan, Fabio Ramos, Eric Nettleton, and Hugh Durrant-Whyte. Gaussian Process Modeling of Large-Scale Terrain. *Journal of Field Robotics*, 26(10):812–840, 2009.

[203] Tom Vercauteren and Xiaodong Wang. Decentralized Sigma-Point Information Filters for Target Tracking in Collaborative Sensor Networks. *IEEE Transactions on Signal Processing*, 53(8):2997–3009, August 2005.

[204] Grace Wahba, Xiwu Lin, Fangyu Gao, Dong Xiang, Ronald Klein, and Barbara Klein. The Bias-Variance Tradeoff and the Randomized GACV. In *Advances in Neural Information Processing Systems 11*, pages 620–626. MIT Press, 1999.

[205] Eric A. Wan and Rudolph van der Merwe. The Unscented Kalman Filter. In Simon Haykin, editor, *Kalman Filtering and Neural Networks*, chapter The Unscented Kalman Filter, pages 221–280. John Wiley & Sons, Inc., 2001.

[206] Greg Welch, B. Danette Allen, Adrian Ilie, and Gary Bishop. Measurement Sample Time Optimization for Human Motion Tracking/Capture Systems. In *Proceedings of Trends and Issues in Tracking for Virtual Environments, Workshop at the IEEE Virtual Reality 2007 Conference*, 2007.

[207] Mike West. Approximating Posterior Distributions by Mixtures. *Journal of the Royal Statistical Society: Series B*, 55(2):409–422, 1993.

[208] Mike West and Jeff Harrison. *Bayesian Forecasting and Dynamic Models*, chapter 14: Exponential Family Dynamic Models, pages 534–555. Springer, 1997.

[209] Jason L. Williams. *Information Theoretic Sensor Management*. PhD thesis, Massachusetts Institute of Technology, February 2007.

[210] Jason L. Williams and Peter S. Maybeck. Cost-Function-Based Hypothesis Control Techniques for Multiple Hypothesis Tracking. In *Proceedings of SPIE Signal and Data Processing of Small Targets*, volume 5428, April, 2004.

[211] W. Murray Wonham. Some Applications of Stochastic Differential Equations to Optimal Nonlinear Filtering. *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, 2(3):347–369, 1964.

[212] Sally A. Wood, Wenxin Jian, and Martin Tanner. Bayesian mixture of splines for spatially adaptive nonparameteric regression. *Biometrika*, 89(3):513–528, 2002.

[213] Yuanxin Wu, Dewen Hu, Meiping Wu, and Xiaoping Hu. A Numerical-Integration Perspective on Gaussian Filters. *IEEE Transactions on Signal Processing*, 54(8):2910–2921, August 2006.

[214] Renato Zanetti. Recursive Update Filtering for Nonlinear Estimation. *IEEE Transactions on Automatic Control*, 57(6):1481–1490, June 2012.

[215] Arnold Zellner. Optimal Information Processing and Bayes's Theorem. *The American Statistician*, 42(4):278–280, 1988.

[216] Yong Zhang and Li Wang. Particle Filtering Method for Source Localization in Wireless Sensor Network. In *Advanced Technology in Teaching: Selected papers from the 2012 International Conference on Teaching and Computational Science (ICTCS 2012)*, volume 163, pages 517–523. Springer, 2013.

Part II

Publications

Paper A

Gaussian Filtering using State Decomposition Methods

Authors: Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck

Edited version of the paper:

F. Beutler, M. F. Huber, and U. D. Hanebeck. Gaussian Filtering using State Decomposition Methods. In *Proceedings of the 12th International Conference on Information Fusion (Fusion)*, pages 579–586, Seattle, WA, USA, July 2009.

Gaussian Filtering using State Decomposition Methods

Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck

Intelligent Sensor-Actuator-Systems Laboratory (ISAS)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{beutler|marco.huber|uwe.hanebeck}@ieee.org

Abstract

State estimation for nonlinear systems generally requires approximations of the system or the probability densities, as the occurring prediction and filtering equations cannot be solved in closed form. For instance, Linear Regression Kalman Filters like the Unscented Kalman Filter or the considered *Gaussian Filter* propagate a small set of sample points through the system to approximate the posterior mean and covariance matrix. To reduce the number of sample points, special structures of the system and measurement equation can be taken into account. In this paper, two principles of system decomposition are considered and applied to the Gaussian Filter. One principle exploits that only a part of the state vector is directly observed by the measurement. The second principle separates the system equations into linear and nonlinear parts in order to merely approximate the nonlinear part of the state. The benefits of both decompositions are demonstrated on a real-world example.

1 Introduction

Estimation is applied to problems, where the state of a dynamical system has to be calculated based on noisy observations and where uncertainties in modeling have to be taken into account. In case of linear systems with additive Gaussian noise, the well-known Kalman filter provides optimal estimates of the system state in form of first and second moments. All required calculations can be performed in closed form. The same is not true when nonlinearities arise in the measurement and system equation. Here, approximation has to be applied, where two different approximation approaches can be found in literature. The first one relies on approximating the system equations as done by the extended Kalman filter. In the second approach, the probability density representing the state estimate is approximated instead. Here, sample-based approaches like particle filters [1] are common. A drawback of particle filters is that sampling is performed randomly and thus, many particles have to be used to achieve accurate results. Moreover, particle filters suffer from the *curse of dimensionality*, i.e., the number of particles increases exponentially with the number of dimensions [4].

Another class of sample-based estimators are *linear regression Kalman filters* (LRKFs), like the well-known unscented Kalman filter (UKF) [6], the divided difference filter [10], the central difference filter [13], or the Gaussian filter [5]. Here, it is assumed that the state estimate can be sufficiently characterized by its first two moments, i.e., mean and covariance matrix. The regression points (sample points) for exactly capturing these moments can be easily determined in a deterministic fashion. Propagating the regression points through the nonlinear system equations and calculating the first two moments of the posterior estimate implicitly linearizes the nonlinear equations. For this procedure, only a small number of regression points is required and the number of regression points grows only linearly with the dimension of the state space.

In this paper, the latest LRKF, the so-called *Gaussian filter* introduced in [5], is extended in such a way that the number of regression points can be reduced without a noticeable effect on the estimation quality. For this purpose, two different approaches for state decomposition are discussed. The first one relies on Rao-Blackwellization [14], which is often applied to particle filters for attenuating the curse of dimensionality and which has been firstly applied to the unscaled version of the UKF in [9]. Here, the equations are separated into a linear and nonlinear substructure, where only the nonlinear part is processed in an approximate fashion.

The second decomposition is described in [8] for the Kalman filter. Transferred to LRKFs, only the directly observed state needs to be represented by means of regression points and updated with the current measurement. Based on the correlation between the observed and the indirectly observed part, the indirectly observed state can be updated after the filter step without further approximations.

Although both approaches are derived for the Gaussian Filter, they can be directly applied to any estimator belonging to the class of LRKFs.

In Section 2, a brief problem formulation is given. The considered Gaussian filter is explained in Section 3. Both decomposition approaches are content of Section 4, while in Section 5, the estimator equations for both decompositions are derived. In Section 6, the example application employed for simulations and experiments is introduced. Here, tracking an object based on reference signals for a large-scale telepresence scenario as described in [12] is considered. Furthermore, a mathematical formulation of the considered example is given. In Section 7, the decomposed case and the full state case are compared in a simulation and an experiment. The paper closes with conclusions.

2 Problem Formulation

A nonlinear discrete-time dynamic system is given by

$$\begin{aligned} \underline{\boldsymbol{x}}_{k+1} &= \underline{a}_k\left(\underline{\boldsymbol{x}}_k, \underline{u}_k, \underline{\boldsymbol{w}}_k\right) \ , \\ \underline{\boldsymbol{y}}_k &= \underline{h}_k\left(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{v}}_k\right) \ , \end{aligned}$$

where the functions $\underline{a}_k(\cdot)$ and $\underline{h}_k(\cdot)$ are known. The vector $\underline{\boldsymbol{x}}_k$ is the state of the system, $\underline{\boldsymbol{y}}_k$ is the measurement vector, and $\underline{u}_k$ is a known system input at the discrete time k. The measurement and process noise are characterized by $\underline{\boldsymbol{v}}_k$ and $\underline{\boldsymbol{w}}_k$, respectively. Based on the measurements $\underline{\boldsymbol{y}}_k$, the density of the system state $\underline{\boldsymbol{x}}_k$ has to be estimated by using filtering and prediction techniques.

Filtering and prediction for nonlinear systems typically cannot be realized in closed form. To achieve an efficient estimation scheme, the exact density or the system equations have to be approximated in an adequate manner. If the approximation is based on the density, the number of parameters, i.e., the number of regression or sample points[1], for approximating the state density has to be constant for efficient state estimation. Furthermore, the approximation quality should be adjustable in order to improve the estimation accuracy if required.

In many cases, however, parts of the estimation problem can be solved in closed form. Hence, only a part of the density has to be represented by a set of re-

1 Throughout this paper, the terms *regression points* and *sample points* are used interchangeably.

gression points. By exploiting the structure of the underlying system, several decomposition methods can be applied for this purpose.

3 The Gaussian Filter (GF)

In contrast to most of the LRKFs like the famous UKF [6], the Gaussian filter [5] considered in this paper allows varying the number of sample points. In doing so, more information of the nonlinear system model is captured by increasing the number of sample points and thus, one can trade estimation quality for computational demand. Moreover, information of higher-order moments can be explicitly incorporated into the estimation process. But still, as typical for LRKFs, the number of regression points grows only linearly with the dimension of the state space.

The determination of the sample points for the Gaussian filter relies on efficiently solving an optimization problem, where a certain distance measure between the Gaussian density representing the state estimate and the Dirac mixture density representing the sample points is minimized. An additional constraint ensures that mean and covariance matrix of the state are captured exactly. In order to calculate the parameters, i.e., the weights and the positions of the regression points, a two step procedure is employed.

In the computationally demanding first step, which can be performed off-line, the samples for an univariate standard Gaussian density are calculated [5]. In this paper, the number of sample points D is assumed to be odd, because in this case one sample point is always located at the mean. In Table 1, the sample points with negative positions for various numbers of samples D are given. The mean point as well as the positive positions are calculated according to

$$\mu_{\frac{D+1}{2}} = 0 \,, \quad \mu_i = -\mu_{D+1-i} \quad \text{for } i = \tfrac{D+3}{2}, \ldots, D \,.$$

In the second step, for on-line approximating an arbitrary N-dimensional Gaussian density $f\left(\underline{x}\right) = \mathcal{N}\left(\underline{x} - \underline{\mu}^x, \mathbf{C}^{x,x}\right)$, N sets of D sample points are placed along the N coordinate axes. By means of affine operations, these sample points are transformed for exactly capturing the mean $\underline{\mu}^x$ and the covariance matrix $\mathbf{C}^{x,x}$. The transformed sample points are calculated via

$$\underline{\mu}_i^x = \underline{\mu}^x + \mathbf{V} \cdot \sqrt{\mathbf{D}} \cdot \underline{S}_i \quad \text{for } i = 1, \ldots, L \,, \tag{1}$$

Table 1: Sample positions for several numbers of samples.

D	μ_1	μ_2	μ_3
3	-1.2247	-	-
5	-1.4795	-0.5578	-
7	-1.6346	-0.8275	-0.3788

where **V** and **D** is the matrix of eigenvectors and diagonal matrix of eigenvalues, respectively, with $\mathbf{C}^{x,x} = \mathbf{V} \cdot \mathbf{D} \cdot (\mathbf{V})^{\mathrm{T}}$. Furthermore, $L = N \cdot (D-1) + 1$ is the total number of sample points, $\underline{S}_i$ is the i-th column of the matrix

$$\mathbf{S} = \left[\underbrace{\underline{1} \cdot \mu_{\frac{D+1}{2}}}_{=\underline{0}} \quad \mathbf{I}_{N,N} \otimes \left[\mu_1, \ldots, \mu_{\frac{D-1}{2}}, \mu_{\frac{D+3}{2}}, \ldots, \mu_D \right] \right]$$

and $\otimes$ is the Kronecker product.

For calculating the mean $\underline{\mu}^x$ by means of set of sample points $\underline{\mu}_i^x$, the same sample weight

$$\omega = \tfrac{1}{L} \tag{2}$$

is used for all sample points. A different weight is required for ensuring that the sample covariance matrix coincides with given covariance matrix $\mathbf{C}^{x,x}$. Here, the weight $\omega_s = 1/D$ has to be used instead of ω, (2). This is shown in the following.

PROOF To prove that the weight for calculating the sample covariance matrix has to be different, the sample covariance matrix for the N-dimensional Gaussian density is considered

$$\sum_{i=1}^{L} \omega_s \cdot \left(\underline{\mu}^x - \underline{\mu}_i^x \right) \cdot \left(\underline{\mu}^x - \underline{\mu}_i^x \right)^{\mathrm{T}} \equiv \mathbf{C}^{x,x} \quad .$$

Using (1) results in

$$\mathbf{C}^{x,x} = \sum_{i=1}^{L} \omega_s \cdot \left(\mathbf{V} \cdot \sqrt{\mathbf{D}} \cdot \underline{S}_i \right) \cdot \left(\mathbf{V} \cdot \sqrt{\mathbf{D}} \cdot \underline{S}_i \right)^{\mathrm{T}}$$

$$= \left(\mathbf{V}\sqrt{\mathbf{D}}\right)\left(\sum_{i=1}^{L}\omega_s \cdot \underbrace{\underline{S}_i \cdot \left(\underline{S}_i\right)^{\mathrm{T}}}_{=\left(\mu_i^x\right)^2 \cdot \mathbf{I}_{N,N}}\right)\left(\mathbf{V}\sqrt{\mathbf{D}}\right)^{\mathrm{T}} .$$

The sum has to be equal to one so that the sample covariance matrix corresponds to the given covariance matrix. Furthermore, the sample points are point-symmetric regarding to the mean of the Gaussian density, which results in

$$\sum_{i=1}^{L}\omega_s \cdot \left(\mu_i^x\right)^2 \equiv 1 \quad \text{and} \quad \sum_{i=1}^{(D-1)/2}\omega_s \cdot 2 \cdot \left(\mu_i^x\right)^2 \equiv 1 \ .$$

From [5], one nonlinear equation from the optimization problem is given by

$$\sum_{i=1}^{(D-1)/2}\left(\mu_i^x\right)^2 - \frac{D}{2} = 0 \ .$$

Based on this relationship, the weight must be $\omega_s = \frac{1}{D}$. ■

4 Decomposition Methods

4.1 Case I: directly observed, indirectly observed

Often, only a part of the state is observed through the measurement model. Based on stochastic dependency, the indirectly observed part is updated. The filter step

$$f\left(\underline{x}|\underline{y}\right) = \frac{f\left(\underline{y}|\underline{x}^o\right) f\left(\underline{x}\right)}{f\left(\underline{y}\right)}$$

can be written in the from

$$f\left(\underline{x}|\underline{y}\right) = f\left(\underline{x}^u|\underline{x}^o\right) \cdot \frac{f\left(\underline{y}|\underline{x}^o\right) f\left(\underline{x}^o\right)}{f\left(\underline{y}\right)} = f\left(\underline{x}^u|\underline{x}^o\right) \cdot f\left(\underline{x}^o|\underline{y}\right) ,$$

where the state vector is decomposed into an observed and an indirectly observed part

$$\underline{\boldsymbol{x}} = \begin{bmatrix} \underline{\boldsymbol{x}}^o \\ \underline{\boldsymbol{x}}^u \end{bmatrix} .$$

The measurement equation can be linear or nonlinear. In the nonlinear case

$$\underline{\boldsymbol{y}} = \underline{h}\left(\underline{\boldsymbol{x}}^o, \underline{\boldsymbol{v}}\right) \ ,$$

the directly observed state $\underline{\boldsymbol{x}}^o$ is estimated by using a linear regression Kalman filter and after that the indirectly observed state is updated. In the linear regression Kalman filter, the state vector $\underline{\boldsymbol{x}}$ is Gaussian distributed with mean and covariance matrix

$$\underline{\mu}^x = \begin{bmatrix} \underline{\mu}^o \\ \underline{\mu}^u \end{bmatrix} \ , \quad \mathbf{C}^{x,x} = \begin{bmatrix} \mathbf{C}^{o,o} & \mathbf{C}^{o,u} \\ \mathbf{C}^{u,o} & \mathbf{C}^{u,u} \end{bmatrix} \ .$$

To update the indirectly observed state, the estimated mean $\underline{\mu}_e^o$ and estimated covariance matrix $\mathbf{C}_e^{o,o}$ of the density $f(\underline{x}^o|\underline{y})$ are used. According to [8], the mean vector of the indirectly observed state is updated by

$$\underline{\mu}_e^u = \underline{\mu}_p^u + \mathbf{L} \cdot \left(\underline{\mu}_e^o - \underline{\mu}_p^o\right) \ , \tag{3}$$

the cross-covariance matrix $\mathbf{C}_e^{u,o}$ and the covariance matrix $\mathbf{C}_e^{u,u}$ are calculated according to

$$\mathbf{C}_e^{u,o} = \mathbf{L} \cdot \mathbf{C}_e^{o,o} \ , \tag{4}$$

$$\mathbf{C}_e^{u,u} = \mathbf{C}_p^{u,u} - \mathbf{L} \cdot \left(\mathbf{C}_p^{o,o} - \mathbf{C}_e^{o,o}\right) \cdot (\mathbf{L})^{\mathrm{T}} \ , \tag{5}$$

respectively, with the matrix $\mathbf{L} = \mathbf{C}_p^{u,o} \cdot \left(\mathbf{C}_p^{o,o}\right)^{-1}$.

4.2 Case II: Linear, Nonlinear

Similar to [14], [9], and [7], conditionally linear models

$$\underline{\boldsymbol{y}} = \underline{g}\left(\underline{\boldsymbol{x}}^n\right) + \mathbf{H}\left(\underline{\boldsymbol{x}}^n\right) \cdot \underline{\boldsymbol{x}}^l$$

are considered. For the Gaussian filter, the joint density $f\left(\underline{x}, \underline{y}\right)$ has to be approximated by a multivariate Gaussian density, where the joint density is

$$f\left(\underline{x}, \underline{y}\right) = \delta\left(\underline{y} - \underline{g}\left(\underline{x}^n\right) + \mathbf{H}\left(\underline{x}^n\right) \cdot \underline{x}^l\right) \cdot f\left(\underline{x}^n, \underline{x}^l\right) \ .$$

The state is decomposed into a nonlinear and linear part

$$\underline{\boldsymbol{x}} = \begin{bmatrix} \underline{\boldsymbol{x}}^n \\ \underline{\boldsymbol{x}}^l \end{bmatrix}$$

and the density of the state is Gaussian distributed with mean and covariance matrix

$$\underline{\mu}^x = \begin{bmatrix} \underline{\mu}^n \\ \underline{\mu}^l \end{bmatrix}, \quad \mathbf{C}^{x,x} = \begin{bmatrix} \mathbf{C}^{n,n} & \mathbf{C}^{n,l} \\ \mathbf{C}^{l,n} & \mathbf{C}^{l,l} \end{bmatrix} .$$

The density of the state is separated by using Bayes' law

$$f\left(\underline{x}^n, \underline{x}^l\right) = f\left(\underline{x}^l | \underline{x}^n\right) \cdot f\left(\underline{x}^n\right) .$$

The conditional density is given by

$$f\left(\underline{x}^l | \underline{x}^n\right) = \mathcal{N}\left(\underline{x}^l - \underline{\mu}\left(\underline{x}^n\right), \mathbf{C}^{l|n}\right)$$

with mean and covariance matrix

$$\begin{aligned} \underline{\mu}\left(\underline{x}^n\right) &= \underline{\mu}^l + \mathbf{C}^{l,n} \cdot \left(\mathbf{C}^{n,n}\right)^{-1} \cdot \left(\underline{x}^n - \underline{\mu}^n\right) \\ \mathbf{C}^{l|n} &= \mathbf{C}^{l,l} - \mathbf{C}^{l,n} \cdot \left(\mathbf{C}^{n,n}\right)^{-1} \cdot \mathbf{C}^{n,l} . \end{aligned}$$

The density for the nonlinear part $f\left(\underline{x}^n\right)$ is approximated with a Dirac mixture density based on the deterministic sampling scheme from Section 3

$$f\left(\underline{x}^n\right) \approx \tilde{f}\left(\underline{x}^n, \underline{\eta}\right) = \sum_{i=1}^{L} \omega \cdot \delta\left(\underline{x}^n - \underline{\mu}_i^n\right) ,$$

with $L = N \cdot (D-1) + 1$. The approximated joint density $\tilde{f}(\underline{x}, \underline{y})$ is given by

$$\tilde{f}(\underline{x}, \underline{y}) = \delta\left(\underline{y} - \underline{g}\left(\underline{x}^n\right) + \mathbf{H}\left(\underline{x}^n\right) \cdot \underline{x}^l\right) \cdot \mathcal{N}\left(\underline{x}^l - \underline{\mu}\left(\underline{x}^n\right), \mathbf{C}^{l|n}\right) \cdot \sum_{i=1}^{L} \omega \cdot \delta\left(\underline{x}^n - \underline{\mu}_i^n\right) .$$

For the Gaussian filter, the joint density is approximated with a Gaussian density according to

$$\tilde{f}(\underline{x}, \underline{y}) \approx \mathcal{N}\left(\begin{bmatrix} \underline{x} \\ \underline{y} \end{bmatrix} - \begin{bmatrix} \underline{\mu}^x \\ \underline{\mu}^y \end{bmatrix}, \begin{bmatrix} \mathbf{C}^{x,x} & \mathbf{C}^{x,y} \\ \mathbf{C}^{y,x} & \mathbf{C}^{y,y} \end{bmatrix} \right) .$$

For calculating the mean $\underline{\mu}^y$, the covariance matrix $\mathbf{C}^{y,y}$ and the cross-covariance matrix $\mathbf{C}^{x,y}$ first the approximated density $\tilde{f}(\underline{y})$ depending on $\underline{y}$ is calculated

$$\begin{aligned}\tilde{f}(\underline{y}) &= \int_{\mathbb{R}}\int_{\mathbb{R}} \tilde{f}(\underline{x},\underline{y})\,\mathrm{d}\underline{x}^l\,\mathrm{d}\underline{x}^n \\ &= \sum_{i=1}^{L} \omega \int \delta\left(\underline{y} - \underline{g}\left(\underline{\mu}_i^n\right) + \mathbf{H}\left(\underline{\mu}_i^n\right)\cdot \underline{x}^l\right)\cdot \mathcal{N}\left(\underline{x}^l - \underline{\mu}\left(\underline{\mu}_i^n\right), \mathbf{C}^{l|n}\right)\mathrm{d}\underline{x}^l\,,\end{aligned}$$

which results in a Gaussian mixture

$$\tilde{f}(\underline{y}) = \sum_{i=1}^{L} \omega\cdot\mathcal{N}\left(\underline{y} - \underline{\mu}_i^y, \mathbf{C}_i^{y,y}\right)\,,$$

with mean and covariance matrix

$$\begin{aligned}\underline{\mu}_i^y &= \underline{g}\left(\underline{\mu}_i^n\right) + \mathbf{H}\left(\underline{\mu}_i^n\right)\cdot\underline{\mu}\left(\underline{\mu}_i^n\right)\,,\\ \mathbf{C}_i^{y,y} &= \mathbf{H}\left(\underline{\mu}_i^n\right)\cdot\mathbf{C}^{l|n}\cdot\left(\mathbf{H}\left(\underline{\mu}_i^n\right)\right)^{\mathrm{T}}\,,\end{aligned}$$

respectively. The first and the second moments are approximated by

$$\underline{\mu}^y = \omega\cdot\sum_{i=1}^{L}\underline{\mu}_i^y\,, \tag{6}$$

$$\mathbf{C}^{y,y} = \sum_{i=1}^{L}\left(\omega\cdot\mathbf{C}_i^{y,y} + \omega_s\cdot\left(\underline{\mu}_i^y - \underline{\mu}^y\right)\cdot\left(\underline{\mu}_i^y - \underline{\mu}^y\right)^{\mathrm{T}}\right)\,, \tag{7}$$

respectively, where the covariance matrix consists of the sample covariance matrix and the covariance matrix of the linear part. The cross-covariance matrix is approximated with

$$\begin{aligned}\mathbf{C}^{x,y} &= \int_{\mathbb{R}}\int_{\mathbb{R}}\left(\underline{x} - \underline{\mu}^x\right)\cdot\left(\underline{y} - \underline{\mu}^y\right)^{\mathrm{T}}\cdot\tilde{f}(\underline{x},\underline{y})\,\mathrm{d}\underline{x}\,\mathrm{d}\underline{y}\\ &= \sum_{i=1}^{L}\left(\omega\cdot\begin{bmatrix}\mathbf{0}\\ \mathbf{C}^{l|n}\mathbf{H}\left(\underline{\mu}_i^n\right)^{\mathrm{T}}\end{bmatrix} + \omega_s\cdot\left(\begin{bmatrix}\underline{\mu}_i^n\\ \underline{\mu}\left(\underline{\mu}_i^n\right)\end{bmatrix} - \underline{\mu}^x\right)\cdot\left(\underline{\mu}_i^y - \underline{\mu}^y\right)^{\mathrm{T}}\right)\,.\end{aligned} \tag{8}$$

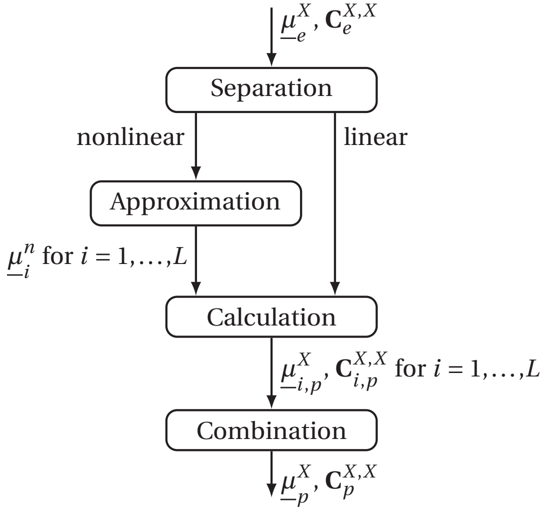

$\underline{\mu}_e^X$, $\mathbf{C}_e^{X,X}$
Separation
nonlinear
linear
Approximation
$\underline{\mu}_i^n$ for $i = 1,\dots,L$
Calculation
$\underline{\mu}_{i,p}^X$, $\mathbf{C}_{i,p}^{X,X}$ for $i = 1,\dots,L$
Combination
$\underline{\mu}_p^X$, $\mathbf{C}_p^{X,X}$

Figure 1: Information flow of the prediction step.

5 Estimation

5.1 Prediction Step

The information flow for the prediction step is shown Figure 1. First, the state has to be separated into a nonlinear and linear part. Then, the system equation has to be converted into the form

$$\underline{\boldsymbol{x}}_{k+1} = \underline{g}_k\left(\underline{\boldsymbol{x}}_k^n, \underline{u}_k, \underline{\boldsymbol{w}}_k^n\right) + \mathbf{H}_k\left(\underline{\boldsymbol{x}}_k^n, \underline{u}_k, \underline{\boldsymbol{w}}_k^n\right) \cdot \begin{bmatrix} \underline{\boldsymbol{x}}_k^l \\ \underline{\boldsymbol{w}}_k^l \end{bmatrix} ,$$

where the system state is augmented with the noise variable $\underline{\boldsymbol{X}}_k = \left[\underline{\boldsymbol{x}}_k^{\mathrm{T}} \quad \underline{\boldsymbol{w}}_k^{\mathrm{T}}\right]^{\mathrm{T}}$ to consider additive and/or multiplicative noise. The mean and covariance matrix is then given by

$$\underline{\mu}_e^X = \left[\left(\underline{\mu}_e^x\right)^{\mathrm{T}} \quad \underline{0}^{\mathrm{T}}\right]^{\mathrm{T}} , \quad \mathbf{C}_e^{X,X} = \begin{bmatrix} \mathbf{C}_e^{x,x} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_k^{w,w} \end{bmatrix} .$$

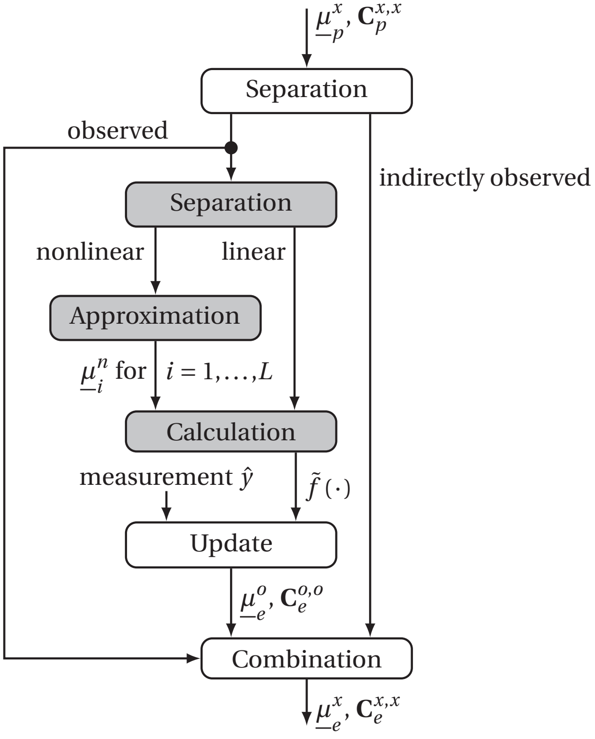

$\underline{\mu}_p^x, \mathbf{C}_p^{x,x}$
Separation
observed
indirectly observed
Separation
nonlinear
linear
Approximation
$\underline{\mu}_i^n$ for $i = 1,\ldots,L$
Calculation
measurement $\hat{y}$
$\tilde{f}(\cdot)$
Update
$\underline{\mu}_e^o, \mathbf{C}_e^{o,o}$
Combination
$\underline{\mu}_e^x, \mathbf{C}_e^{x,x}$

Figure 2: Information flow of the filter step. The gray boxes indicate components that are identical with the prediction step.

In the second step, the sample points for the nonlinear part are calculated based on the scheme described in Section 3. For calculating the predicted mean and covariance matrix, equations (6) and (7) are used, where the components of the Gaussian Mixture are used.

5.2 Filter Step

In a first step, the directly observed and the indirectly observed state are separated. Then the measurement equation has to be converted, as in the prediction step and the sample points for the nonlinear state are determined. Then, the covariance matrix (7), cross-covariance matrix (8), and the predicted measurement (6) are calculated. Based on the approximated joint density, mean and covariance matrix are estimated according to the conditional density $f\left(\underline{x}|\underline{\hat{y}}\right)$ for a given measurement $\underline{\hat{y}}$

$$\underline{\mu}_e^o = \underline{\mu}_p^o + \mathbf{C}^{x,y} \cdot \left(\mathbf{C}^{y,y}\right)^{-1} \cdot \left(\underline{\hat{y}} - \underline{\mu}^y\right) ,$$
$$\mathbf{C}_e^{o,o} = \mathbf{C}_p^{o,o} - \mathbf{C}^{x,y} \cdot \left(\mathbf{C}^{y,y}\right)^{-1} \cdot \left(\mathbf{C}^{x,y}\right)^{\mathrm{T}} .$$

After the mean and the covariance matrix of the directly observed state is updated, the indirectly observed state is calculated based on (3), (4) and (5). In Figure 2, the information flow for the filter step is shown.

6 Considered Example

The proposed algorithm is evaluated in a pose tracking scenario. Based on the observation of two different sensors, the translation and the rotation of an object is estimated. The estimation for the pose with respect to a global coordinate system is performed by known reference signals. Several loudspeakers emit signals, which are received by microphones attached to the tracked object.

The other type of sensors measures the inertial angular velocity of the object with respect to the object coordinate system.

The state vector consists of the translation, the rotation, the translation velocity, and the angular velocity in three-dimensional space. Based on the proposed principles, the twelve-dimensional state vector is decomposed to reduce the computational effort. In this example, the sensors only observe a part of the state vector. Furthermore, the kinematics of the object is described by a nonlinear system equation, which can be separated into a linear and a nonlinear part.

6.1 Prediction

The system model describes the evolution of the state over time. For the considered example, the dynamic behavior of the translation and the rotation has to be characterized in a adequate manner.

Translation

For the translation, a constant velocity model is assumed. This model is given by a linear differential equation, which is represented in discrete time as

$$\underline{\boldsymbol{w}}_{k+1} = \mathbf{A} \cdot \underline{\boldsymbol{w}}_k + \underline{\boldsymbol{w}}_k^z \,, \quad \text{with } \mathbf{A} = \begin{bmatrix} \mathbf{I}_{3,3} & T \cdot \mathbf{I}_{3,3} \\ \mathbf{0} & \mathbf{I}_{3,3} \end{bmatrix} ,$$

and $\underline{\boldsymbol{w}}_k$ consists of the translation $\underline{\boldsymbol{T}}_k$ and the velocity $\underline{\boldsymbol{v}}_k$. T is the sampling time. The process noise $\underline{\boldsymbol{w}}_k^z$ is assumed to be white, zero-mean, and Gaussian with covariance matrix

$$\mathbf{Q}_{\underline{w}_k^z} = \begin{bmatrix} \frac{T^3}{3} \cdot \mathbf{q} & \frac{T^2}{2} \cdot \mathbf{q} \\ \frac{T^2}{2} \cdot \mathbf{q} & T \cdot \mathbf{q} \end{bmatrix} , \quad \text{with } \mathbf{q} = \operatorname{diag}\left(\begin{bmatrix} q_{w_{V,x}}^2 & q_{w_{V,y}}^2 & q_{w_{V,z}}^2 \end{bmatrix}^{\mathrm{T}} \right) .$$

Rotation

The rotation is described by a rotation vector [2]. In a tracking scenario, the rotation vector is typically time-variant. This dynamic behavior can be described by a nonlinear differential equation [3]

$$\dot{\underline{r}}(t) = \begin{cases} \mathbf{a}(\underline{r}(t)) \cdot \underline{\omega}(t) & \text{for } \left\|\underline{r}(t)\right\| \in \,]0,\pi] \\ \underline{\omega}(t) & \text{for } \left\|\underline{r}(t)\right\| = 0 \end{cases} , \tag{9}$$

which depends on the angular velocity $\underline{\omega}(t)$. The matrix $\mathbf{a}(\cdot)$ is given by

$$\mathbf{a}(\underline{r}(t)) = \mathbf{I}_{3,3} + \tfrac{1}{2} \cdot \mathbf{C}(\underline{r}(t)) + \frac{1 - \frac{1}{2} \cdot \|\underline{r}(t)\| \cdot \cot\left(\frac{\|\underline{r}(t)\|}{2}\right)}{\|\underline{r}(t)\|^2} \mathbf{C}(\underline{r}(t)) \cdot \mathbf{C}(\underline{r}(t)) \;,$$

where $\cot(\cdot)$ is the cotangent and the matrix $\mathbf{C}(\underline{r}(t))$ is a skew-symmetric matrix

$$\mathbf{C}(\underline{r}(t)) = \begin{bmatrix} 0 & -r_z(t) & r_y(t) \\ r_z(t) & 0 & -r_x(t) \\ -r_y(t) & r_x(t) & 0 \end{bmatrix} .$$

The nonlinear differential equation (9) is discretized by the Euler formula

$$\dot{\underline{r}}(t) \approx \tfrac{\underline{r}_{k+1} - \underline{r}_k}{T}$$

and the resulting discrete-time difference equation is

$$\underline{r}_{k+1} = \underline{r}_k + T \cdot \mathbf{a}(\underline{r}_k) \cdot \underline{\omega}_k \ .$$

In (9), the range of the norm of the rotation vector lies in the interval zero to π. To achieve this constraint in the estimation procedure, a forward inference is performed by

$$\underline{\boldsymbol{r}}_{k,\text{new}} = \underline{\boldsymbol{r}}_k \cdot \left(1 - \tfrac{2\pi}{\|\underline{\boldsymbol{r}}_k\|}\right) \ ,$$

if the norm of the rotation vector is higher than π. The forward inference can be calculated by a prediction step, where no process noise is assumed. The angular velocity is modeled as a random walk according to

$$\underline{\boldsymbol{\omega}}_{k+1} = \underline{\boldsymbol{\omega}}_k + \underline{\boldsymbol{w}}_k^{\omega} \ ,$$

where the process noise $\underline{\boldsymbol{w}}_k^{\omega}$ is white, zero-mean, and Gaussian with covariance matrix

$$\mathbf{Q}_{\underline{w}^{\omega}} = \text{diag}\left(\begin{bmatrix} q^2_{w_{\omega,x}} & q^2_{w_{\omega,y}} & q^2_{w_{\omega,z}} \end{bmatrix}^{\mathrm{T}}\right) \ .$$

System Equation

In this example, the nonlinear part depends on the rotation vector. In the linear part of the state $\underline{\boldsymbol{x}}_k^l$, the translation, the velocity, and the angular velocity is considered. The system equation is written in the form

$$\begin{bmatrix} \underline{\boldsymbol{r}}_{k+1} \\ \underline{\boldsymbol{\omega}}_{k+1} \\ \underline{\boldsymbol{w}}_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} \underline{\boldsymbol{r}}_k \\ \underline{0} \\ \underline{0} \end{bmatrix}}_{\underline{g}(\underline{\boldsymbol{r}}_k)} + \underbrace{\begin{bmatrix} T \cdot \mathbf{a}(\underline{\boldsymbol{r}}_k) & \mathbf{0} \\ \mathbf{I}_{3,3} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix}}_{\mathbf{H}(\underline{\boldsymbol{r}}_k)} \underbrace{\begin{bmatrix} \underline{\boldsymbol{\omega}}_k \\ \underline{\boldsymbol{w}}_k \end{bmatrix}}_{\underline{\boldsymbol{x}}_k^l} + \underline{\boldsymbol{w}}_k \ ,$$

where the covariance matrix of the noise $\underline{\boldsymbol{w}}_k$ is given by

$$\mathbf{Q} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{\underline{w}^{\omega}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Q}_{\underline{w}_n^z} \end{bmatrix} \ .$$

6.2 Filtering

Inertial Sensors

The inertial sensors measure the angular velocity with respect to the object coordinate system. The measurement equation is modeled by a linear equation

$$\underline{\boldsymbol{y}}_k^{\omega} = k_G \cdot \mathbf{A} \cdot \underline{\boldsymbol{\omega}}_k + \underline{b} + \underline{\boldsymbol{v}}_k^{\omega} \ ,$$

where k_G is a sensor specific factor, $\mathbf{A}$ is a misalignment matrix, $\underline{b}$ is the sensor offset, and $\underline{\boldsymbol{v}}_k^{\omega}$ is the measurement noise. If a measurement is available, the filtering step can be performed by using the Kalman filter equation, because the measurement equation is linear and the state is approximated with a Gaussian distribution.

Acoustic Sensor

The second sensor measures the wave field. The measurement equation depends on the translation and rotation. The directly observed state is estimated by using the Gaussian filter. Based on the estimated mean and covariance matrix, the indirectly observed state is updated. The nonlinear measurement equation is given by

$$\boldsymbol{y}_k^j = \sum_{i=1}^{N} \frac{1}{4\pi \cdot \left\| \mathbf{D}(\underline{\boldsymbol{r}}_k) \cdot \underline{\tilde{p}}^j + \underline{\boldsymbol{T}}_k - \underline{x}^i \right\|} \cdot \sum_{m=0}^{k} s_m^i \cdot \operatorname{sinc}\left((k-m) \cdot \pi - \frac{\left\| \mathbf{D}(\underline{\boldsymbol{r}}_k) \cdot \underline{\tilde{p}}^j + \underline{\boldsymbol{T}}_k - \underline{x}^i \right\|}{c \cdot T} \right) + \boldsymbol{v}_k^j \ ,$$

where $\mathbf{D}\left(\underline{\boldsymbol{r}}_k\right)$ is the rotation matrix

$$\mathbf{D}\left(\underline{\boldsymbol{r}}_k\right) = \mathbf{I}_{3,3} + \frac{\sin(\|\underline{\boldsymbol{r}}_k\|)}{\|\underline{\boldsymbol{r}}_k\|} \mathbf{C}\left(\underline{\boldsymbol{r}}_k\right) + \frac{1-\cos(\|\underline{\boldsymbol{r}}_k\|)}{\|\underline{\boldsymbol{r}}_k\|^2} \mathbf{C}\left(\underline{\boldsymbol{r}}_k\right) \mathbf{C}\left(\underline{\boldsymbol{r}}_k\right) \ .$$

The measurement equation describes the wave propagation of the N sources to the jth microphone. The signal of each source is characterized by the symbols s_m^i, where m is the discrete time of the source and k of the sensor. The sinc-interpolation [11] is used to get a continuous time signal in order to achieve subsample resolution. T is the sampling interval, c the velocity of sound, and $\boldsymbol{v}_k^j$ the measurement noise of the sensor j.

7 Results

The proposed approach is evaluated in a simulation and in an experiment.

7.1 Simulation

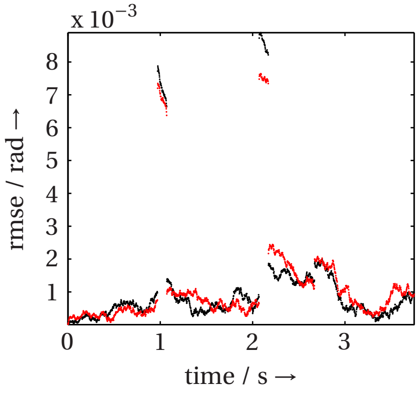
In the simulation setup, a moving target object is considered. The trajectory of the object is simulated by piecewise constant translational and angular velocities. Four microphones are attached to the target, which are receiving multi carrier-code division multiple access (MC-CDMA) signals from four loudspeakers in order to achieve a distinguishable mapping. For the simulation, the signals are delayed depending on the time-variant distance between microphone and loudspeaker. Furthermore, an inertial measurement unit consisting of three gyroscopes measures the angular velocity with respect to the target coordinate system. The measurement frequency of the microphones and the gyroscopes are 48,000 Hz and 480 Hz, respectively. The signals received by the microphones are corrupted with noise. This noise is generated by mirror image sources in order to model reverberations. The signals emitted by the four loudspeakers are reflected at walls of the room, which is modeled by 24 mirror image sources. The attenuation factor of the walls is set to be 0.5, which results in an SNR of 4.247 dB. The angular velocity is corrupted by additive zero-mean Gaussian noise with covariance matrix $\mathbf{R}_{\underline{v}^\omega} = \mathbf{I}_{3,3} \cdot 10^{-4}$.

In the simulation, a constant position model is used due to the fact that the simulated velocity has points of discontinuity. The initial states and covariance matrices are set to

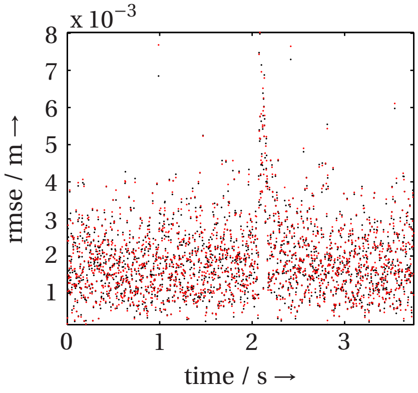
$$
\begin{aligned}
\underline{\mu}_0^r &= \begin{bmatrix} 0 & -0.0245 & 0 \end{bmatrix}^\mathrm{T} , & \mathbf{C}_0^{r,r} &= \mathbf{I}_{3,3} \cdot (10^{-3} \cdot \pi/180)^2 , \\
\underline{\mu}_0^T &= \begin{bmatrix} -0.9535 & -0.9969 & 2 \end{bmatrix}^\mathrm{T} , & \mathbf{C}_0^{T,T} &= \mathbf{I}_{3,3} \cdot 10^{-6} , \\
\underline{\mu}_0^\omega &= \begin{bmatrix} 0 & -0.6911 & 0 \end{bmatrix}^\mathrm{T} , & \mathbf{C}_0^{\omega,\omega} &= \mathbf{I}_{3,3} \cdot 10^{-3} ,
\end{aligned}
$$

and the process and measurement noise covariance matrices to

$$
\begin{aligned}
&\mathbf{Q}_{\underline{w}^\omega} = \mathbf{I}_{3,3} \cdot 5.1404 \cdot 10^{-5} , \quad \mathbf{Q}_{\underline{w}_n^z} = \operatorname{diag}\begin{bmatrix} 5.0 \cdot 10^{-7} & 5.0 \cdot 10^{-7} & 5.0 \cdot 10^{-8} \end{bmatrix} , \\
&\mathbf{R}_{\underline{v}} = \mathbf{I}_{4,4} \cdot 10^{-3} , \quad \mathbf{R}_{\underline{v}^\omega} = \mathbf{I}_{3,3} \cdot 10^{-4} .
\end{aligned}
$$



(a) rmse of the rotation vector.



(b) rmse of the translation vector.

Figure 3: Rmse of the rotation and translation vector, where the black points represent the decomposed algorithm and the red points result from the full state case.

For the approximation, five sample points for each dimension are used. In the first simulation run, no decomposition is performed. Furthermore, the measurement and process noise is augmented in the state vector. In this case, a total of 89 sample points are used for approximation.

The second simulation run takes advantages of the structure of the system and measurement equation in order to reduce the computational effort. For the prediction step, the state vector is separated into the linear and nonlinear part, where only 13 sample points are used. For the filtering step, the Kalman filter equations are used when measurements from the inertial measurement unit are available. In the other case, when the microphones measures the wave field, the

Table 2: Average and standard deviation values of the rmse for the rotation and translation. The Gaussian filter is compared to the UKF.

	Translation in 10^{-4} m	Rotation in 10^{-4} rad
GF (decomposed)	19 ± 9.8515	11 ± 17
GF (full state)	19 ± 9.9196	12 ± 16
UKF (decomposed)	19 ± 10.3126	20 ± 35
UKF (full state)	18 ± 10.0988	19 ± 14

state vector is separated into the directly observed and indirectly observed part, which results in an approximation with 25 sample points.

The computation time decreases by a factor of four compared to the first simulation run. Furthermore, regarding the rotation, the root mean square error (rmse) of the decomposed GF is lower compared to full state algorithms (GF, UKF) . However, the accuracy may decrease for the rotation vector when the change of the angular velocity is high. The average rmses of rotation and translation are shown in Table 2. The rmses of rotation and translation for an example simulation run are shown in Figure 3.

7.2 Experiment

An object is moved ten times from a starting point to an end point on a straight line. The experimental setup is similar to the simulation, where in the experiment a constant velocity model is assumed. Furthermore, five loudspeakers are periodically emitting the signals and the sampling frequency of the gyroscopes are 200 Hz. The initial state and covariance matrix is set to

$$\begin{aligned}
\underline{\mu}_0^r &= \begin{bmatrix} 0 & 0 & \pi/2 \end{bmatrix}^{\mathrm{T}} , & \mathbf{C}_0^{r,r} &= \mathbf{I}_{3,3}(1\cdot\pi/180)^2 , \\
\underline{\mu}_0^T &= \begin{bmatrix} -0.05 & -0.75 & 1.16 \end{bmatrix}^{\mathrm{T}} , & \mathbf{C}_0^{T,T} &= \mathbf{I}_{3,3}\cdot 10^{-4} , \\
\underline{\mu}_0^\omega = \underline{\mu}^V &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}} , & \mathbf{C}_0^{\omega,\omega} = \mathbf{C}_0^{V,V} &= \mathbf{I}_{3,3}\cdot 9\cdot 10^{-6} ,
\end{aligned}$$

respectively. The covariance matrix of process and measurement noise was

$$\begin{aligned}
&\mathbf{Q}_{\underline{w}^\omega} = \mathbf{I}_{3,3}\cdot 6.3462\cdot 10^{-6} , \\
&\mathbf{Q}_{\underline{w}_n^x} = \mathbf{Q}_{\underline{w}_n^y} = \begin{bmatrix} 3.014\cdot 10^{-14} & 2.170\cdot 10^{-10} \\ 2.170\cdot 10^{-10} & 2.083\cdot 10^{-6} \end{bmatrix} , \\
&\mathbf{Q}_{\underline{w}_n^z} = \mathbf{Q}_{\underline{w}_n^y}\cdot 10^{-4} , \quad \mathbf{R}^M = \mathbf{I}_{4,4}\cdot 0.09 , \quad \mathbf{R}^\omega = \mathbf{I}_{3,3}\cdot 0.01 .
\end{aligned}$$

In Figure 4, the results for the five test runs are shown. The measured end point and the distance between start and end point is given in Table 3. In addition, the average over the ten test runs and the standard deviation is listed.

Table 3: The average results of GF from ten test runs.

	End point			Distance
	x in m	*y* in m	*z* in m	
True	-0.05	1.33	1.16	2.08
Full state	0.016 ± 0.019	1.310 ± 0.022	1.048 ± 0.040	2.066 ± 0.019
Decomp.	0.016 ± 0.017	1.306 ± 0.026	1.036 ± 0.055	2.062 ± 0.025

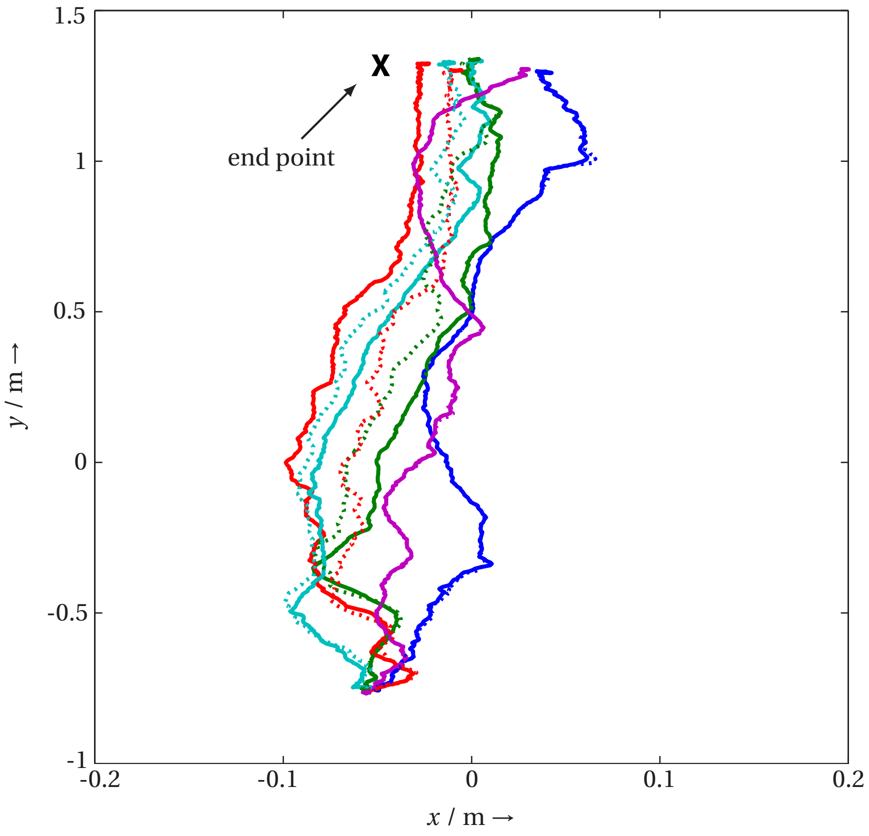


Figure 4: Results from five test runs. The solid lines are the estimates from the full state case. The dashed lines are the results from the decomposed case.

8 Conclusions

In this paper, two principles for reducing the computational effort for state estimation in nonlinear systems are discussed and exploited for the Gaussian filter. These decompositions exploit the structure of the nonlinear system equations and facilitate to reduce the number of sample points for approximating the state density. In doing so, the computation time can be significantly decreased without significantly affecting the estimation quality. The advantages are shown in simulations and experiments.

In contrast to a comparable linear/nonlinear decomposition approach presented in [9] for the unscaled version of the UKF, the proposed approaches are more generally applicable. For instance, scaling the sample points can be considered, which is essential for very high-dimensional problems. Furthermore, the decomposition into directly/indirectly observed states is also taken into account.

References

[1] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[2] Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck. Instantaneous Pose Estimation using Rotation Vectors. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, April 2009.

[3] J. E. Bortz. A New Mathematical Formulation for Strapdown Inertial Navigation. *IEEE Transactions on Aerospace and Electronic Systems*, AES-7(1):61–66, Januar 1971.

[4] Fred Daum and Jim Huang. Curse of Dimensionality and Particle Filters. In *Proceedings of the 2003 IEEE Aerospace Conference*, volume 4, pages 1979–1993, March 2003.

[5] Marco F. Huber and Uwe D. Hanebeck. Gaussian Filter based on Deterministic Sampling for High Quality Nonlinear Estimation. In *Proceedings of the 17th IFAC World Congress*, Seoul, Republic of Korea, July 2008.

[6] Simon J. Julier and Jeffrey K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[7] Vesa Klumpp, Felix Sawo, Uwe D. Hanebeck, and Dietrich Fränken. The Sliced Gaussian Mixture Filter for Efficient Nonlinear Estimation. In *Proceedings of the 11th International Conference on Information Fusion (Fusion 2008)*, pages 1–8, Cologne, Germany, July 2008.

[8] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter. *Nonlinear Kalman Filtering for Force-Controlled Robot Tasks.* Springer, 2005.

[9] Mark R. Morelande and Bill Moran. An Unscented Transformation for Conditionally Linear Models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages III–1417–III–1420, April 2007.

[10] Magnus Nørgaard, Niels K. Poulsen, and Ole Ravn. New Developments in State Estimation for Nonlinear Systems. *Automatica*, 36(11):1627–1638, November 2000.

[11] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-time Signal Processing.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2nd edition, 1999.

[12] Patrick Rößler, Frederik Beutler, Uwe D. Hanebeck, and Norbert Nitzsche. Motion Compression Applied to Guidance of a Mobile Teleoperator. In *Proceedings of the 2005 IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2495–2500, 2005.

[13] Tor S. Schei. A finite difference method for linearizing in nonlinear estimation algorithms. *Automatica*, 33(11):2051–2058, November 1997.

[14] Thomas B. Schön, Fredrik Gustafsson, and Per-Johan J. Nordlund. Marginalized Particle Filters for Mixed Linear/Nonlinear State-Space Models. *IEEE Transactions on Signal Processing*, 53(7):2279–2289, July 2005.

Paper B

Semi-Analytic Gaussian Assumed Density Filter

Authors: Marco F. Huber, Frederik Beutler, and Uwe D. Hanebeck

Edited version of the paper:

Marco F. Huber, Frederik Beutler, and Uwe D. Hanebeck. Semi-Analytic Gaussian Assumed Density Filter. In *Proceedings of the 2011 American Control Conference (ACC)*, pages 3006–3011, San Francisco, CA, USA, June 2011.

Semi-Analytic Gaussian Assumed Density Filter

Marco F. Huber[*], Frederik Beutler[**], and Uwe D. Hanebeck[**]

[*] Variable Image Acquisition and Processing Research Group
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB
Karlsruhe, Germany
marco.huber@ieee.org

[**] Intelligent Sensor-Actuator-Systems Laboratory (ISAS)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{beutler|uwe.hanebeck}@ieee.org

Abstract

For Gaussian Assumed Density Filtering based on moment matching, a framework for the efficient calculation of posterior moments is proposed that exploits the structure of the given nonlinear system. The key idea is a careful discretization of some dimensions of the state space only in order to decompose the system into a set of nonlinear subsystems that are conditionally integrable in closed form. This approach is more efficient than full discretization approaches. In addition, the new decomposition is far more general than known Rao-Blackwellization approaches relying on conditionally linear subsystems. As a result, the new framework is applicable to a much larger class of nonlinear systems.

1 Introduction

In Bayesian estimation, the hidden internal state of arbitrary systems has to be estimated based on measured input and output sequences that are typically corrupted by noise. For linear systems affected with Gaussian noise, the Kalman Filter [8] is the best estimator and allows a closed-form calculation. In case of nonlinearities in the system and measurement equation, however, estimation cannot be performed analytically in general. Instead, approximate state estimators have to be employed. Popular estimators for nonlinear systems make use of the Gaussian assumption for representing the posterior density of the state. Examples are sample-based “black box” approaches utilizing random sampling such as the Gaussian Particle Filter [9], deterministic sampling [2], or statistical linear regression [3, 7, 10, 12]. Alternatively to sampling, analytic approaches may be applicable, where analytic moment calculation [13] leads to closed form formulas of the required first two moments. However, this requires carefully performed derivations and thus, is impractical for high-dimensional systems. If the first two moments cannot be calculated in closed form, an nth-order Taylor series expansion can be used to approximate the underlying system and measurement equation for calculating the moments, which is used in the extended Kalman filter (EKF, [15]) or the Gaussian second-order filter [5]. On the other hand, if the system exhibits conditionally linear substructures, sample-based approaches can exploit this fact by using Rao-Blackwellization [14] in order to calculate the required moments more efficiently [1, 11].

In this paper, a combination of sample-based estimation and analytic moment calculation for Gaussian assumed density filters is proposed by using a very general form of Rao-Blackwellization. Instead of merely exploiting linear substructures, the new approach relies on an intelligent decomposition of the system and measurement equation into *nonlinear* subsystems that are conditionally integrable in closed form by means of discretizing *some dimensions of the state space*. Thus, given a discretization of some dimensions, moment calculation of the remaining dimensions of the state space can be performed analytically and exactly, which improves the overall estimation accuracy and reduces the computational burden.

The structure of the paper is as follows. In Section 2, a problem formulation is given, providing a brief introduction into Bayesian estimation and Gaussian assumed density filtering. If the moments can be calculated in closed form, we come up with the analytic Gaussian assumed density filter (AGF) as described in Section 3. The class of sample-based Gaussian assumed density filters (SGFs) is explained in Section 4. The proposed combination of analytic moment calculation and the sample-based approach, named semi-analytic Gaussian assumed density filter (SAGF), is part of Section 5. The performance of the SAGF is shown by means of simulation examples in Section 6. A final discussion and an outlook to future research work are part of Section 7.

2 Problem Formulation

A nonlinear discrete-time dynamic system is given by

$$\underline{\boldsymbol{x}}_{k+1} = \underline{a}_k\left(\underline{\boldsymbol{x}}_k, \underline{u}_k, \underline{\boldsymbol{w}}_k\right) , \quad (1)$$
$$\underline{\boldsymbol{y}}_k = \underline{h}_k\left(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{v}}_k\right) , \quad (2)$$

where the functions $\underline{a}_k(\cdot,\cdot,\cdot)$ and $\underline{h}_k(\cdot,\cdot)$ are known. The vector $\underline{\boldsymbol{x}}_k$ is the state of the system, $\underline{\boldsymbol{y}}_k$ is the measurement vector, and $\underline{u}_k$ is a known system input at the discrete time step k. The terms $\underline{\boldsymbol{v}}_k$ and $\underline{\boldsymbol{w}}_k$ represent zero-mean measurement and process noise, respectively. Measurement values $\underline{\hat{y}}_k$ are realizations of the measurement process (2).

2.1 Bayesian Estimation

In Bayesian estimation, two alternating steps, i.e., prediction and filtering, are performed for estimating the system state $\underline{\boldsymbol{x}}_k$. In the prediction step, the result $f^e(\underline{x}_k) \triangleq f(\underline{x}_k | \underline{\hat{y}}_{1:k})$ of the previous filter step is propagated from time step k to $k+1$ by means of

$$f^p\left(\underline{x}_{k+1}\right) \triangleq f\left(\underline{x}_{k+1} | \underline{\hat{y}}_{1:k}\right) = \int f\left(\underline{x}_{k+1} | \underline{x}_k\right) \cdot f^e\left(\underline{x}_k\right) \mathrm{d}\underline{x}_k , \quad (3)$$

where $f\left(\underline{x}_{k+1} | \underline{x}_k\right)$ is the transition density defined by (1). $\underline{\hat{y}}_{1:k} = \left(\underline{\hat{y}}_1, \ldots, \underline{\hat{y}}_k\right)$ summarizes all measurement values up to and including time step k. In the filter step, the current measurement value $\underline{\hat{y}}_k$ is used for updating the result of the prediction step $f^p(\underline{x}_k)$ according to Bayes' rule

$$f^e\left(\underline{x}_k\right) \triangleq f\left(\underline{x}_k | \underline{\hat{y}}_{1:k}\right) = \tfrac{1}{c_k} \cdot f\left(\underline{\hat{y}}_k | \underline{x}_k\right) \cdot f^p\left(\underline{x}_k\right) , \quad (4)$$

where $c_k = \int f\left(\underline{\hat{y}}_k | \underline{x}_k\right) \cdot f^p\left(\underline{x}_k\right) \mathrm{d}\underline{x}_k$ is a normalization constant and $f\left(\underline{\hat{y}}_k | \underline{x}_k\right)$ is the likelihood defined by (2).

For both prediction and filtering, closed-form solutions of the occurring integrals are not available in general and thus, appropriate approximations have to be applied. In this paper, we restrict attention to Gaussian assumed density filters [4], i.e., the densities in (3) and (4) are assumed to be Gaussian. Furthermore, the parameters of these Gaussian densities are calculated by moment matching. In doing so, filtering and prediction boils down to the efficient calculation of the mean vector and the covariance matrix.

2.2 Gaussian Assumed Density Filter

For deriving a Gaussian assumed density filter, it is sufficient to concentrate on the nonlinear transformation

$$\underline{\boldsymbol{y}} = \underline{g}(\underline{\boldsymbol{x}}) \tag{5}$$

of the Gaussian random vector $\underline{\boldsymbol{x}}$ with density function $f(\underline{x}) = \mathcal{N}(\underline{x}; \underline{\mu}^x, \mathbf{C}^x)$ to the random vector $\underline{\boldsymbol{y}}$ with density $f(\underline{y})$.

Forward Inference

In the forward inference, the random vector $\underline{\boldsymbol{x}}$ is propagated through the nonlinear transformation (5) in order to calculate the first two moments of $\underline{\boldsymbol{y}}$, i.e., mean $\underline{\mu}^y$ and covariance $\mathbf{C}^y$. This type of inference occurs in the prediction step, where the nonlinear transformation (5) corresponds to the system function (1) and the density of $\underline{\boldsymbol{x}}$ is given by $f^e(\underline{x})$ with mean $\underline{\mu}_k^e$ and covariance $\mathbf{C}_k^e$. Furthermore, $\underline{\mu}^y$ and $\mathbf{C}^y$ correspond to the predicted mean $\underline{\mu}_{k+1}^p$ and predicted covariance $\mathbf{C}_{k+1}^p$, respectively. Analogously, forward inference is employed for calculating the predicted measurement, which is required for backward inference as described next.

Backward Inference

The goal of the backward inference is to determine the conditional density $f(\underline{x}|\underline{y})$, i.e., the conditional mean $\underline{\mu}^{x|y}$ and covariance $\mathbf{C}^{x|y}$. For this purpose, the joint density $f(\underline{x}, \underline{y})$ is determined first. By additionally assuming that this joint density is Gaussian[1], we need to compute the cross-covariance $\mathbf{C}^{x,y}$ of the joint covariance block matrix

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}^x & \mathbf{C}^{x,y} \\ (\mathbf{C}^{x,y})^{\mathrm{T}} & \mathbf{C}^y \end{bmatrix} .$$

1 This assumption is common in Gaussian assumed density filtering (as in the EKF or the UKF) and is only true for linear systems affected with Gaussian noise. Otherwise it is an approximation.

Given the cross-covariance, the desired mean and covariance are then given by

$$\begin{aligned}\underline{\mu}^{x|y} &= \underline{\mu}^{x} + \mathbf{C}^{x,y} \cdot \left(\mathbf{C}^{y}\right)^{-1} \cdot \left(\hat{\underline{y}} - \underline{\mu}^{y}\right) , \\ \mathbf{C}^{x|y} &= \mathbf{C}^{x} - \mathbf{C}^{x,y} \cdot \left(\mathbf{C}^{y}\right)^{-1} \cdot \left(\mathbf{C}^{x,y}\right)^{\mathrm{T}} ,\end{aligned} \tag{6}$$

where $\underline{\mu}^y$ and $\mathbf{C}^y$ result from a forward inference step and $\hat{\underline{y}}$ is a realization of $\underline{\boldsymbol{y}}$.

3 Analytic Gaussian Assumed Density Filter (AGF)

At first, we demonstrate that the mean $\underline{\mu}^y$ and covariance $\mathbf{C}^y$ for forward inference as well as the cross-covariance $\mathbf{C}^{x,y}$ for backward inference can be determined analytically and exactly in special cases. Thus, the prediction step (3) and filter step (4) can be calculated in closed form. In order to calculate the mean $\underline{\mu}^y$, it can be utilized that

$$\underline{\mu}^y = \mathrm{E}\left\{\underline{y}\right\} = \int \underline{y} \cdot f\left(\underline{y}\right) \mathrm{d}\underline{y} = \iint \underline{y} \cdot f\left(\underline{x}, \underline{y}\right) \mathrm{d}\underline{x}\, \mathrm{d}\underline{y} \tag{7}$$

holds. With (5) and Bayes' rule, the joint density function $f\left(\underline{x}, \underline{y}\right)$ of $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$ can be written according to

$$f\left(\underline{x}, \underline{y}\right) = f\left(\underline{y}|\underline{x}\right) \cdot f\left(\underline{x}\right) = \delta\left(\underline{y} - \underline{g}\left(\underline{x}\right)\right) \cdot f\left(\underline{x}\right) , \tag{8}$$

where $\delta\left(\underline{x} - \underline{\mu}\right)$ is the Dirac delta distribution at position $\underline{\mu}$. Plugging (8) into (7) and utilizing the sifting property of the Dirac delta distribution results in

$$\underline{\mu}^y = \int \underline{g}\left(\underline{x}\right) \cdot f\left(\underline{x}\right) \mathrm{d}\underline{x} . \tag{9}$$

Hence, the mean of $\underline{\boldsymbol{y}}$ can be calculated directly based on the nonlinear function $\underline{g}(\cdot)$ and the density of $\underline{\boldsymbol{x}}$. With (8), the covariance of $\underline{\boldsymbol{y}}$ can be derived in a similar manner, which leads to

$$\begin{aligned}\mathbf{C}^y &= \iint \left(\underline{y} - \underline{\mu}^y\right) \cdot \left(\underline{y} - \underline{\mu}^y\right)^{\mathrm{T}} \cdot f\left(\underline{x}, \underline{y}\right) \mathrm{d}\underline{x}\, \mathrm{d}\underline{y} \\ &= \int \underline{g}\left(\underline{x}\right) \cdot \underline{g}\left(\underline{x}\right)^{\mathrm{T}} \cdot f\left(\underline{x}\right) \mathrm{d}\underline{x} - \underline{\mu}^y \cdot \left(\underline{\mu}^y\right)^{\mathrm{T}} .\end{aligned} \tag{10}$$

For the filter step, the cross-covariance $\mathbf{C}^{x,y}$ is required. Similar to the covariance $\mathbf{C}^{y}$, the cross-covariance $\mathbf{C}^{x,y}$ is calculated by

$$\begin{aligned}\mathbf{C}^{x,y} &= \iint \left(\underline{x}-\underline{\mu}^{x}\right)\cdot\left(\underline{y}-\underline{\mu}^{y}\right)^{\mathrm{T}}\cdot f\left(\underline{x},\underline{y}\right)\mathrm{d}\underline{x}\,\mathrm{d}\underline{y} \\ &= \int \underline{x}\cdot\underline{g}(\underline{x})^{\mathrm{T}}\cdot f(\underline{x})\,\mathrm{d}\underline{x}-\underline{\mu}^{x}\cdot\left(\underline{\mu}^{y}\right)^{\mathrm{T}}\,. \end{aligned} \tag{11}$$

Again, merely $\underline{g}(\cdot)$ and $f(\underline{x})$ are necessary for calculating the covariance. Unfortunately, analytically solving the integrals in (9), (10), and (11) is not possible in general. For special function types such as polynomials or trigonometric functions and their combination, however, a closed-form solution is available. If the moments can be calculated in closed form, the resulting estimator is called analytic Gaussian assumed density filter (AGF).

Example 1: Quadratic Transformation

In this example, the nonlinear transformation

$$\boldsymbol{y} = \boldsymbol{x}^2 \tag{12}$$

is considered. For this simple polynomial transformation, equations (9)–(11) can be solved analytically exactly. The mean μ^{y}, the variance C^{y}, and the cross-variance $C^{x,y}$ are given by

$$\begin{aligned}\mu^{y} &= \left(\mu^{x}\right)^2 + C^{x}\,,\\ C^{y} &= 2\cdot C^{x}\cdot\left(C^{x}+2\cdot\left(\mu^{x}\right)^2\right),\\ C^{x,y} &= 2\cdot C^{x}\cdot\mu^{x}\,,\end{aligned} \tag{13}$$

respectively.

It is worth mentioning that for linear transformations $\underline{\boldsymbol{y}} = \mathbf{A}\cdot\underline{\boldsymbol{x}}$, forward and backward inference correspond to the prediction and filter step of the well-known Kalman filter. That is, the Kalman filter is a special case of an AGF.

4 Sample-Based Gaussian Assumed Density Filter (SGF)

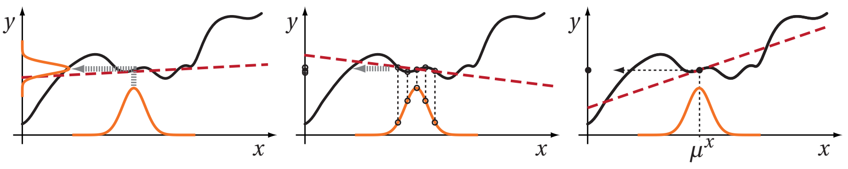
For nonlinear functions $\underline{g}(\cdot)$ that prevent a closed-form solution, the so-called linear regression Kalman filters (LRKFs) or sample-based Gaussian assumed density filters (SGFs) allow approximately calculating the first two moments of $\underline{\boldsymbol{y}}$. Examples for LRKFs/SGFs are the well-known unscented Kalman filter (UKF) [7], the Divided Difference Filter [12], or the Gaussian Filter [3]. These filters utilize a sample representation of $\underline{\boldsymbol{x}}$ given by the Dirac mixture

$$f(\underline{x}) \approx \sum_{i=1}^{L} w_i \cdot \delta\Big(\underline{x} - \underline{\mu}_i\Big) , \tag{14}$$

where L is the number of samples, w_i are non-negative weighting factors, and $\underline{\mu}_i$ are the sample positions. The Dirac mixture (14) exactly captures the mean $\underline{\mu}^x$ and the covariance $\mathbf{C}^x$ of $\underline{\boldsymbol{x}}$. Propagating the samples through (5) corresponds to approximating (9) and (10) by a weighted sample mean and sample covariance, respectively. In case of the mean $\underline{\mu}^y$, replacing the density $f(\underline{x})$ in (9) by the Dirac mixture (14) leads to

$$\underline{\mu}^y \approx \int \underline{g}(\underline{x}) \cdot \left(\sum_i w_i \cdot \delta\Big(\underline{x} - \underline{\mu}_i\Big) \right) \mathrm{d}\underline{x} = \sum_i w_i \cdot \underline{g}\Big(\underline{\mu}_i\Big) . \tag{15}$$

The quality of this mean approximation (and similarly of the covariance approximation) depends on both the function $\underline{g}(\underline{x})$ and the quality of the sample representation of $f(\underline{x})$ in (14). Especially in cases of strong nonlinearities and a small number L of sample points, (15) provides a poor approximation of the true mean $\underline{\mu}^y$.



(a) Analytic stochastic linearization (AGF) **(b)** Sample-based stochastic linearization (SGF) **(c)** Taylor-series based linearization (EKF)

Figure 1: Illustration of the different linearization approaches: the nonlinear function (black) and its linearized versions (red dashed). (a) AGF propagates the entire density $f(\underline{x})$ for linearization. (b) The linearization of an SGF is based on an approximate sample representation of $f(\underline{x})$. (c) The EKF linearizes the nonlinear function around a single point.

Example 2: Quadratic Transformation (cont'd)

We again consider the quadratic transform (12) of Example 1. The density $f(x)$ is approximately represented by the samples

$$\mu_1 = \mu^x + \sqrt{C^x}\,, \quad \mu_2 = \mu^x - \sqrt{C^x}\,, \quad w_1 = w_2 = 0.5\,,$$

which exactly capture the mean μ^x and variance C^x. Propagating the samples through the quadratic transformation and calculating the weighted mean and variance yields

$$\begin{aligned} \mu^y &= \left(\mu^x\right)^2 + C^x\,, \\ C^y &= 0.5 \sum_{i=1}^{2} \left(\left(\mu_i\right)^2 - \mu^y\right)^2 = 4 \cdot C^x \cdot \left(\mu^x\right)^2\,, \\ C^{x,y} &= 2 \cdot C^x \cdot \mu^x\,. \end{aligned}$$

Comparison with (13) shows that the sample-based variance C^y is not exact.

Calculating the mean and covariance of $\underline{\boldsymbol{y}}$ corresponds to a linearization of the nonlinear transformation $\underline{g}(\cdot)$ as the density functions $f(\underline{x})$ and $f(\underline{y})$ are both assumed to be Gaussian (see Figure 1). In case of the AGF, linearization is performed implicitly under consideration of the entire Gaussian density $f(\underline{x})$. In contrast to this, SGFs merely propagate a sample-based approximation of $f(\underline{x})$. Even if the mean and covariance of $\underline{\boldsymbol{x}}$ are captured exactly by the samples, the same is not true for (all) higher-order moments due to the finite number of samples. But again, linearization is performed implicitly due to the calculation of the sample mean and sample covariance (see for example equation (15)). The extended Kalman filter (EKF, [15]) is an example of a Gaussian filter, where linearization is done explicitly by first-order Taylor-series expansion of $\underline{g}(\cdot)$ around the mean $\underline{\mu}^x$. Here, linearization is only performed at a single point (the mean of $\underline{\boldsymbol{x}}$) and no uncertainty information about $\underline{\boldsymbol{x}}$ is considered. This typically leads to an inferior performance in comparison with SGFs or the AGF. Merely in case of a linear transformation, AGF, SGF, and EKF provide identical results. The same is true in case of AGF and SGF, if an infinite number of samples is used.

5 Semi-Analytic Gaussian Assumed Density Filter (SAGF)

To attenuate the drawbacks of a purely sample-based approximation of the mean and covariance, the key idea of the proposed semi-analytic Gaussian assumed density filter (SAGF) is to combine the sample-based Gaussian assumed density filter (SGF) and the analytic Gaussian assumed density filter (AGF) such that only some dimensions of the random vector $\underline{\boldsymbol{x}}$ are discretized by means of a sample representation. Thus, only some parts of the nonlinear transformation (5) have to be evaluated approximately. For this purpose, we rearrange the nonlinear equation (5) to

$$\underline{\boldsymbol{y}} = \underline{g}\big(\underline{\boldsymbol{x}}^a, \underline{\boldsymbol{x}}^b\big) \,, \tag{16}$$

where the Gaussian random vector $\underline{\boldsymbol{x}} = \left[\big(\underline{\boldsymbol{x}}^a\big)^{\mathrm{T}}, \big(\underline{\boldsymbol{x}}^b\big)^{\mathrm{T}}\right]^{\mathrm{T}}$ consists of the substates $\underline{\boldsymbol{x}}^a$ and $\underline{\boldsymbol{x}}^b$ with mean and covariance

$$\underline{\mu}^x = \begin{bmatrix} \underline{\mu}^a \\ \underline{\mu}^b \end{bmatrix} , \quad \mathbf{C}^x = \begin{bmatrix} \mathbf{C}^a & \mathbf{C}^{a,b} \\ \mathbf{C}^{b,a} & \mathbf{C}^b \end{bmatrix} .$$

Generally, there exists no closed-form expression for the desired moments. However, the decomposition into $\underline{\boldsymbol{x}}^a$ and $\underline{\boldsymbol{x}}^b$ is chosen in such a way that the integrals in (9)–(11) can be calculated in closed form for any given fixed value of $\underline{\boldsymbol{x}}^b$. Hence, we say that $\underline{g}(\cdot,\cdot)$ is *conditionally integrable*. For determining a sample-based representation of $\underline{\boldsymbol{x}}^b$, the sampling techniques of the SGFs are applied.

Example 3: Conditionally Integrable Nonlinear Transformation

In the following, two example transformations are discussed. At first, a nonlinear transformation with conditionally linear function $g(\cdot,\cdot)$ is given by

$$\boldsymbol{y} = \mathrm{e}^{-\boldsymbol{x}_1^b} \cdot \left(\boldsymbol{x}_2^b\right)^2 \cdot \boldsymbol{x}^a \ .$$

To see this, we replace the substate $\underline{\boldsymbol{x}}^b = \left[\boldsymbol{x}_1^b, \boldsymbol{x}_2^b\right]^{\mathrm{T}}$ with a single sample point $\underline{\mu} = \left[\mu_1, \mu_2\right]^{\mathrm{T}}$, which leads to $\boldsymbol{y} = c \cdot \boldsymbol{x}^a$, where $c \triangleq \mathrm{e}^{-\mu_1} \cdot (\mu_2)^2$. Hence, given the sample point $\underline{\mu}$, the first two moments of $\boldsymbol{y}$ can be calculated via the Kalman predictor. The scope of this paper is even more general. Also decompositions into conditionally integrable *nonlinear* transformations $g(\cdot,\cdot)$ are covered. For the example equation

$$\boldsymbol{y} = \mathrm{e}^{-\boldsymbol{x}^b} \cdot \left(\boldsymbol{x}_1^a\right)^2 \cdot \boldsymbol{x}_2^a \ ,$$

the function $g\left(\underline{\boldsymbol{x}}^a, \mu\right) = c \cdot \left(\boldsymbol{x}_1^a\right)^2 \cdot \boldsymbol{x}_2^a$ with $c \triangleq \mathrm{e}^{-\mu}$ is now nonlinear when replacing $\boldsymbol{x}^b$ by a sample point μ. However, $g(\cdot, \mu)$ is polynomial and thus, the moments can be calculated analytically since $\underline{\boldsymbol{x}}^a$ is Gaussian distributed.

The AGF (Section 3) and SGF (Section 4) are extreme cases of the SAGF: if $\underline{\boldsymbol{x}}^b$ is an empty vector, SAGF becomes an AGF and if $\underline{\boldsymbol{x}}^a$ is an empty vector, SAGF degenerates to an SGF.

5.1 General Solution

For the general transformation given by (16), the desired moments of $\underline{\boldsymbol{y}}$ can be calculated as follows. At first, the joint density $f(\underline{x}, \underline{y})$ is separated by employing Bayes' rule

$$f(\underline{x}, \underline{y}) = \underbrace{\delta\left(\underline{y} - \underline{g}(\underline{x}^a, \underline{x}^b)\right)}_{= f(\underline{y}|\underline{x})} \cdot \underbrace{f(\underline{x}^a|\underline{x}^b) \cdot f(\underline{x}^b)}_{= f(\underline{x})} ,$$

where the density $f(\underline{x})$ is replaced by $f(\underline{x}^a|\underline{x}^b) \cdot f(\underline{x}^b)$ and the conditional density $f(\underline{x}^a|\underline{x}^b) = \mathcal{N}(\underline{x}^a; \underline{\mu}^{a|b}, \mathbf{C}^{a|b})$ is (conditionally) Gaussian with mean and covariance

$$\begin{aligned} \underline{\mu}^{a|b} &= \underline{\mu}^a + \mathbf{C}^{a,b} \cdot \left(\mathbf{C}^b\right)^{-1} \cdot \left(\underline{x}^b - \underline{\mu}^b\right) , \\ \mathbf{C}^{a|b} &= \mathbf{C}^a - \mathbf{C}^{a,b} \cdot \left(\mathbf{C}^b\right)^{-1} \cdot \mathbf{C}^{b,a} . \end{aligned} \tag{17}$$

For determining the mean $\underline{\mu}^y$ in (9), the density $f(\underline{x}^b)$ of the substate $\underline{\boldsymbol{x}}^b$ is represented by means of a Dirac mixture as in (14) in order to allow applying an SGF. To integrate over $\underline{x}^b$, the sifting property of the Dirac delta distribution is exploited. Hence, the mean of $\underline{\boldsymbol{y}}$ is given by

$$\underline{\mu}^y \approx \sum_i w_i \cdot \underline{\mu}_i^y \quad \text{with} \quad \underline{\mu}_i^y = \int \underline{g}\left(\underline{x}^a, \underline{\mu}_i\right) \cdot f\left(\underline{x}^a|\underline{\mu}_i\right) \mathrm{d}\underline{x}^a . \tag{18}$$

Analogously, the covariance of $\underline{\boldsymbol{y}}$ results in

$$\begin{aligned} \mathbf{C}^y &\approx \sum_i w_i \cdot \left(\mathbf{C}_i^y - \underline{\mu}_i^y (\underline{\mu}^y)^{\mathrm{T}} - \underline{\mu}^y \left(\underline{\mu}_i^y\right)^{\mathrm{T}} + \underline{\mu}^y (\underline{\mu}^y)^{\mathrm{T}}\right) , \\ \mathbf{C}_i^y &= \int \underline{g}\left(\underline{x}^a, \underline{\mu}_i\right) \cdot \underline{g}\left(\underline{x}^a, \underline{\mu}_i\right)^{\mathrm{T}} \cdot f\left(\underline{x}^a|\underline{\mu}_i\right) \mathrm{d}\underline{x}^a . \end{aligned} \tag{19}$$

It is important to note that the integrals in (18) and (19) can be evaluated analytically as the function $\underline{g}(\cdot,\cdot)$ is chosen to be conditionally integrable. Furthermore, solving these integrals is an off-line task and the solution is characterized by a parametric representation for efficient on-line evaluation.

5.2 Estimation

With the results of the previous section, a complete SAGF consisting of a prediction and a filter step is now derived.

Prediction Step

In the prediction step, the predicted mean $\underline{\mu}_{k+1}^{p}$ and covariance $\mathbf{C}_{k+1}^{p}$ of $f^{p}(\underline{x}_{k+1})$ for time step $k+1$ have to be calculated. For this purpose, the system function (1) can be directly mapped to the nonlinear transformation (16) according to

$$\underline{\boldsymbol{x}}_{k+1} = \underline{a}_k\left(\underline{\boldsymbol{x}}_k, \underline{u}_k, \underline{\boldsymbol{w}}_k\right) = \underline{g}\left(\underline{\boldsymbol{x}}_k^a, \underline{\boldsymbol{x}}_k^b\right) .$$

Here, the (deterministic) system input $\underline{u}_k$ becomes a part of the function $\underline{g}(\cdot,\cdot)$ and the substates $\underline{\boldsymbol{x}}_k^a, \underline{\boldsymbol{x}}_k^b$ are augmented with the noise variables $\underline{\boldsymbol{w}}_k^a, \underline{\boldsymbol{w}}_k^b$, where $\underline{\boldsymbol{w}}_k^{\mathrm{T}} = \left[(\underline{\boldsymbol{w}}_k^a)^{\mathrm{T}}, (\underline{\boldsymbol{w}}_k^b)^{\mathrm{T}}\right]$, in order to consider additive and/or multiplicative noise.

The sample points of substate $\underline{\boldsymbol{x}}^b$ are calculated based on the sampling scheme of the used SGF. For the mean $\underline{\mu}_{k+1}^{p}$ and covariance $\mathbf{C}_{k+1}^{p}$ of $\underline{\boldsymbol{x}}_{k+1}$, (18) and (19) are employed.

Filter Step

The measurement equation (2) is mapped to the nonlinear transformation (16) according to

$$\underline{\boldsymbol{y}}_k = \underline{h}_k(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{v}}_k) = \underline{g}\left(\underline{\boldsymbol{x}}_k^a, \underline{\boldsymbol{x}}_k^b\right) ,$$

where the measurement noise $\underline{\boldsymbol{v}}_k$ is part of the substates $\underline{\boldsymbol{x}}_k^a, \underline{\boldsymbol{x}}_k^b$. It is worth mentioning that the decomposition of $\underline{\boldsymbol{x}}_k$ into the substates for the filter step is independent of the decomposition of the prediction step.

The goal of the filter step is to determine the mean $\underline{\mu}_k^e$ and covariance $\mathbf{C}_k^e$ of the estimated density $f^e(\underline{x}_k) = \mathcal{N}(\underline{x}_k; \underline{\mu}_k^e, \mathbf{C}_k^e)$ by using (6), where $\underline{\mu}_k^y$ and $\mathbf{C}_k^y$ of $\underline{\boldsymbol{y}}_k$ correspond to (18) and (19), respectively. The cross-covariance $\mathbf{C}^{x,y} = \left[\mathbf{C}^{a,y}, \mathbf{C}^{b,y}\right]^{\mathrm{T}}$ in (6) consists of

$$\mathbf{C}^{a,y} = \sum_i w_i \cdot \left(\mathbf{C}_i^{a,y} - \underline{\mu}_i^{a|b}\left(\underline{\mu}^y\right)^{\mathrm{T}} + \underline{\mu}^a\left(\underline{\mu}^y - \underline{\mu}_i^y\right)^{\mathrm{T}}\right) ,$$

$$\mathbf{C}^{b,y} = \sum_i w_i \cdot \left(\underline{\mu}_i - \underline{\mu}^b\right) \cdot \left(\underline{\mu}_i^y - \underline{\mu}^y\right)^{\mathrm{T}} ,$$

with

$$\mathbf{C}_i^{a,y} = \int \underline{x}^a \cdot \left(\underline{g}\left(\underline{x}^a, \underline{\mu}_i\right)\right)^{\mathrm{T}} \cdot f\left(\underline{x}^a | \underline{\mu}_i\right) \mathrm{d}\underline{x}^a .$$

Here, $\underline{\mu}_i^y$ results from (18) and $\underline{\mu}_i^{a|b}$ is calculated according to (17) by replacing $\underline{x}^b$ with $\underline{\mu}_i$.

6 Simulation Examples

The proposed approach is now compared with two sample-based estimators, i.e., the unscented Kalman filter (UKF, [7]) and the Gaussian particle filter (GPF, [9]). The UKF makes use of a deterministic sampling method and assumes that state and measurement are jointly Gaussian. The GPF is a special sequential importance sampling Particle Filter. Here, after each prediction and filtering, the *randomly* drawn samples are used for determining the mean and the covariance of the state. Thus, a Gaussian representation is provided after each step and no resampling is necessary. As any Particle Filter, the GPF does not require the assumption of a jointly Gaussian state and measurement.

6.1 System Equation

In the simulations, the estimation of the altitude $\boldsymbol{\alpha}_k$, velocity $\boldsymbol{\beta}_k$, and constant ballistic coefficient $\boldsymbol{\gamma}_k$ of a falling body is considered [6, 15]. The system equation is given by

$$\underline{\boldsymbol{x}}_{k+1} = \begin{bmatrix} \boldsymbol{\alpha}_k \\ \boldsymbol{\beta}_k \\ \boldsymbol{\gamma}_k \end{bmatrix} + \Delta_t \begin{bmatrix} -\boldsymbol{\beta}_k \\ -\mathrm{e}^{-\rho \cdot \boldsymbol{\alpha}_k} \cdot (\boldsymbol{\beta}_k)^2 \cdot \boldsymbol{\gamma}_k \\ 0 \end{bmatrix} + \begin{bmatrix} \boldsymbol{w}_k^\alpha \\ \boldsymbol{w}_k^\beta \\ \boldsymbol{w}_k^\gamma \end{bmatrix} ,$$

where $\underline{\boldsymbol{x}}_k = [\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k, \boldsymbol{\gamma}_k]^{\mathrm{T}}$ is the state vector, Δ_t the discretization constant, ρ a constant factor, and $\boldsymbol{w}_k^\alpha$, $\boldsymbol{w}_k^\beta$, $\boldsymbol{w}_k^\gamma$ are process noise. The discretization constant Δ_t is set to 1 and the constant factor ρ is $5 \cdot 10^{-5}$. The noises $\boldsymbol{w}_k^\alpha$, $\boldsymbol{w}_k^\beta$, $\boldsymbol{w}_k^\gamma$ are zero-mean Gaussian with joint covariance matrix $\mathbf{Q} = 0.1 \cdot \mathbf{I}_3$, where $\mathbf{I}_n$ is the $n \times n$

identity matrix. The initial state of the falling body is $\underline{x}_0^{\mathrm{T}} = \left[3\cdot 10^5, 2\cdot 10^4, 10^{-3}\right]$. The initial mean and covariance of the estimators for all simulation runs is set to

$$\underline{\mu}^x = \begin{bmatrix} 3\cdot 10^5 \\ 2\cdot 10^4 \\ 10^{-5} \end{bmatrix} \quad , \quad \mathbf{C}^x = \begin{bmatrix} 10^6 & 0 & 0 \\ 0 & 4\cdot 10^6 & 0 \\ 0 & 0 & 20 \end{bmatrix} .$$

The state variables can be decomposed into $\underline{\boldsymbol{x}}_k^a = \left[\boldsymbol{\beta}_k, \boldsymbol{\gamma}_k\right]^{\mathrm{T}}$ and $\boldsymbol{x}_k^b = \boldsymbol{\alpha}_k$. If the density of the variable $\boldsymbol{x}_k^b$ is represented by a Dirac mixture, the remaining moment integrals in (18) and (19) can be calculated in closed form due to the remaining polynomial system function (see Example 3).

6.2 Case I: Linear Measurement Equation

In the first case, a linear measurement equation is considered, where the altitude is measured directly according to

$$\boldsymbol{r}_k = \boldsymbol{\alpha}_k + \boldsymbol{v}_k ,$$

where $\boldsymbol{v}_k$ is zero-mean Gaussian measurement noise with variance $R = 10^3$. Due to the linearity of the measurement equation, all three estimators (SAGF, UKF, and GPF) are solving the filter step via the Kalman corrector equations.

6.3 Case II: Nonlinear Measurement Equation

In the second case, a radar measures the altitude, where the measurement equation is given by

$$\boldsymbol{r}_k = \sqrt{M^2 + (\boldsymbol{\alpha}_k - H)^2 + (\boldsymbol{v}_k)^2} . \tag{20}$$

$M = 10^4$ is the horizontal range and $H = 10^4$ is the altitude of the radar. The measurement noise $\boldsymbol{v}_k$ is Gaussian distributed with the variance $R = 10^3$.

In order to allow a closed-form solution of the filter step in case of the SAGF, squared ranges are considered. Thus, the measurement equation becomes polynomial according to

$$\boldsymbol{r}_k^2 = M^2 + (\boldsymbol{\alpha}_k - H)^2 + (\boldsymbol{v}_k)^2 , \tag{21}$$

and can now be used in the AGF formalism. In doing so, the measurement $\underline{\hat{y}}$ in equation (5) is given by $\hat{y} = \hat{r}_k^2$, where $\hat{r}_k$ is the measured altitude value.

6.4 Simulation Results

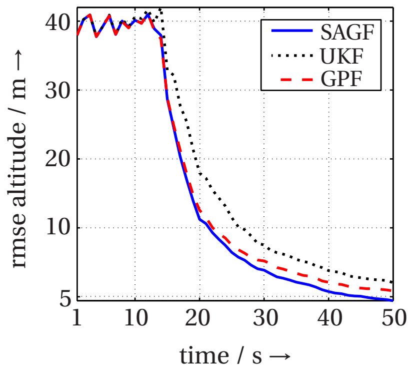
For each case, 1000 simulation runs are performed. In the first case (linear measurement equation), the SAGF is compared with the UKF as well as with the GPF with 100 and 1000 particles. In the second case (nonlinear measurement equation), the SAGF is compared with the UKF. The GPF is omitted, as it provides no reliable estimates in this scenario due to the non-additive noise term in (20).

Case I

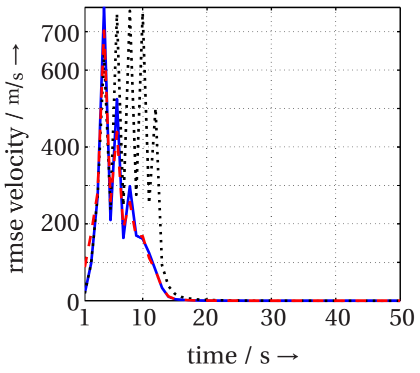
In Figure 2, the root mean square error (rmse) over the 1000 simulation runs is shown. It is obvious that the proposed SAGF converges faster than the UKF. Compared to the GPF with 100 particles, the SAGF has a smaller error in the altitude. Furthermore, the average rmses of the SAGF and GPF are smaller than the error of the UKF (see Table 1). In terms of run time, the SAGF is two times faster than the GPF with 100 particles and four times faster than the UKF.

Case II

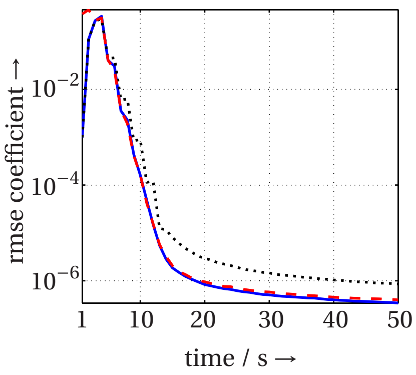
The differences between the two estimators, SAGF and UKF, significantly increase if the altitude is measured according to the nonlinear measurement equation (20). In Figure 3 the rmses are shown. At 10 seconds, the error for the altitude and the velocity is increasing due to the drag of the nonlinear motion [6]. This can be also seen in the average rmse in Table 2. It is important to note that applying the UKF to measurement equation (20) or its modified version (21) has no notable impact on the results.



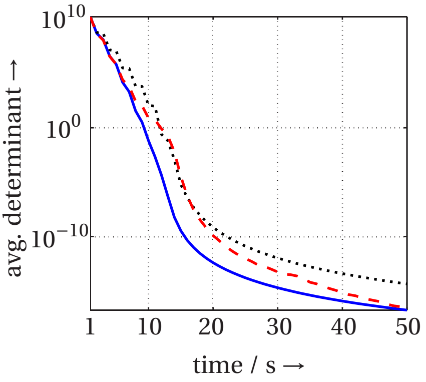
(a) Altitude.



(b) Velocity.



(c) Ballistic coefficient.

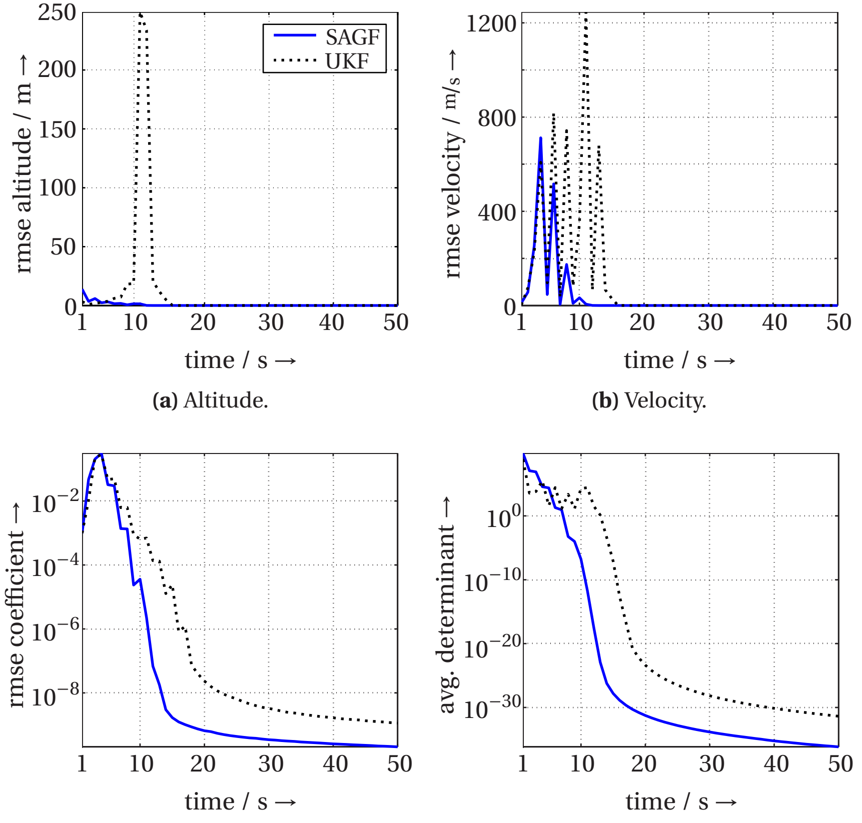


(d) Covariance determinant.

Figure 2: Rmse over 1000 test runs for case I. For GPF 100 particles are used.

Table 1: Average rmse and its standard deviation over all test runs for case I.

	Altitude	Velocity	Ballistic coefficient
SAGF	**12.6** ± **8.3**	**59.3** ± **143.7**	**0.016** ± **0.063**
UKF	14.2 ± 7.9	100.1 ± 212.4	**0.016** ± **0.058**
GPF 100 p.	13.0 ± 8.0	60.2 ± 134.9	0.029 ± 0.098
GPF 1000 p.	**12.7** ± **8.2**	**59.3** ± **142.1**	0.019 ± 0.066



(a) Altitude. **(b)** Velocity. **(c)** Ballistic coefficient. **(d)** Covariance determinant.

Figure 3: Rmse over 1000 test runs for case II.

Table 2: Average rmse and its standard deviation over all test runs for case II.

	Altitude	Velocity	Ballistic Coefficient
SAGF	**0.7 ± 2.2**	**36.5 ± 128.5**	**0.0126 ± 0.052**
UKF	11.7 ± 47.7	102.5 ± 258.9	**0.0127 ± 0.051**

7 Discussion and Future Work

A new framework for the efficient calculation of posterior moments in the context of Gaussian assumed density filtering based on moment matching has been proposed, which exploits the structure of the given nonlinear system. For this purpose, the system is decomposed into a set of nonlinear subsystems that are conditionally integrable in closed form by means of discretizing *some dimensions of the state space*.

For systems of moderate complexity, a suitable decomposition can typically be found by inspection. For large systems, however, automatic methods for the optimal decomposition according to some predefined quality measure are required and will be pursued in future research. Quality measures might include i) the minimum number of samples or ii) the minimum total number of computations for a given estimation accuracy.

References

[1] Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck. Gaussian Filtering using State Decomposition Methods. In *Proceedings of the 12th International Conference on Information Fusion*, Seattle, WA, USA, July 2009.

[2] Uwe D. Hanebeck, Marco F. Huber, and Vesa Klumpp. Dirac Mixture Approximation of Multivariate Gaussian Densities. In *Proceedings of the 2009 IEEE Conference on Decision and Control (CDC)*, Shanghai, China, December 2009.

[3] Marco F. Huber and Uwe D. Hanebeck. Gaussian Filter based on Deterministic Sampling for High Quality Nonlinear Estimation. In *Proceedings of the 17th IFAC World Congress*, volume 17, Seoul, Korea, July 2008.

[4] Kazufumi Ito and Kaiqi Xiong. Gaussian Filters for Nonlinear Filtering Problems. *IEEE Transactions on Automatic Control*, 45(5):910–927, May 2000.

[5] Andrew H. Jazwinski. Filtering for Nonlinear Dynamical Systems. *IEEE Transactions on Automatic Control*, 11(4):765–766, October 1966.

[6] Simon Julier, Jeffrey Uhlmann, and Hugh F. Durrant-Whyte. A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, 2000.

[7] Simon J. Julier and Jeffrey K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[8] Rudolf E. Kalman. A new Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME, Journal of Basic Engineering*, 82 (Series D)(1):35–45, 1960.

[9] Jayesh H. Kotecha and Petar M. Djurić. Gaussian Particle Filtering. *IEEE Transactions on Signal Processing*, 51(10):2592–2601, October 2003.

[10] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter. *Nonlinear Kalman Filtering for Force-Controlled Robot Tasks*. Springer, 2005.

[11] Mark R. Morelande and Bill Moran. An Unscented Transformation for Conditionally Linear Models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages III–1417–III–1420, April 2007.

[12] Magnus Nørgaard, Niels K. Poulsen, and Ole Ravn. New Developments in State Estimation for Nonlinear Systems. *Automatica*, 36(11):1627–1638, November 2000.

[13] Athanasios Papoulis and S. Unnikrishna Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, 4th edition, 2002.

[14] Thomas Schön, Fredrik Gustafsson, and Per-Johan Nordlund. Marginalized Particle Filters for Mixed Linear/Nonlinear State-Space Models. *IEEE Transactions on Signal Processing*, 53(7):2279–2289, July 2005.

[15] Dan Simon. *Optimal State Estimation: Kalman, H-Infinity, and Nonlinear Approaches*. John Wiley & Sons, Inc., 1st edition, 2006.

Paper C

Chebyshev Polynomial Kalman Filter

Authors: Marco F. Huber

Edited version of the paper:

M. F. Huber. Chebyshev Polynomial Kalman Filter. In *Digital Signal Processing*, vol. 23, no. 5, pages 1620–1629, September 2013.

Chebyshev Polynomial Kalman Filter

Marco F. Huber

AGT International
Darmstadt, Germany
marco.huber@ieee.org

Abstract

A novel Gaussian state estimator named Chebyshev Polynomial Kalman Filter is proposed that exploits the exact and closed-form calculation of posterior moments for polynomial nonlinearities. An arbitrary nonlinear system is at first approximated via a Chebyshev polynomial series. By exploiting special properties of the Chebyshev polynomials, exact expressions for mean and variance are then provided in computationally efficient vector-matrix notation for prediction and measurement update. Approximation and state estimation are performed in a black-box fashion without the need of manual operation or manual inspection. The superior performance of the Chebyshev Polynomial Kalman Filter compared to state-of-the-art Gaussian estimators is demonstrated by means of numerical simulations and a real-world application.

1 Introduction

Closed-form solutions for recursive Bayesian state estimation are merely available in some special cases. For linear systems affected by Gaussian noise, the famous Kalman filter [17] is optimal and allows Bayesian estimation in a very efficient manner. In case of a discrete state space with a finite number of states, the grid-based filters are optimal [5]. For nonlinear non-Gaussian estimation problems being typical in many technical fields, e.g., localization, robotics, or signal processing, closed-form recursive Bayesian estimation is not possible and thus, approximations are inevitable.

A common approximation is the so-called Gaussian assumed density filtering, where the system state is restricted to be Gaussian [30]. Here, the approximation is performed in such a way that the exact mean and variance of the state distribution are preserved via moment matching. This however, requires the solution of expectation integrals, which are not available in closed form in general. Besides computationally expensive numerical integration, many fast approximate Gaussian assumed density filters utilizing the Kalman filter equations have been developed in the past. The extended Kalman filter (EKF, [15, 28]) for instance employs linearization via first-order Taylor series expansion, while the second-order EKF additionally considers the Hessian for improved estimation performance [28]. However, differentiability of the considered nonlinear functions is required and no uncertainty information of the system state for the approximation is taken into account. The unscented Kalman filter (UKF, [16, 31]) instead propagates a finite set of the so-called sigma-points that captures the mean and variance of the prior density. This idea is also exploited for the cubature Kalman filter [3] or the Gauss-Hermite Kalman filter [4, 14], both utilizing quadrature integration points. In [13], the sigma-points are determined based on a shape approximation of the Gaussian distribution, which allows a variable number of sigma-points. Though, all these approximations are rather coarse as only a small set of sigma-points is typically employed. It is not possible to capture higher-order information of the prior density or the nonlinear functions, since only a point representation is propagated instead of the entire Gaussian distribution.

An alternative branch for solving nonlinear estimation problems is the discretization of the state space as done in grid filters or particle filters [5]. Theoretically, these estimation techniques allow approaching the true statistics of the state with arbitrary accuracy. This however comes with the expense of an applicability to low-dimensional problems only as the computational complexity growth exponentially with the dimension of the state space. To attenuate this computational burden, sophisticated techniques like adaptive discretization [9] or Rao-Blackwellization [26] have to be applied. In order to avoid the sample depletion problem of particle filters, the so-called Gaussian particle filter [20] has been proposed, which links sampling-based estimation with Gaussian filtering.

The novel Gaussian estimator proposed in this paper named *Chebyshev Polynomial Kalman Filter* relies on a recent finding that for polynomial nonlinearities, mean and variance can be determined exactly in closed form [12, 22]. In order to apply this finding, a given nonlinear system is in a first step expanded in a series of Chebyshev polynomials as proposed in [33]. This specific type of orthogonal polynomials is well suited for approximation purposes for two reasons. Firstly, the series coefficients can be determined efficiently by means of the discrete cosine transform. Secondly, a Chebyshev series approximation is typically very close to the best approximating polynomial, whereas the best polynomial is hard to find in general.

Given the Chebyshev series expansion of the system, closed-form solutions for moment propagation are derived and expressed in vector-matrix notation, which allows a computationally efficient implementation. These moment propagation expressions are then employed in a Kalman filter framework in order to obtain a recursive Gaussian estimator. Thus, the contributions of this paper are three-fold: (1) Accurate approximation of arbitrary nonlinear models by means of Chebyshev polynomial series expansion. This approximation can be performed automatically and computationally cheap via the discrete cosine transform. (2) Closed-form and exact calculation of the non-central moments of a Gaussian distribution propagated through polynomials. The entire probability mass of the distribution is taken into account to allow for accurate results. (3) Computationally efficient mean and variance propagation for Chebyshev polynomial series by means of exploiting the recursive and sparse structure of Chebyshev polynomials.

The paper is structured as follows. The next section formulates the considered recursive Gaussian state estimation problem. In Section 3, a brief introduction to Chebyshev polynomials and series expansions is provided. Section 4 is concerned with closed-form moment calculation. These results are then employed in Section 5 for deriving the complete Gaussian estimator. This estimator is compared against state-of-the-art estimators based on numerical simulations and a real-world application in Section 6. A discussion on strengths, limitations, and potential extensions of the proposed approach is part of Section 7. The paper closes with a conclusion.

In the following, lower-case bold font letters indicate random variables, upper-case bold font letters indicate matrices, and underlined letters indicate vectors.

2 Problem Formulation

In this paper, the nonlinear discrete-time system with dynamics equation

$$\boldsymbol{x}_{k+1} = a_k(\boldsymbol{x}_k, u_k) + \boldsymbol{w}_k \tag{1}$$

and measurement equation

$$\boldsymbol{z}_k = h_k(\boldsymbol{x}_k) + \boldsymbol{v}_k \tag{2}$$

is considered, where $a_k(.,.)$ and $h_k(.)$ are the known time-variant system function and measurement function, respectively. The scalar random variable $\boldsymbol{x}_k$ is the system state at time step k and $\boldsymbol{z}_k$ is the scalar measurement, where an actual measurement value $\hat{z}_k$ is a realization of the random variable $\boldsymbol{z}_k$. The terms $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are the zero-mean white Gaussian system noise and measurement noise, respectively. The system input u_k is assumed to be known.

In Bayesian estimation, two alternating steps, i.e., prediction and measurement update, are performed for estimating the system state $\boldsymbol{x}_k$. In prediction, the current estimate of $\boldsymbol{x}_k$ is propagated to time step $k+1$ by means of the dynamics equation (1). In the measurement update, a given measurement value $\hat{z}_k$ is incorporated for updating $\boldsymbol{x}_k$ under the consideration of the measurement relation (2).

For both steps, closed-form solutions are not available for arbitrary functions $a_k(.,.)$, $h_k(.)$ and arbitrary distributed random variables. Instead, appropriate approximations have to be applied for practical use. In order to facilitate a computationally efficient approximation, the focus in this paper is on Kalman filter like calculations. For this purpose it is sufficient to consider a nonlinear transformation

$$\boldsymbol{y} = g(\boldsymbol{x}) \, , \tag{3}$$

where the goal is to find the mean and variance of the random variable $\boldsymbol{y}$ from propagating the Gaussian random variable $\boldsymbol{x} \sim \mathcal{N}\big(x; \mu_x, \sigma_x^2\big)$ with mean μ_x and variance σ_x^2. In the prediction step $\boldsymbol{y}$ corresponds to the prediction $\boldsymbol{x}_{k+1}$, while for filtering $\boldsymbol{y}$ coincides with the measurement $\boldsymbol{z}_k$.

For arbitrary nonlinear transformations (3) but Gaussian $\boldsymbol{x}$, a closed-form calculation of the desired moments of $\boldsymbol{y}$ is still not possible in general. An exception are polynomial transformations. Thus, the key idea is to approximate an arbitrary nonlinear transformation (3) by means of a polynomial series and then, to perform the moment calculations of $\boldsymbol{y}$ in closed form. For this purpose, a highly accurate approximation based on Chebyshev polynomials is considered next.

3 Chebyshev Polynomials

In this paper, the focus is on the Chebyshev polynomials of the first kind[1], which are defined compactly via (see [23])

$$T_n(x) = \cos(n \cdot \arccos x)\ , \quad n = 0,1,\ldots$$

or equivalently by means of the recursion

$$T_n(x) = 2x \cdot T_{n-1}(x) - T_{n-2}(x)\ , \quad n = 2,3,\ldots\ , \tag{4}$$

with initial conditions

$$T_0(x) = 1\ , \quad T_1(x) = x\ . \tag{5}$$

It is easy to deduce from (4) that the function $T_n(x)$ is a polynomial of degree n. If n is even (odd), then $T_n(x)$ is a sum of even (odd) monomials, i.e., $T_n(x)$ is of the form

$$T_n(x) = \sum_{i=0}^{n} \alpha_{n,i} \cdot x^i = \sum_{j=0}^{\lfloor n/2 \rfloor} \alpha_{n,n-2j} \cdot x^{n-2j}\ , \tag{6}$$

where $\alpha_{n,i}$ is the *Chebyshev coefficient* of the i-th monomial of the n-th Chebyshev polynomial. The coefficient $\alpha_{n,i}$ is non-zero only if i is even (odd).

3.1 Properties

Besides many other important properties (a detailed description can be found for example in [23]), the product of two Chebyshev polynomials T_i, T_j is of particular interest in this paper. This product can be expressed by means of the sum of two other Chebyshev polynomials according to

$$T_i(x) \cdot T_j(x) = \tfrac{1}{2}\left(T_{i+j}(x) + T_{|i-j|}(x)\right)\ . \tag{7}$$

Furthermore, the zeros for $x \in \Omega \triangleq [-1,1]$ of $T_n(x)$ are

$$x = x_m = \cos(\theta_m)\ , \tag{8}$$

1 The results presented in this paper can easily be applied to Chebyshev polynomials of the second, third, and forth kind.

with

$$\theta_m \triangleq \frac{\pi(m-\frac{1}{2})}{n}$$

for $m = 1,\ldots,n$. All Chebyshev polynomials $T_i(x), i = 0,1,\ldots$ form a *complete orthogonal* system on the interval Ω with respect to the weight

$$w(x) = \left(1-x^2\right)^{-\frac{1}{2}} \ .$$

Thus, they satisfy

$$\langle T_i(x), T_j(x)\rangle \triangleq \int_{-1}^{1} w(x)\cdot T_i(x)\cdot T_j(x)\,\mathrm{d}x \tag{9}$$

$$= \begin{cases} 0\,, & i \neq j \\ \frac{\pi}{2}\,, & i = j \neq 0 \\ \pi\,, & i = j = 0 \end{cases} . \tag{10}$$

An orthogonal system of polynomials is beneficial especially when employed for approximating a nonlinear function as described next.

3.2 Chebyshev Series

Expanding a given function $g(x)$ by a series of Chebyshev polynomials on Ω gives

$$g(x) \approx \sum_{i=0}^{n} c_i \cdot T_i(x)\,, \quad \forall x \in \Omega \tag{11}$$

with the *series coefficients*

$$c_i = \frac{\langle g(x), T_i(x)\rangle}{\langle T_i(x), T_i(x)\rangle} \ . \tag{12}$$

As any series expansion based on orthogonal polynomials, the series coefficients can be calculated independent of each other. Furthermore, since the system of Chebyshev polynomials is complete, the corresponding series converges for $n \to \infty$ with regard to the $\mathcal{L}_2$-norm if $g(x)$ is a piece-wise continuous or an $\mathcal{L}_2$-integrable function, which implies $c_n \to 0$ for $n \to \infty$. By a change of variables, a Chebyshev series can be transformed into a Fourier cosine series and thus, the well-known theory of Fourier series also applies to Chebyshev series.

3.3 Approximate Series Expansion

For practical implementations, it is only possible to expand a given function $g(x)$ in a finite number of series components $n < \infty$, which results in the so-called *truncated* Chebyshev series. Of course, a truncated expansion merely provides an approximate representation of the function $g(x)$. Fortunately, it can be shown that a truncated Chebyshev series yields a *near-minimax approximation*, i.e., a truncated Chebyshev series of degree n is very close to the best possible polynomial approximation of the same degree (see e.g. Chapter 5.5 in [23]). Compared to other orthogonal polynomial systems, a series expansion based on Chebyshev polynomials benefits from the fact that all $T_i(x)$, $i = 0, \ldots, n$ are bounded between ± 1 and are oscillatory functions in Ω. Thus, the error that results from neglecting components of degree higher than n is spread smoothly over the interval Ω. For instance, if the function $g(x)$ has $r+1$ continuous derivatives on Ω, then the approximation error is with $\mathcal{O}(n^{-r})$ for all $x \in \Omega$ and thus, the error decreases rapidly with the degree n. Here, $\mathcal{O}(.)$ is the big O in Landau notation. Furthermore, if $g(x)$ is a polynomial function with degree r, then a truncated Chebyshev series is optimal, i.e., without approximation error, if $n = r$.

For many functions $g(x)$, the series coefficients (12) cannot be calculated in closed form due to the integration in the numerator. In such cases, one can exploit that Chebyshev polynomials $T_i(x)$, $i = 0, \ldots, n-1$ are also orthogonal in a discrete sense on the zeros x_m, $m = 1, \ldots, n$ of $T_n(x)$ (see (8)), i.e.,

$$\sum_{m=1}^{n} T_i(x_m) \cdot T_j(x_m) = \begin{cases} 0\,, & i \neq j \\ \frac{n}{2}\,, & i = j \neq 0 \\ n\,, & i = j = 0 \end{cases} .$$

This *discrete orthogonality* property leads to a very efficient numerical calculation of (12) by means of

$$c_i \approx \frac{2 - \delta_{0,n}}{n} \sum_{m=1}^{n} g(x_m) \cdot T_i(x_m) \tag{13}$$

for $i = 0, 1, \ldots, n$, where $\delta_{i,j}$ is the Kronecker delta being one only if $i = j$ and zero otherwise. By plugging the zeros (8) in the definition (4) of the Chebyshev polynomials, it follows $T_i(x_m) = \cos(i \cdot \theta_m)$ and thus, the approximate calculation (13) coincides with the well-known *discrete cosine transform* (see e.g. [7]) used in image and video compression, for which a plethora of efficient algorithms is available. Similar approximate calculation schemes of the series coefficients are typically not available for other orthogonal polynomial series expansions like the Fourier-Hermite series.

It can be shown that a truncated Chebyshev series of degree n with approximate series coefficients (13) exactly interpolates $g(x)$ in the zeros (8) of $T_n(x)$. Furthermore, it still satisfies the near-minimax property (see e.g. Chapter 6.5 in [23]). While the best polynomial representation of $g(x)$ is typically difficult to obtain, the approximate Chebyshev series is very close to the best solution and thanks to (13) very easy to calculate.

4 Closed-Form Moment Propagation

The numerical calculation (13) of the series coefficients facilitates a polynomial representation (11) of a given nonlinear function $g(x)$ on the interval Ω in a black-box fashion, i.e., without any user interaction. The restriction on the interval Ω is uncritical. By means of the affine transformation

$$x' = \tfrac{2}{b-a} \cdot x - \tfrac{a+b}{b-a} \,, \tag{14}$$

any function $g(x)$ defined on an arbitrary interval $[a, b]$ can be mapped on the interval Ω (see [27]). In doing so, the Gaussian random variable $\boldsymbol{x}$ has to be transformed as well, which yields the transformed mean and variance

$$\begin{aligned}\mu_{x'} &= \tfrac{2}{b-a} \cdot \mu_x - \tfrac{a+b}{b-a} \,, \\ \sigma_{x'}^2 &= \left(\tfrac{2}{b-a}\right)^2 \cdot \sigma_x^2 \,,\end{aligned}$$

respectively. Furthermore, for the calculation of the series coefficients by means of the discrete cosine transform (13), the zeros x_m defined on Ω (see (8)) have to be mapped to the interval $[a,b]$. This can be achieved by inverting (14), which yields

$$x = \tfrac{b-a}{2} \cdot x' + \tfrac{a+b}{2} \,. \tag{15}$$

Thus, in the following it is sufficient to assume that $g(x)$ is defined on Ω and a Chebyshev series representation of $g(x)$ is given. The next step towards a Kalman filter like estimator is the derivation of closed-form calculations of the mean μ_y and variance σ_y^2 of random variable $\boldsymbol{y}$ given a Gaussian random variable $\boldsymbol{x}$. The whole process of moment calculation is summarized in Figure 1 and detailed in the following sections.

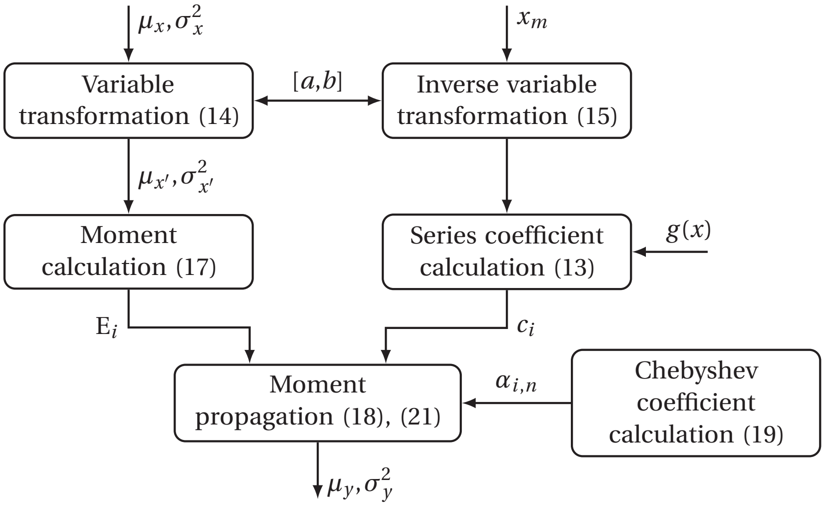

μ_x, σ_x^2
x_m
Variable transformation (14)
$[a,b]$
Inverse variable transformation (15)
$\mu_{x'}, \sigma_{x'}^2$
Moment calculation (17)
Series coefficient calculation (13)
$g(x)$
E_i
c_i
Moment propagation (18), (21)
$\alpha_{i,n}$
Chebyshev coefficient calculation (19)
μ_y, σ_y^2

Figure 1: Flow chart of proposed closed-form moment propagation.

4.1 Non-central Moments of a Gaussian

When propagating a Gaussian $\boldsymbol{x}$ through a Chebyshev series representation of $g(x)$, calculating the mean μ_y can be formulated as

$$\begin{aligned}
\mu_y &= \mathrm{E}\{g(x)\} \\
&= \int g(x) \cdot \mathcal{N}\left(x; \mu_x, \sigma_x^2\right) \mathrm{d}x \\
&\overset{(11)}{\approx} \sum_{i=0}^{n} c_i \int T_i(x) \cdot \mathcal{N}\left(x; \mu_x, \sigma_x^2\right) \mathrm{d}x \\
&\overset{(6)}{=} \sum_{i=0}^{n} c_i \sum_{j=0}^{i} \alpha_{i,j} \underbrace{\int x^j \cdot \mathcal{N}\left(x; \mu_x, \sigma_x^2\right) \mathrm{d}x}_{= \mathrm{E}\{x^j\} \triangleq \mathrm{E}_j} \,,
\end{aligned} \tag{16}$$

i.e., it requires a closed-form calculation of all non-central moments E_j of a Gaussian random variable up to order n. It is important to note that integrating over the entire real line instead of integrating over Ω in the above equation is only valid, if Ω contains almost the complete support of the Gaussian $\boldsymbol{x}$. This can be ensured if the interval $[a,b]$ used in the aforementioned transformation (15) is chosen sufficiently wide (for a detailed discussion see Section 5.1).

In the following, a computationally very efficient scheme for a closed-form calculation of all moments up to order n is proposed, which exploits the well-known fact that the moments of order three and higher of a Gaussian random variable merely depend on its first two moments. For the j-th non-central moment, the recursion

$$
\begin{aligned}
\mathrm{E}_j &= \int x^j \cdot \mathcal{N}\left(x;\mu_x,\sigma_x^2\right) \mathrm{d}x \\
&= \int x^j \frac{1}{\sqrt{2\pi}\sigma_x} \exp\left(-\frac{\left(x-\mu_x\right)^2}{2\sigma_x^2}\right) \mathrm{d}x \\
&\overset{(a)}{=} \underbrace{\frac{x^{j+1}}{j+1} \cdot \mathcal{N}\left(x;\mu_x,\sigma_x^2\right)\Big|_{-\infty}^{\infty}}_{=0} - \int \frac{x^{j+1}}{j+1} \left(\eta_1 + \eta_2 \cdot x\right) \cdot \mathcal{N}\left(x;\mu_x,\sigma_x^2\right) \mathrm{d}x \\
&\overset{(b)}{=} -\frac{1}{j+1}\eta_1 \,\mathrm{E}_{j+1} - \frac{1}{j+1}\eta_2 \,\mathrm{E}_{j+2} \,,
\end{aligned}
$$

holds, with $\eta_1 \triangleq \frac{\mu_x}{\sigma_x^2}$ and $\eta_2 \triangleq -\frac{1}{\sigma_x^2}$. In detail, (a) follows from partial integration and in (b) the linearity of the expected value and the moment definition is exploited.

Thus, by commencing the recursion from $\mathrm{E}_0 = 1$ and $\mathrm{E}_1 = \mu_x$, calculating all higher-order moments from E_2 up to E_n can be described by means of the linear system of equations

$$
\mathbf{R} \cdot \begin{bmatrix} \mathrm{E}_2 \\ \vdots \\ \mathrm{E}_n \end{bmatrix} = \left[\mathrm{E}_0 + \eta_1 \cdot \mathrm{E}_1 \,,\; \mathrm{E}_1,\; \underbrace{0,\,\ldots,\,0}_{n-3 \text{ times}}\right]^{\mathrm{T}} , \tag{17}
$$

where $\mathbf{R}$ is an $(n-1) \times (n-1)$ matrix with entries

$$
R_{ij} = \begin{cases} -\frac{1}{i}\eta_2 & , i = j \\ -\frac{1}{i}\eta_1 & , i = j+1 \\ 1 & , i = j+2 \\ 0 & , \text{otherwise} \end{cases} ,
$$

i.e, the matrix **R** is triangular with zeros everywhere except of the main diagonal and the two diagonals below the main diagonal. Thanks to this specific structure of **R**, the linear system of equations (17) can be solved efficiently by means of forward substitution with complexity $\mathcal{O}(n)$.

It is worth mentioning that the proposed efficient calculation of all higher-order moments is universally valid and thus, can be employed directly for any polynomial series expansion.

4.2 Efficient Mean Propagation

Given the solution of (17), the desired mean μ_y of a propagated Gaussian random variable in (16) can be easily calculated via

$$\mu_y \approx \underline{c}_n^{\mathrm{T}} \cdot \mathbf{A}_n \cdot \underline{\mathrm{E}}_n \,, \tag{18}$$

where $\underline{\mathrm{E}}_n \triangleq [\mathrm{E}_0, \mathrm{E}_1, \ldots, \mathrm{E}_n]^{\mathrm{T}}$ is the vector of non-central moments up to and including order n and $\underline{c}_n \triangleq [c_0, c_1, \ldots, c_n]^{\mathrm{T}}$ is the vector of series coefficients. Further, $\mathbf{A}_n$ is the $(n+1) \times (n+1)$ matrix of Chebyshev coefficients defined by

$$\mathbf{A}_n \triangleq \begin{bmatrix} \underline{\alpha}_{0,n} & \underline{\alpha}_{1,n} & \cdots & \underline{\alpha}_{n,n} \end{bmatrix}^{\mathrm{T}} .$$

Here, $\underline{\alpha}_{i,n} \triangleq [\alpha_{i,0}, \alpha_{i,1}, \ldots, \alpha_{i,n}]^{\mathrm{T}} \in \mathbb{N}^{n+1}$, $i = 0, 1, \ldots, n$ comprises all coefficients of the i-th Chebyshev polynomial up to and including the n-th monomial. It is calculated via the recursion

$$\underline{\alpha}_{i,n} = 2 \cdot \Big[0 \;\; \underline{\alpha}_{i-1,i-1}^{\mathrm{T}} \;\; \underbrace{0 \;\ldots\; 0}_{n-i \text{ times}} \Big]^{\mathrm{T}} + \Big[\underline{\alpha}_{i-2,i-2}^{\mathrm{T}} \;\; \underbrace{0 \;\ldots\; 0}_{n-i+2 \text{ times}} \Big]^{\mathrm{T}} , \tag{19}$$

where the recursion commences from

$$\underline{\alpha}_{0,n} = \begin{bmatrix} 1 & 0 & \ldots & 0 \end{bmatrix}^{\mathrm{T}} , \quad \underline{\alpha}_{1,n} = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \end{bmatrix}^{\mathrm{T}}$$

and exploits the definition of the Chebyshev polynomials (4). According to (5), the coefficients $\alpha_{i,j}$ are zero for $j > i$. Thus, $\mathbf{A}_n$ is a sparse lower triangular matrix, which significantly reduces the computations of the matrix-vector products in (18).

4.3 Efficient Variance Propagation

In order to determine the variance σ_y^2 of the propagated random variable $\boldsymbol{y}$, the relation

$$\begin{aligned}\sigma_y^2 &= \mathrm{E}\left\{(y-\mu_y)^2\right\} = \mathrm{E}\left\{(g(x)-\mu_y)^2\right\} \\ &= \mathrm{E}\left\{g(x)^2\right\} - \mu_y^2 \end{aligned} \tag{20}$$

is exploited, where the propagated mean μ_y is already known from the previous section. It remains to evaluate the first term in (20). Assuming again that $g(x)$ is approximated by means of a Chebyshev series according to (11), evaluating the expectation can be boiled down to calculating multiple expected values over products of Chebyshev polynomials. For an efficient computation, the relation (7) is utilized. In doing so, the expectation $\mathrm{E}\left\{g(x)^2\right\}$ can be determined according to

$$\begin{aligned}\mathrm{E}\left\{g(x)^2\right\} &= \int g(x)\cdot g(x)\cdot \mathcal{N}\left(x;\mu_x,\sigma_x^2\right)\mathrm{d}x \\ &\approx \sum_{i=0}^{n}\sum_{j=0}^{n} c_i\cdot c_j\cdot \int T_i(x)\cdot T_j(x)\cdot \mathcal{N}\left(x;\mu_x,\sigma_x^2\right)\mathrm{d}x \\ &= \left(\underline{c}_n\otimes\underline{c}_n\right)^{\mathrm{T}}\cdot \mathbf{P}_{2n}\cdot \underline{\mathrm{E}}_{2n}\,, \end{aligned} \tag{21}$$

with $\otimes$ being the Kronecker product, $\underline{\mathrm{E}}_{2n}$ comprising all non-central moments up to order $2n$ calculated according to (17), and $\mathbf{P}_{2n}$ being an $(n+1)^2\times(2n+1)$ matrix

$$\mathbf{P}_{2n} \triangleq \frac{1}{2}\cdot \begin{bmatrix} \left(\underline{\alpha}_{0,2n}+\underline{\alpha}_{0,2n}\right)^{\mathrm{T}} \\ \left(\underline{\alpha}_{1,2n}+\underline{\alpha}_{1,2n}\right)^{\mathrm{T}} \\ \vdots \\ \left(\underline{\alpha}_{i+j,2n}+\underline{\alpha}_{|i-j|,2n}\right)^{\mathrm{T}} \\ \vdots \\ \left(\underline{\alpha}_{2n,2n}+\underline{\alpha}_{0,2n}\right)^{\mathrm{T}} \end{bmatrix}$$

comprising the coefficients resulting from all possible products $T_i(x)\cdot T_j(x)$, $i,j=0,1,\ldots,n$ of the Chebyshev series expansion of $g(x)$.

5 The Gaussian Estimator

Based on the previous derivations, the next step is to present a Gaussian filter based on Chebyshev series expansions of the dynamics model (1) and measurement model (2).

5.1 Approximation Interval

Since both the system function and the measurement function can be defined on arbitrary intervals, it is necessary to transform them on the interval Ω via (14) prior to performing the Chebyshev series expansion and the prediction or measurement update. For this purpose, the interval $[a,b]$ has to be determined. One option is to keep this interval fixed for all times steps. This for example is reasonable if it is known a priori that the system state $\boldsymbol{x}_k$ will remain within $[a,b]$ for all times. Here, for time-invariant systems, the series coefficients c_i have to be determined initially only once and remain constant for all times. Admittedly, the interval $[a,b]$ and thus, the number of series components n can be large, which may cause high computational loads for state estimation.

Another option is to determine the interval $[a,b]$ dynamically at each time step based on the current result of the prediction $\boldsymbol{x}_k^p$ and measurement update $\boldsymbol{x}_k^e$, respectively. Given the estimate $\boldsymbol{x}_k^\bullet \sim \mathcal{N}\left(x_k; \mu_k^\bullet, (\sigma_k^\bullet)^2\right)$, the interval borders can be determined according to

$$a_k^\bullet = \mu_k^\bullet - l \cdot \sigma_k^\bullet\,, \quad b_k^\bullet = \mu_k^\bullet + l \cdot \sigma_k^\bullet\,,$$

with $l \in \mathbb{N}_+$ and $\bullet \in \{e,p\}$. In doing so, the interval $[a_k^\bullet, b_k^\bullet]$ corresponds to the l-sigma bound of the state estimate $\boldsymbol{x}_k^\bullet$. Typical choices for l are three and higher. This ensures that 99.7% and more of the probability mass of the estimate $\boldsymbol{x}_k^\bullet$ are within $[a_k^\bullet, b_k^\bullet]$.

The dynamic determination of the interval allows performing the Chebyshev series expansion on the support of the Gaussian estimate only, which reduces to number of series components compared to the aforementioned fixed interval selection. On the other hand, the series expansion has to be performed for any prediction and measurement update. For the simulations in this paper, the dynamic interval calculation is utilized, where l is set to four.

5.2 Prediction

Predicting the estimate $\boldsymbol{x}_k^e \sim \mathcal{N}\left(x_k; \mu_k^e, (\sigma_k^e)^2\right)$ of the previous measurement update from time step k to $k+1$ requires the calculation of the predicted mean and variance according to

$$\begin{aligned} \mu_{k+1}^p &= \mathrm{E}\{a_k(\boldsymbol{x}_k, u_k)\} , \\ \left(\sigma_{k+1}^p\right)^2 &= \mathrm{E}\left\{\left(a_k(\boldsymbol{x}_k, u_k) - \mu_{k+1}^p\right)^2\right\} + \left(\sigma_k^w\right)^2 , \end{aligned}$$

where $\left(\sigma_k^w\right)^2$ is the variance of the system noise $\boldsymbol{w}_k$. Assuming that the system function $a_k(.,u_k)$ for a given system input u_k is represented by means of a Chebyshev series with n components, the mean and variance can be calculated via

$$\mu_{k+1}^p = \left(\underline{c}_{k,n}^a\right)^{\mathrm{T}} \cdot \mathbf{A}_n \cdot \underline{\mathrm{E}}_{k,n}^e , \tag{22}$$

$$\left(\sigma_{k+1}^p\right)^2 = \left(\underline{c}_{k,n}^a \otimes \underline{c}_{k,n}^a\right)^{\mathrm{T}} \cdot \mathbf{P}_{2n} \cdot \underline{\mathrm{E}}_{k,2n}^e - \left(\mu_{k+1}^p\right)^2 + \left(\sigma_k^w\right)^2 , \tag{23}$$

respectively. Here, (22) coincides with (18), where $\underline{\mathrm{E}}_{k,n}^e$ is the vector of non-central moments of the current estimate $\boldsymbol{x}_k^e$ transformed to the interval Ω (see (16)) and $\underline{c}_{k,n}^a$ is the vector of series coefficients resulting from approximating $a_k(.,u_k)$ at time step k. In contrast, the matrix of Chebyshev coefficients $\mathbf{A}_n$ remains constant for all time steps unless the number of components n is varied over time. Similarly, (23) follows directly from (20) and (21), with matrix $\mathbf{P}_{2n}$ being constant at all times.

5.3 Measurement Update

For the measurement updates, the goal is to update the prediction $\boldsymbol{x}_k^p$ given the latest measurement value $\hat{z}_k$. Therefor, it is assumed that $\boldsymbol{x}_k^p$ and $\boldsymbol{z}_k$ are jointly Gaussian[2], which requires to compute the joint mean vector and joint covariance matrix

$$\underline{\mu}_k^{xz} = \begin{bmatrix} \mu_k^p \\ \mu_k^z \end{bmatrix} , \quad \mathbf{C}_k^{xz} = \begin{bmatrix} \left(\sigma_k^p\right)^2 & \sigma_k^{xz} \\ \sigma_k^{xz} & \left(\sigma_k^z\right)^2 \end{bmatrix} ,$$

2 This assumption is common in Kalman filtering (as in the EKF or the UKF) and is only true for linear systems affected with Gaussian noise. Otherwise it is an approximation.

respectively. Given both, the updated mean and variance are then calculated according to

$$\begin{aligned}\mu_k^e &= \mu_k^p + K_k \cdot \left(\hat{z}_k - \mu_k^z\right) \ , \\ \left(\sigma_k^e\right)^2 &= \left(\sigma_k^p\right)^2 - K_k \cdot \sigma_k^{xz} \ ,\end{aligned}$$

which coincides with the well-known Kalman filter update step, where $K_k \triangleq \sigma_k^{xz} / \left(\sigma_k^z\right)^2$ is the Kalman gain. The update step requires determining the mean μ_k^z and variance $\left(\sigma_k^z\right)^2$ of the measurement $\boldsymbol{z}_k$ as well as the calculation of the covariance σ_k^{xz} of state and measurement.

Measurement Mean and Variance

The calculation of μ_k^z and $\left(\sigma_k^z\right)^2$ requires a prediction based on the measurement model (2). Analogously to the prediction of the system state described in Section 5.2, the desired moments result to

$$\begin{aligned}\mu_k^z &= \left(\underline{c}_{k,n}^h\right)^{\mathrm{T}} \cdot \mathbf{A}_n \cdot \underline{\mathrm{E}}_{k,n}^p \ , \\ \left(\sigma_k^z\right)^2 &= \left(\underline{c}_{k,n}^h \otimes \underline{c}_{k,n}^h\right)^{\mathrm{T}} \cdot \mathbf{P}_{2n} \cdot \underline{\mathrm{E}}_{k,2n}^p - \left(\mu_k^p\right)^2 + \left(\sigma_k^v\right)^2 \ ,\end{aligned} \tag{24}$$

where $\mathrm{E}_{k,n}^p$ comprises all non-central moments up to order n of the prediction $\boldsymbol{x}_k^p$ transformed to interval Ω and $\underline{c}_{k,n}^h$ is the vector of series coefficients of the Chebyshev series expansion of $h_k(.)$. Both matrices $\mathbf{A}_n$ and $\mathbf{P}_{2n}$ are independent of the measurement function $h_k(.)$ and thus, can be used in both the prediction and the measurement update. It is worth mentioning that the degree n of the series expansion needs not to be identical for the prediction and measurement update.

Covariance of State and Measurement

The remaining unknown parameter is the covariance σ_k^{xz} of state and measurement. Similar to (20), the covariance can be formulated to

$$\begin{aligned}\sigma_k^{xz} &= \mathrm{E}\{(\boldsymbol{x}_k - \mu_k^p)(\boldsymbol{z}_k - \mu_k^z)\} \\ &= \mathrm{E}\{\boldsymbol{x}_k \cdot h_k(\boldsymbol{x}_k)\} - \mu_k^p \cdot \mu_k^z \ .\end{aligned}$$

Here, the first term cannot be evaluated in closed form for arbitrary measurement functions, but given the Chebyshev approximation of $h_k(.)$, evaluating the expectation boils down to calculating the expectation of a product of a linear function (x_k) with a Chebyshev series. This again can be easily calculated in matrix-vector fashion according to

$$\sigma_k^{xz} = \left(\underline{c}_{k,n}^h\right)^{\mathrm{T}} \cdot \mathbf{A}_{k,n}^* \cdot \underline{\mathrm{E}}_{k,n}^p - \mu_k^p \cdot \mu_k^z \,,$$

where the $(n+1) \times (n+2)$ matrix $\mathbf{A}_{k,n}^*$ is given by

$$\mathbf{A}_{k,n}^* \triangleq \tfrac{1}{2}\left(\left[\underline{0} \quad (b_k^p - a_k^p) \cdot \mathbf{A}_n\right] + \left[(a_k^p + b_k^p) \cdot \mathbf{A}_n \quad \underline{0}\right]\right) . \tag{25}$$

This matrix comprises the Chebyshev coefficients resulting from the multiplication of the Chebyshev series with x_k and simultaneously transforms the result of this multiplication back to the interval $[a_k^p, b_k^p]$ by means of the inverse variable transform (15). The multiplication increases the degree of any involved Chebyshev polynomial by one, which coincides with shifting the coefficients one column to the right. This can be easily seen by setting $a_k^p = -1$ and $b_k^p = 1$, where (25) becomes $\mathbf{A}_{k,n}^* = \left[\underline{0} \; \mathbf{A}_n\right]$.

6 Results

Two numerical simulations (Sections 6.1 and 6.3) as well as a real-world application example (Section 6.2) are conducted for evaluating the performance of the proposed estimator.

6.1 Example I: Higher-Order Moments

In the first simulation example, the transformation

$$\boldsymbol{y} = |\boldsymbol{x}|$$

is considered, where $\boldsymbol{x}$ has mean $\mu_x = 0.5$ and variance $\sigma_x^2 = 1$. The resulting random variable $\boldsymbol{y}$ is non-Gaussian and the odd moments are non-zero [29]. Furthermore, the transformation is non-smooth and thus, not ideally suited of a polynomial series expansion.

Monte Carlo (MC) integration with 10 million samples is performed to determine the true (non-central) moments of $\boldsymbol{y}$. The results are shown in Table 1

Table 1: Calculation of the moments of the random variable $\boldsymbol{y} = |\boldsymbol{x}|$.

	Moments				
	1	2	3	4	5
MC	0.896	1.25	2.206	4.562	10.619
UKF	0.983	1.25	1.879	3.063	5.165
CPKF, $n = 2$	0.747	1.25	3.795	13.563	50.348
CPKF, $n = 5$	0.997	1.25	1.987	4.563	15.775
CPKF, $n = 10$	0.932	1.25	2.198	4.563	10.630
CPKF, $n = 15$	0.882	1.25	2.209	4.563	10.623

together with the estimates of the unscented Kalman filter (UKF[3], [16, 31]) and the proposed estimator (denoted as CPKF). For the CPKF, the number of series components n is varied. For $n = 2$, the number of series components is three and thus, identical with the number of sigma-points of the UKF. In this case, the UKF is closer to the true moments of $\boldsymbol{y}$ than the CPKF. But with an increasing number of components n, CPKF approximates the moments of $\boldsymbol{y}$ with increasing accuracy. Further, the CPKF is now also able to accurately capture moments of order three and higher, while the UKF cannot provide accurate estimates of these moments. The main reason for this can be found in the way moments are calculated by the UKF. While the CPKF approximates the transformation and then propagates the entire density of $\boldsymbol{x}$, the UKF captures the density of $\boldsymbol{x}$ merely by a small set of sigma-points. This set cannot capture and propagate higher-order information with sufficient accuracy.

6.2 Example II: Real-World Application

In this example, real-world data from monitoring the advertising effectiveness of a TV commercial campaign for a single product is considered [25, 32]. This data is obtained by means of weekly surveys, where a given number of individuals from the population of TV viewers in UK is sampled in order to count the number being aware of current or recent TV commercials for the product. The result of each survey is measured in standardized units known as television ratings (TVRs).

3 The Matlab implementation of the UKF available at `http://www.cs.ubc.ca/~nando/software.html` is used. The parameters of the UKF are set to $\alpha = 1$, $\beta = 0$ and $\kappa = 0.5$.

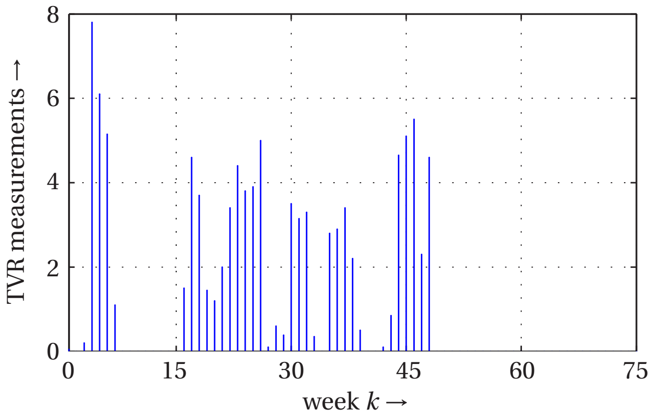


Figure 2: Weekly TVR measurements forming the input u_k.

Let u_k denote the TVR measurement for week k. In Figure 2, TVR measurements for 75 weeks are depicted.

The TVR measurements drive the nonlinear five-dimensional dynamics equation

$$\underline{\boldsymbol{x}}_{k+1} = \underline{a}(\underline{\boldsymbol{x}}_k + \underline{\boldsymbol{w}}_k, u_k) \quad , \quad \underline{\boldsymbol{w}}_k \sim \mathcal{N}\left(\underline{w}_k; \underline{0}, 0.03 \cdot \mathbf{C}_k^x\right) ,$$

with $\mathbf{C}_k^x$ being the covariance matrix of $\underline{\boldsymbol{x}}_k$ and system function

$$\underline{a}(\underline{\boldsymbol{x}}, u) = \left[\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \boldsymbol{x}_3 \quad \boldsymbol{x}_4 \quad (\boldsymbol{x}_2 - \boldsymbol{x}_1) - (\boldsymbol{x}_2 - \boldsymbol{x}_1 - \boldsymbol{x}_3 \cdot \boldsymbol{x}_5) \cdot \exp(-\boldsymbol{x}_4 \cdot u)\right]^{\mathrm{T}} . \quad (26)$$

The state vector $\underline{x}$ comprises the minimum level of awareness $\boldsymbol{x}_1$, maximum level of awareness $\boldsymbol{x}_2$, memory decay rate $\boldsymbol{x}_3$, penetration $\boldsymbol{x}_4$, and effect of TVR on the awareness $\boldsymbol{x}_5$ (for details see [32]). The measurement equation is given by

$$\boldsymbol{z}_k = \boldsymbol{x}_{1,k} + \boldsymbol{x}_{5,k} + \boldsymbol{v}_k = \mathbf{H} \cdot \underline{\boldsymbol{x}}_k + \boldsymbol{v}_k \quad (27)$$

with $\mathbf{H} \triangleq [1, 0, 0, 0, 1]$, where $\boldsymbol{z}_k$ corresponds to the awareness proportion and the noise $\boldsymbol{v}_k$ has standard deviation $\sigma_k^v = 0.05$. The initial state estimate is given by

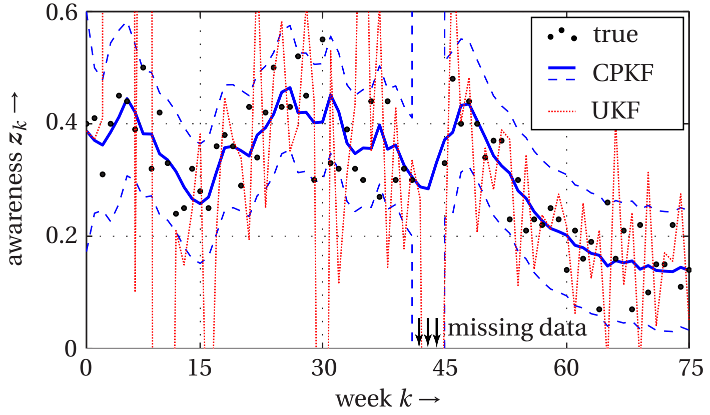


Figure 3: Predicted awareness proportions of the CPKF (blue solid line) with 95% confidence region (blue dashed) as well as the predictions of the UKF (red dotted). The true awareness proportion values are indicated by the black dots. For the weeks $k = 42, 43, 44$ no awareness measurements are available.

$\underline{x}_0 \sim \mathcal{N}\left(\underline{x}_0; \underline{\mu}_0^x; \mathbf{C}_0^x\right)$ with mean vector and covariance matrix

$$\underline{\mu}_0^x = \begin{bmatrix} 0.10 \\ 0.85 \\ 0.90 \\ 0.02 \\ 0.30 \end{bmatrix} \quad \text{and} \quad \mathbf{C}_0^x = \begin{bmatrix} 6.25 & 6.25 & 0 & 0 & 0 \\ 6.25 & 406.25 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2.25 & 0 \\ 0 & 0 & 0 & 0 & 100 \end{bmatrix} ,$$

respectively.

As the system state has dimension five, the proposed CPKF is merely applied on the one-dimensional nonlinear mapping $\exp(-\boldsymbol{x}_4 \cdot u)$ in (26). The reaming four states are processed by means of the UKF. A more detailed derivation of this state decomposition can be found in Appendix A. The CPKF is compared with the UKF, where the unscented transform is applied to all five state dimensions. For both estimators, the measurement update step is performed via a Kalman filter, as the measurement equation (27) is linear. The true awareness proportion values for performing the update step are taken from [32] and are depicted in Figure 3. It is important to note that for the weeks 42, 43, and 44 no awareness measurements are available.

In the following the predictions of the awareness proportion $\boldsymbol{z}_k$ of the CPKF and UKF are compared before updating the state estimates with the true awareness value $\hat{z}_k$. These predictions are calculated as a by-product of the measurement update step (see (24)). In Figure 3, the predicted awareness proportions are depicted. It is obvious that the UKF behaves very unsteady and is heavily fluctuating. Thus, the resulting awareness predictions are very inaccurate. This effect can be explained by overly confident estimates, i.e., the covariance matrix of the system state contains too small variances.

The behavior of the CPKF is different, which is surprising as the CPKF is merely applied on $\exp(-\boldsymbol{x}_4 \cdot u)$, while the remaining parts of the system equation are processed via the UKF. Thus, the CPKF has a stabilizing effect on the UKF resulting in awareness predictions that accurately follow the ground truth. Furthermore, the CPKF is not overconfident as the predicted measurement variances $(\sigma_k^z)^2$ are sufficiently large to capture the true awareness proportions. Even for the weeks with missing data, the predictions of CPKF are meaningful as the variances grow and thus, indicate an increasing uncertainty. Though, the trend is still correct.

6.3 Example III: Time Series

In the following numerical simulation example, the non-stationary growth model with nonlinear dynamics equation

$$\boldsymbol{x}_{k+1} = \frac{\boldsymbol{x}_k}{2} + 25 \cdot \frac{\boldsymbol{x}_k}{1+\boldsymbol{x}_k^2} + 8 \cdot \cos(1.2 \cdot k) + \boldsymbol{w}_k$$

and nonlinear measurement equation

$$\boldsymbol{z}_k = \frac{\boldsymbol{x}_k^2}{20} + \boldsymbol{v}_k$$

is considered. Due to its highly nonlinear and bimodal nature, this example has often been utilized before for comparing the performance of state estimators [10, 19, 20]. The bimodality follows from the quadratic measurement equation, which possesses a bimodal likelihood in case of positive measurement values, i.e., for $z_k > 0$.

For simulation purposes, the parameters for the noise terms and initial system state are chosen as follows. The system noise $\boldsymbol{w}_k$ has variance $\left(\sigma_k^w\right)^2 = 10$ and the variance of the measurement noise $\boldsymbol{v}_k$ is $\left(\sigma_k^v\right)^2 = 1$. The initial state estimate at time step $k = 0$ is $\boldsymbol{x}_0 \sim \mathcal{N}(x_0; 0, 10)$, the true system state x_0 is sampled from $\boldsymbol{x}_0$.

Besides the proposed CPKF, the following estimators are considered for comparison:

UKF	Unscented Kalman filter with two and three sigma-points.
GHKF	Gauss-Hermite Kalman filter [4, 14] allows an arbitrary number of sigma-points.
CKF	Cubature Kalman filter [3] operates with two sigma-points.
EKF	Extended Kalman filter is based on first-order Taylor series expansion.
PF	Particle filter with residual resampling [8].

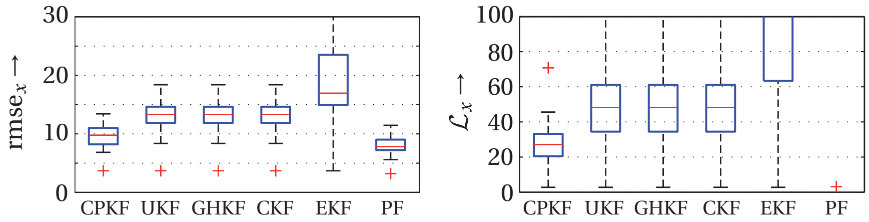
The CPKF is applied with two ($n = 1$), three ($n = 2$), and ten ($n = 9$) series components. In doing so, for low n the performance of the CPKF can be compared with the UKF and CKF with a corresponding number of sigma-points. Both numbers of sigma-points are also employed for the GHKF[4], but in addition a GHKF with ten sigma-points is used as well. The PF is the only non-Gaussian state estimator and thus, ideally suited for the nonlinearities considered in this simulation example. The number of particles is chosen in such a way that the runtime of the PF is close to the runtime of the CPKF. On this account, PFs with 10, 20, and 100 particles are applied. Their runtimes correspond to CPKFs with two, three, and ten series components, respectively.

The simulation is performed for 100 Monte Carlo runs, where each run comprises 50 time steps with alternating predictions and measurement updates. The performance of all estimators is evaluated by means of two measures, the root mean square error (rmse) and the negative log-likelihood of the state estimate (see for instance [21])

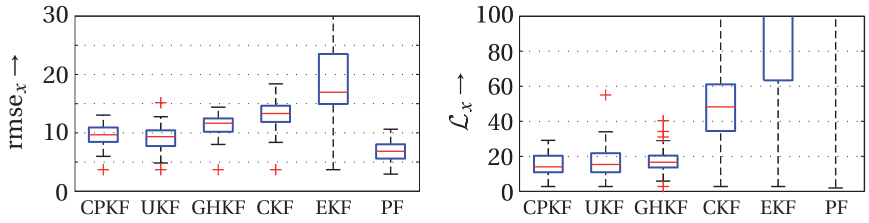
$$\mathcal{L}_x \triangleq \log\left(\sqrt{2\pi}\sigma_k^e\right) + \tfrac{1}{2}\left(\tfrac{x_k - \mu_k^e}{\sigma_k^e}\right)^2 .$$

For both, lower values indicate a better performance. While the rmse merely penalizes the deviation of the estimated state $\boldsymbol{x}_k^e$ from the true state x_k, the log-likelihood penalizes both inconsistency and uncertainty. The log-likelihood has high values in case of a strong error between state and estimate as well as in case of an overestimation (variance $\left(\sigma_k^e\right)^2$ is too high) or underestimation (variance $\left(\sigma_k^e\right)^2$ is too low) of the error.

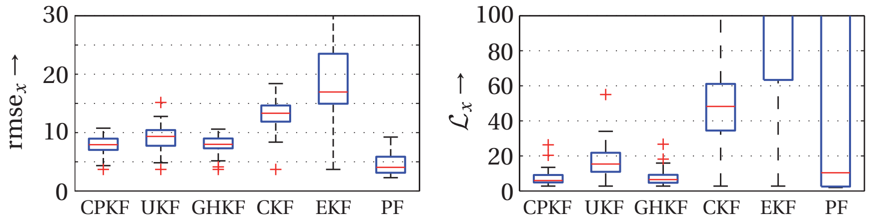
4 For both the GHKF and CKF the Matlab implementation available at `http://becs.aalto.fi/en/research/bayes/ekfukf/` is used.



(a) Two series components/sigma-points for CPKF, UKF, GHKF, CKF as well as ten particles for PF.



(b) Three series components/sigma-points for CPKF, UKF, and GHKF as well as 20 particles for PF. CKF is identical with (a) and (c), as it only operates with two sigma-points.



(c) Ten series components/sigma-points for CPKF and GHKF as well as 100 particles for PF. UKF is identical with (b), as the maximum number of sigma-points is three.

Figure 4: Median (red line), lower and upper quantiles (blue box), and spread (black lines) of rmse (left row) and negative log-likelihood (right row) of all estimators. Red crosses indicate outliers. The results of the EKF are identical for (a), (b), and (c).

In Figure 4, the rmse and negative log-likelihood over all simulation runs for different numbers of series components/sigma-points are depicted. It can be seen that the CPKF—together with the GHKF—provides the lowest log-likelihood

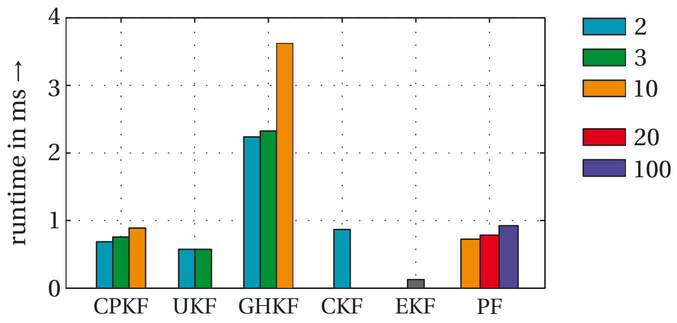


Figure 5: Average runtime of all estimators for different number of series coefficients, sigma-points, and particles, respectively.

values, i.e., the estimates of the CPKF are the most consistent ones without being overly uncertain. In terms of the rmse, CPKF outperforms all other Gaussian estimators clearly for two series components/sigma-points (see Figure 4b). Here, UKF, GHKF and CKF provide almost identical results, as the sigma-points of CKF and GHKF are identical and very close to the sigma-points of the UKF. The approximation approach of the CPKF is completely different to these Gaussian estimators, as it relies on a function approximation instead of a point-based approximation of the state distribution. Thus, CPKF provides different estimates. The EKF often loses track of the true state without being aware of it—the EKF on average has the lowest rmse and at the same time the lowest variances of all Gaussian estimators and thus, is too confident.

For a higher number of components, the rmse of the CPKF is comparable to UKF and GHKF, but in terms of the log-likelihood merely the GHKF is comparable to CPKF. There is a clear gain of estimation performance when the number of series components is increased, but since CPKF is still a Gaussian estimator, this performance improvement is bounded and spending too many components will have no visible impact. Instead, merely the computational demand will increase. This holds for any (approximate) Gaussian assumed density filter.

If the runtime (see Figure 5) is also taken into account as a performance criterion, it becomes obvious that the CPKF has a good estimation accuracy/runtime ratio. The runtime is close to the runtime of the fastest sigma-point approaches. This low runtime is due to the sparse and computationally efficient matrix-vector notation of all CPKF estimation steps. The GHKF for instance, which is very close to the CPKF in terms of estimation performance, is by far the slowest Gaussian estimator. On the other hand, the EKF is the fastest estimator as the linearization is performed off-line, but its estimates are inaccurate.

As mentioned above, the number of particles of the PF is chosen such that the runtime is similar to the runtime of the CPKF. As the PF is not a Gaussian estimator, it can provide estimates that are much closer to the true state compared to the Gaussian estimators. However, the PF underestimates its error and thus, the log-likelihood values are the worst of all estimators as the variance of the particles is by far too low. Only by spending many particles—significantly more than 100—the PF estimates become consistent. But this in turn comes at the expense of a higher computational load.

7 Discussion

In the following strengths and limitations of the proposed approach as well as potential extensions are discussed.

7.1 Strengths

The currently most prominent state estimation techniques rely on sampling. This is done for instance by means of deterministic sigma-point calculation as in the UKF or by means of random sampling as in particle filters. Given the sample representation, prediction and the calculation of statistics is fairly simple, as the samples merely have to be propagated through the nonlinearity. The main drawback of such techniques is that only a partial representation of the state distribution is considered. This is different to the proposed CPKF. Here, the entire distribution is propagated through the nonlinearity. The approximation is applied on the given nonlinear function instead. Thanks to the Chebyshev polynomial series, this approximation is typically very accurate.

The EKF and its derivatives form a class of estimators that also propagates the entire distribution. However, the approximation of nonlinearity is much coarser compared to the CPKF as merely a linear approximation in case of the EKF is employed. Furthermore, the EKF relies on a priori calculation of derivatives of the nonlinearity, which requires manual operations. The CPKF instead operates fully automated, i.e., all approximations are performed on-line without any user interaction.

In [22], a Gaussian estimator for polynomial nonlinearities has been proposed. Here, all moment propagations rely on Taylor-series expansions. In doing so, the resulting prediction and measurement update steps involve non-sparse matrices with elements requiring complex computations. The computational complexity of the CPKF is low as merely matrix-vector operations with sparse matrices are involved. For instance, the complexity of the mean propagation (18) and variance propagation (21) are in $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$, respectively. But this is just the worst-case consideration. As the involved matrices $\mathbf{A}_n$ and $\mathbf{P}_{2n}$ are sparse, an even lower computational complexity can be achieved.

Utilizing Chebyshev polynomials for approximating nonlinear systems in the context of Gaussian filtering has firstly been proposed in [33]. The authors demonstrated the superior prediction performance compared to the EKF and UKF. In contrast to [33], in the proposed approach the given nonlinear system can be defined over arbitrary intervals. While [33] merely supports the prediction step, the proposed approach additionally performs measurement updates. Finally, [33] leverages the moment propagations from [22], which are exact but computationally more demanding than the proposed ones (see above).

7.2 Limitations and Potential Extensions

All the aforementioned Gaussian estimators, including the CPKF, rely on an additional approximation for the measurement update step, which is the joint Gaussian assumption of the measurement and the state. This assumption is only valid for a linear mapping and a good approximation for nonlinear functions as long as the measurement density is unimodal and close to a Gaussian density. In case of multi-modal or heavily skewed densities, the joint Gaussian assumption is inappropriate and thus, a Gaussian estimator is expected to provide inaccurate estimation results or even may completely diverge. One potential solution of this limitation is the extension towards Gaussian mixture estimators [2, 11]. Here, instead of a single Gaussian, a sum of Gaussians is propagated. As Gaussian mixtures are universal function approximators [24], convergence towards the exact estimate is possible [1], but comes at the expense of an increased computational burden.

By increasing the degree of the Chebyshev series, the approximation quality of the true nonlinear function increases as well. Unfortunately, due to numerical instabilities resulting from the calculation of the Chebyshev coefficients and the series coefficients, series of a high degree—typically $n > 20$—can result in a worse estimation performance compared to a series with a much lower degree. To avoid this issue, a piece-wise approximation of the nonlinear function with a set of Chebyshev series of low degree should be applied instead.

So far, the CPKF is merely applicable to one-dimensional states. The extension towards the multi-dimensional case requires a multi-variate Chebyshev polynomial series expansion and multi-variate moment calculation for polynomials. The first step is straightforward, as the Chebyshev series expansion can be applied dimension-wise. The resulting polynomial approximation consists of the product of one-dimensional Chebyshev series. The multi-variate moment calculation, however, is still a not fully solved issue. There exists a promising solution proposed in [18], which still requires a significant amount of computations. An alternative approach is the extension of the closed-form moment calculation proposed in Section 4.1 to the multi-variate case.

8 Conclusions

The proposed Chebyshev Polynomial Kalman Filter allows analytical predictions and measurement updates in a black-box fashion. This is achieved firstly by expanding nonlinear functions in Chebyshev polynomial series, where no derivatives are required and the computationally efficient discrete cosine transform can be employed. And secondly by closed-form moment calculation and propagation, where all operations can be formulated in computationally cheap and sparse vector-matrix expressions. This allows for a runtime that is in the same order of magnitude as the runtime of Gaussian estimators relying on sigma-points like the unscented Kalman filter or the cubature Kalman filter. But in contrast to these estimators, the proposed approach exploits the entire distribution for moment propagation, which yields more accurate estimation results.

A State Decomposition

The real-world example exploited in Section 6.2 utilizes a five-dimensional system state $\underline{\boldsymbol{x}}_k$. As the CPKF is so far just applicable to one-dimensional states, state decomposition techniques developed in [6] have to be employed. Here, the state is decomposed into the four-dimensional sub-state $\underline{\boldsymbol{x}}^a \triangleq [x_1, x_2, x_3, x_5]^{\mathrm{T}} \sim$

$\mathcal{N}\left(\underline{x}^a;\underline{\mu}^a,\mathbf{C}_k^a\right)$ and the one-dimensional sub-state $\boldsymbol{x}^b \triangleq x_4 \sim \mathcal{N}\left(x^b;\mu^b,(\sigma^b)^2\right)$. In doing so, the calculation of the predicted mean is given by

$$\begin{aligned}\underline{\mu}_{k+1}^p &= \mathrm{E}\{\underline{a}(\underline{\boldsymbol{x}},u)\} \\ &= \int \underline{a}(\underline{x}_k,u_k)\cdot\mathcal{N}\left(\underline{x}_k;\underline{\mu}_k^x,\mathbf{C}_k^x\right)\mathrm{d}\underline{x}_k \\ &= \int \underline{a}(\underline{x}_k^a,x_k^b,u_k)\cdot\mathcal{N}\left(x_k^b;\mu_k^{b|a},(\sigma_k^{b|a})^2\right)\cdot\mathcal{N}\left(\underline{x}_k^a;\underline{\mu}_k^a,\mathbf{C}_k^a\right)\mathrm{d}\underline{x}_k\,, \qquad (28)\end{aligned}$$

with $\underline{a}(.,.)$ according to (26) as well as conditional mean and variance (see for instance Chapter 2.6 in [15])

$$\begin{aligned}\mu_k^{b|a} &= \mu_k^b + \mathbf{G}_k\cdot\left(\underline{x}_k^a-\underline{\mu}_k^a\right)\,, \\ (\sigma_k^{b|a})^2 &= \left(\sigma_k^b\right)^2 - \mathbf{G}_k\mathbf{C}_k^{ba}\,,\end{aligned}$$

respectively, where $\mathbf{G} \triangleq \mathbf{C}_k^{ab}\left(\mathbf{C}_k^a\right)^{-1}$ and $\mathbf{C}_k^{ab} = \left(\mathbf{C}_k^{ba}\right)^{\mathrm{T}}$ is the cross-covariance matrix between $\underline{\boldsymbol{x}}_k^a$ and $\boldsymbol{x}_k^b$. The sub-state $\underline{\boldsymbol{x}}_k^a$ is processed via the UKF, i.e., a set of s sigma-points $\mathcal{X}_i$, $i=1,\ldots,s$, with weights ω_i is drawn from the Gaussian $\mathcal{N}\left(\underline{x}_k^a;\underline{\mu}_k^a,\mathbf{C}_k^a\right)$. By substituting the Gaussian density with the sigma-points, (28) becomes

$$\begin{aligned}\underline{\mu}_{k+1}^p &\approx \int \underline{a}\left(\underline{x}_k^a,x_k^b,u_k\right)\cdot\mathcal{N}\left(x_k^b;\mu_k^{b|a},(\sigma_k^{b|a})^2\right)\cdot\left(\sum_{i=1}^s\omega_i\cdot\mathcal{X}_i\right)\mathrm{d}\underline{x}_k \\ &= \sum_{i=1}^s\omega_i\cdot\int\underline{a}\left(\mathcal{X}_i,x_k^b,u_k\right)\cdot\mathcal{N}\left(x_k^b;\mu_k^{b|a},(\sigma_k^{b|a})^2\right)\mathrm{d}x_k^b\,,\end{aligned}$$

where the remaining one-dimensional part can now be processed by means of the CPKF. The calculation of the predicted covariance matrix $\mathbf{C}_{k+1}^p$ is done analogously.

References

[1] Simo Ali-Löytty. *Gaussian Mixture Filters in Hybrid Positioning*. PhD thesis, Tampere University of Technology, Tampere, Finland, August 2009.

[2] Daniel L. Alspach and Harold W. Sorenson. Nonlinear Bayesian Estimation using Gaussian Sum Approximation. *IEEE Transactions on Automatic Control*, 17(4):439–448, August 1972.

[3] Ienkaran Arasaratnam and Simon Haykin. Cubature Kalman Filters. *IEEE Transactions on Aut*, 54(6):1254–1269, June 2009.

[4] Ienkaran Arasaratnam, Simon Haykin, and Robert J. Elliott. Discrete-Time Nonlinear Filtering Algorithms Using Gauss-Hermite Quadrature. *Proceedings of the IEEE*, 95(5):953–977, 2007.

[5] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[6] Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck. Gaussian Filtering using State Decomposition Methods. In *Proceedings of the 12th International Conference on Information Fusion (Fusion)*, pages 579–586, Seattle, Washington, July 2009.

[7] Vladimir Britanak, Patrick C. Yip, and Kamisetty R. Rao. *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*. Academic Press, 2006.

[8] James Carpenter, Peter Clifford, and Paul Fearnhead. Improved particle filter for nonlinear problems. In *IEE Proceedings Radar, Sonar and Navigation*, volume 146, pages 2–7, February 1999.

[9] Kerim Demirbaş. A novel real-time adaptive suboptimal recursive state estimation scheme for nonlinear discrete dynamic systems with non-Gaussian noise. *Digital Signal Processing*, 22(4):593–604, July 2012.

[10] Simon J. Godsill, Arnaud Doucet, and Mike West. Monte Carlo Smoothing for Nonlinear Time Series. *Journal of the American Statistical Association*, 99(465):156–168, March 2004.

[11] Marco F. Huber. Adaptive Gaussian Mixture Filter Based on Statistical Linearization. In *Proceedings of the 14th International Conference on Information Fusion (Fusion)*, Chicago, Illinois, July 2011.

[12] Marco F. Huber, Frederik Beutler, and Uwe D. Hanebeck. Semi-Analytic Gaussian Assumed Density Filter. In *Proceedings of the 2011 American Control Conference (ACC)*, San Francisco, California, June 2011.

[13] Marco F. Huber and Uwe D. Hanebeck. Gaussian Filter based on Deterministic Sampling for High Quality Nonlinear Estimation. In *Proceedings of the 17th IFAC World Congress*, Seoul, Republic of Korea, July 2008.

[14] Kazufumi Ito and Kaiqi Xiong. Gaussian Filters for Nonlinear Filtering Problems. *IEEE Transactions on Automatic Control*, 45(5):910–927, May 2000.

[15] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*. Dover Publications, Inc., 2007.

[16] Simon J. Julier and Jeffrey K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[17] Rudolf E. Kalman. A new Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME, Journal of Basic Engineering*, 82 (Series D)(1):35–45, 1960.

[18] Raymond Kan. From Moments of Sum to Moments of Product. *Journal of Multivariate Analysis*, 99(3):542–554, March 2008.

[19] Genshiro Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[20] Jayesh H. Kotecha and Petar M. Djurić. Gaussian Particle Filtering. *IEEE Transactions on Signal Processing*, 51(10):2592–2601, 2003.

[21] Miguel Lázaro-Gredilla, Joaquin Quiñonero-Candela, Carl Edward Rasmussen, and Aníbal R. Figueiras-Vidal. Sparse Specturm Gaussian Process Regression. *Journal of Machine Learning Research*, 11:1865–1881, June 2010.

[22] Mihai Bogdan Luca, Stéphane Azou, Gilles Burel, and Alexandru Serbanescu. On Exact Kalman Filtering of Polynomial Systems. *IEEE Transactions on Circuits and Systems—I: Regular Papers*, 53(6):1329–1340, June 2006.

[23] John C. Mason and David C. Handscomb. *Chebyshev Polynomials*. Chapman & Hall/CRC, 2003.

[24] Vladimir Maz'ya and Gunther Schmidt. On approximate approximations using gaussian kernels. *IMA J. Numer. Anal.*, 16:13–29, 1996.

[25] H. S. Migon and P. J. Harrison. An application of non-linear Bayesian forecasting to television advertising. In J. M. Bernardo, M. H DeGroot, D. V. Lindley, and A. F. M. Smith, editors, *Bayesian Statistics 2*. Valencia University Press, 1985.

[26] Kevin Murphy and Stuart Russel. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[27] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*, chapter 6: Evaluation of Functions. Cambridge University Press, 3rd edition, 2007.

[28] Dan Simon. *Optimal State Estimation: Kalman, H-Infinity, and Nonlinear Approaches*. John Wiley & Sons, Inc., 1st edition, 2006.

[29] Dirk Tenne and Tarunray Singh. The Higher Order Unscented Filter. In *Proceedings of the American Control Conference*, pages 2441–2446, June 2003.

[30] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[31] Eric A. Wan and Rudolph van der Merwe. The Unscented Kalman Filter. In Simon Haykin, editor, *Kalman Filtering and Neural Networks*, chapter The Unscented Kalman Filter, pages 221–280. John Wiley & Sons, Inc., 2001.

[32] Mike West and Jeff Harrison. *Bayesian Forecasting and Dynamic Models*, chapter 14: Exponential Family Dynamic Models, pages 534–555. Springer, 1997.

[33] Moussa Yahia, Pascal Acco, and Malek Benslama. Estimation of Nonlinear Systems via a Chebyshev Approximation Approach. *International Journal of Control, Automation, and Systems*, 9(6):1021–1027, 2011.

Paper D

Gaussian Filtering for Polynomial Systems Based on Moment Homotopy

Authors: Marco F. Huber and Uwe D. Hanebeck

Edited version of the paper:

M. F. Huber and U. D. Hanebeck. Gaussian Filtering for Polynomial Systems Based on Moment Homotopy. In *Proceedings to the 16th International Conference on Information Fusion (Fusion)*, pages 1080–1087, Istanbul, Turkey, July 2013.

Gaussian Filtering for Polynomial Systems Based on Moment Homotopy

Marco F. Huber* and Uwe D. Hanebeck**

*AGT International
Darmstadt, Germany
marco.huber@ieee.org

** Intelligent Sensor-Actuator-Systems
Laboratory (ISAS)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{beutler|uwe.hanebeck}@ieee.org

Abstract

This paper proposes Gaussian filters for polynomial systems with efficient solutions for both the prediction and the filter step. For the prediction step, computationally efficient closed-form solutions are derived for calculating the exact moments. In order to achieve a higher estimation quality, the filter step is solved without the usual additional assumption that state and measurement are jointly Gaussian distributed. As this significantly complicates the required moment calculation, a homotopy continuation method is employed that yields almost optimal results.

1 Introduction

Closed-form recursive Bayesian state estimation can only be performed for a few special systems, such as linear continuous systems or systems with finite state and measurement spaces. The famous Kalman filter is the best linear estimator, being optimal for the linear Gaussian case. For finite state and measurement spaces, grid-based filters are optimal [2]. For arbitrary nonlinearities, however, that are typical in real-world applications such as target tracking, financial forecasting, medical surveillance, or robotics, recursive Bayesian state estimation requires approximate solutions.

A practical approximation known as Gaussian assumed density filtering restricts the state estimate to be Gaussian distributed [12]. Preserving the mean and variance of the state via moment matching requires the solution of expectation integrals. Since closed-form expectation calculation is not possible in general and numerical integration is computationally infeasible, many fast approximate Gaussian assumed density filters utilizing the Kalman filter equations have been developed in the past. Employing first-order Taylor-series expansion to linearize the given system and measurement models leads to the extended Kalman filter [7]. This fast filter is only applicable to mild nonlinearities and requires differentiability. To cover also stronger nonlinearities, so-called linear regression Kalman filters have become popular in the recent years, where the approximation relies on deterministic sampling. Part of this group of Gaussian filters are the unscented Kalman filter [13], the cubature Kalman filter [1], or the Gaussian estimator [6].

Compared to these generic Gaussian filters, improved estimation performance can be achieved by focusing on a particular type of nonlinearity. In this paper, the focus is on polynomial nonlinearities. First results on Bayesian estimation for polynomial dynamics but linear measurement models can be found in [3]. The more general case is treated in [8], where exact moment calculation for the prediction step is derived based on Taylor-series expansion.

In Section 4 of this paper, special properties of exponential densities with polynomial exponents are exploited to efficiently calculate the moments after a polynomial transformation. Compared to [8], this leads to a simplified and computationally cheaper calculation of moments after a prediction step. A straightforward application of these insights to the measurement step requires the assumption of a jointly Gaussian distributed state and measurement, which is a typical assumption in Gaussian filtering. For polynomial nonlinearities, however, the posterior density is an exponential density and thus, it is a conjugate density to the prior Gaussian density. Unfortunately, exponential densities allow no closed-form calculation of the moments in general, which is necessary for Gaussian filtering. To overcome this limitation, a homotopy continuation approach for calculating the posterior moments is proposed in Section 5. The continuation starts with the known moments of the Gaussian prior, while the likelihood, which depends on the polynomial nonlinearity, is gradually introduced into the measurement update step. This causes a continuous transformation of the prior moments towards the posterior moments. The transformation can be expressed via a system of first-order ordinary differential equations, for which a plethora of efficient numerical solvers exists.

The proposed Gaussian filters for polynomial nonlinearities are compared to the state-of-the-art by means of numerical simulations in Section 6. The paper closes with a conclusion and an outlook to future work.

2 Problem Formulation

In this paper, nonlinear discrete-time system and measurement equations

$$\boldsymbol{x}_{k+1} = a_k(\boldsymbol{x}_k) + \boldsymbol{w}_k \ , \tag{1}$$

$$\boldsymbol{z}_k = h_k(\boldsymbol{x}_k) + \boldsymbol{v}_k \ , \tag{2}$$

are considered, where $\boldsymbol{x}_k$ is the scalar system state at time step $k = 0,1,\ldots$ and $\boldsymbol{z}_k$ is the scalar measurement. An actual measurement value $\hat{z}_k$ is a realization of the random variable $\boldsymbol{z}_k$. Both $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are white zero-mean Gaussian noise processes with variance $\left(\sigma_k^w\right)^2$ and $\left(\sigma_k^v\right)^2$, respectively.

In Bayesian estimation, two alternating steps, i.e., prediction and measurement update, are performed for estimating the system state $\boldsymbol{x}_k$. The latest estimate of $\boldsymbol{x}_{k-1}$ is propagated to time step k by means of the system equation (1) in the prediction step. In the measurement update, a given measurement value $\hat{z}_k$ is exploited for updating $\boldsymbol{x}_k$ under consideration of the measurement equation (2).

Exact closed-form solutions for the prediction and the measurement update are not available for arbitrary $a_k(.)$, $h_k(.)$ and arbitrarily distributed random variables. This paper is restricted to polynomial system and measurement functions $a_k(.)$ and $h_k(.)$, respectively. It is further assumed, that the system state $\boldsymbol{x}_k$ can be represented by means of Gaussian distributions for all time steps k. Thus, it is sufficient to investigate polynomial transformations of the form

$$\boldsymbol{y} = g(\boldsymbol{x}) + \boldsymbol{w} = \sum_{i=0}^{n} c_i \cdot \boldsymbol{x}^i + \boldsymbol{w} \ , \tag{3}$$

where a Gaussian $\boldsymbol{x} \sim \mathcal{N}\left(x; \mu_x, \sigma_x^2\right)$ with mean μ_x and variance σ_x^2 is mapped to a random variable $\boldsymbol{y}$. In case of a prediction, $\boldsymbol{y}$ corresponds to $\boldsymbol{x}_{k+1}$, while for a measurement update, $\boldsymbol{y}$ is the measurement $\boldsymbol{z}_k$. The transformation is affected by zero-mean Gaussian noise $\boldsymbol{w} \sim \mathcal{N}\left(w; 0, \sigma_w^2\right)$, which is assumed to be uncorrelated with $\boldsymbol{x}$.

The goal now is two-fold: (i) calculating the mean and variance of $\boldsymbol{y}$ for prediction purposes and (ii) incorporation of a realization $\hat{y}$ of $\boldsymbol{y}$ to perform a measurement update.

3 Exponential Densities

At first, a brief introduction to the so-called family of *exponential densities with polynomial exponents* is provided. Their properties regarding recursive moment calculation play a significant role for solving the problem at hand.

3.1 Definition

An unnormalized one-dimensional exponential density is defined as

$$f(x) = \exp\left(\sum_{i=0}^{n} \eta_i \cdot x^i\right) = \exp\left(\underline{\eta}^{\mathrm{T}} \cdot \underline{x}\right) ,$$

with parameter vector $\underline{\eta}^{\mathrm{T}} \triangleq [\eta_0, \eta_1, \ldots, \eta_n]$ and $\underline{x}^{\mathrm{T}} \triangleq [1, x, x^2, \ldots, x^n]$ being the vector of monomials. To ensure that the exponential density is non-negative for all $x \in \mathbb{R}$ and has finite moments, the maximum degree $n \in \mathbb{N}$ must be even and the highest-order coefficient η_n must be negative, i.e., $\eta_n < 0$. If desired, the exponential density can be normalized by adding a term $\log\bigl(c(\underline{\eta})\bigr)$ to the first coefficient η_0, where $c(\underline{\eta})$ is a normalization constant.

An important special case of the family of exponential densities is the (unnormalized) Gaussian density

$$f(x) = \exp\left(-\tfrac{1}{2}\left(\tfrac{x-\mu_x}{\sigma_x}\right)^2\right) = \exp\left(\eta_0 + \eta_1 \cdot x + \eta_2 \cdot x^2\right) ,$$

with $\eta_0 = -\mu_x^2/(2\sigma_x^2)$, $\eta_1 = \mu_x/\sigma_x^2$, and $\eta_2 = -1/\bigl(2\sigma_x^2\bigr)$. To obtain a normalized Gaussian density, the first coefficient is modified according to $\eta_0 = -\mu_x^2/(2\sigma_x^2) + \log\bigl(c(\underline{\eta})\bigr)$ with $c(\underline{\eta}) = 1/\sqrt{2\pi}\sigma_x$, while η_1 and η_2 remain unchanged[1].

1 The term $\mathcal{N}\bigl(x; \mu_x, \sigma_x^2\bigr)$ always refers to a *normalized* Gaussian density in this paper.

3.2 Recursive Moment Calculation

In general, no analytic expressions for the (non-central) moments

$$\mathrm{E}_i \triangleq \mathrm{E}\{x^i\} = \int x^i \cdot f(x)\,\mathrm{d}x \tag{4}$$

of an exponential density for $i \in \mathbb{N}_0$ exist, not even for the zeroth-order moment E_0, which is required for determining the normalization constant $c(\underline{\eta})$. Only for some special cases like the Gaussian density, it is possible to derive analytic expressions. However, if at least the first n moments $\mathrm{E}_0, \mathrm{E}_1, \ldots, \mathrm{E}_{n-1}$ are given, *all* higher-order moments can be determined recursively. As shown in [4, 10], integrating (4) by parts with respect to x yields

$$\begin{aligned}\mathrm{E}_i &= \underbrace{\left(\frac{x^{i+1}}{i+1} f(x)\right)\Bigg|_{-\infty}^{\infty}}_{=0} - \int \frac{x^{i+1}}{i+1}\frac{\partial f(x)}{\partial x}\,\mathrm{d}x \\ &= -\int \frac{x^{i+1}}{i+1}\left(\sum_{j=1}^{n} j\cdot\eta_j\cdot x^{j-1}\right)\cdot f(x)\,\mathrm{d}x\,,\end{aligned}$$

which finally gives

$$\mathrm{E}_i = -\sum_{j=1}^{n} \frac{j}{i+1}\eta_j\,\mathrm{E}_{i+j}\ . \tag{5}$$

Thus, if the n lower-order moments $\underline{\mathrm{E}}_{0:n-1}^{\mathrm{T}} \triangleq [\mathrm{E}_0, \ldots, \mathrm{E}_{n-1}]$ are given and the moments up to E_m, $m \geq n$ are of interest, solving the linear system of equations

$$\mathbf{Q}(\underline{\eta})\cdot\underline{\mathrm{E}}_{0:n-1} = \mathbf{R}(\underline{\eta})\cdot\underline{\mathrm{E}}_{n:m} \tag{6}$$

gives the desired higher-order moments $\underline{\mathrm{E}}_{n:m}^{\mathrm{T}} \triangleq [\mathrm{E}_n, \ldots, \mathrm{E}_m]$. The linear system of equations in (6) follows from rearranging the result in (5), where the matrices $\mathbf{Q}(\underline{\eta}) \triangleq \left[\mathbf{A}(\underline{\eta})\right]_{0:n-1}$ and $\mathbf{R}(\underline{\eta}) \triangleq -\left[\mathbf{A}(\underline{\eta})\right]_{n:m}$ are based on the $(m-n+1)\times(m+1)$ matrix $\mathbf{A}(\underline{\eta})$ in (7). Here, $[\mathbf{A}]_{n:m}$ indicates the columns n to m, $n \leq m$, of matrix $\mathbf{A}$. The matrix $\mathbf{R}(\underline{\eta})$ is triangular with zeros everywhere except of the main diagonal and the n diagonals below the main diagonal. Thus, (6) can be efficiently solved by means of forward substitution.

It is worth mentioning that it is not possible in general to deduce the parameter vector $\underline{\eta}$ from given moments $\underline{\mathrm{E}}_{0:n-1}$. Again, the Gaussian density is an exception from this general statement.

$$
\begin{aligned}
\mathbf{A}(\underline{\eta}) &= \begin{cases} 1 & i = j \\ \frac{j-i}{i}\eta_{j-i} & i < j \leq i+n \\ 0 & \text{otherwise} \end{cases} \qquad (7) \\
&= \left[\underbrace{\begin{matrix} 1 & \frac{1}{1}\eta_1 & \frac{2}{1}\eta_2 & \cdots \\ 0 & 1 & \frac{1}{2}\eta_1 & \frac{2}{2}\eta_2 \\ \vdots & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & 1 \\ 0 & \cdots & 0 & 0 \end{matrix}}_{=\mathbf{Q}(\underline{\eta})} \; \underbrace{\begin{matrix} \frac{n}{1}\eta_n & 0 & 0 & \cdots & 0 \\ \cdots & \frac{n}{2}\eta_n & 0 & \cdots & 0 \\ \ddots & & \ddots & \ddots & \vdots \\ \frac{1}{m-n}\eta_1 & \frac{2}{m-n}\eta_2 & \cdots & \frac{n}{m-n}\eta_n & 0 \\ 1 & \frac{1}{m-n+1}\eta_1 & \frac{2}{m-n+1}\eta_2 & \cdots & \frac{n}{m-n+1}\eta_n \end{matrix}}_{=-\mathbf{R}(\underline{\eta})} \right]
\end{aligned}
$$

4 Gaussian Filtering

Based on the properties of exponential densities, it is now possible to derive closed-form and computationally efficient expressions for the mean and variance of the transformed random variable $\boldsymbol{y}$.

4.1 Mean Propagation

When propagating the Gaussian random variable $\boldsymbol{x}$ through the polynomial transformation $g(.)$ in (3), the mean μ_y of $\boldsymbol{y}$ can be expressed as

$$
\mu_y = \mathrm{E}\{g(\boldsymbol{x}) + \boldsymbol{w}\} = \mathrm{E}\{g(\boldsymbol{x})\} = \sum_{i=0}^{n} c_i \cdot \int x^i \cdot \mathcal{N}\left(x; \mu_x, \sigma_x^2\right) \mathrm{d}x = \sum_{i=0}^{n} c_i \cdot \underbrace{\mathrm{E}\{x^i\}}_{=\mathrm{E}_i} \,. \qquad (8)
$$

Thus, the mean μ_y results in a weighted sum of non-central moments of a Gaussian random variable. Given the first two moments $\mathrm{E}_0 = 1$ and $\mathrm{E}_1 = \mu_x$ of $\boldsymbol{x}$, all remaining moments up to order n can be calculated by means of solving (6). In doing so, (8) can be expressed as

$$
\mu_y = \underline{c}_n^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:n} = \underline{c}_n^{\mathrm{T}} \cdot \begin{bmatrix} \mathbf{I}_2 \\ \mathbf{L} \end{bmatrix} \cdot \underline{\mathrm{E}}_{0:1} \,, \qquad (9)
$$

with $\mathbf{L} \triangleq \left(\mathbf{R}(\underline{\eta})\right)^{-1}\mathbf{Q}(\underline{\eta})$, where the parameter vector $\underline{\eta}$ comprises the parameters of a (normalized) Gaussian density as defined in Section 3.1. Furthermore, $\underline{c}_n^{\mathrm{T}} \triangleq [c_0, c_1, \ldots, c_n]$ is the vector of polynomial coefficients and $\mathbf{I}_n$ is the $n \times n$ identity matrix.

It is important to note that the second equation in (9) is merely of formal use. From a computational and numerical point of view, it is recommended to first determine the missing higher-order moments $\underline{\mathrm{E}}_{2:n}$ as described in Section 3.2 by solving the linear system of equations (6) via forward substitution. In a second step, the solution for $\underline{\mathrm{E}}_{2:n}$ is applied to the first equation in (9).

4.2 Variance Propagation

In a similar fashion as before, the variance σ_y^2 of $\boldsymbol{y}$ can be determined. For this purpose, the relation

$$\sigma_y^2 = \mathrm{E}\left\{(\boldsymbol{y} - \mu_y)^2\right\} = \mathrm{E}\left\{(g(\boldsymbol{x}) + \boldsymbol{w} - \mu_y)^2\right\} = \mathrm{E}\left\{g(\boldsymbol{x})^2\right\} - \mu_y^2 + \sigma_w^2 \tag{10}$$

is exploited, where both the noise variance σ_w^2 and the propagated mean μ_y are known. Merely the first term (10) has to be determined, which yields

$$\begin{aligned} \mathrm{E}\left\{g(\boldsymbol{x})^2\right\} &= \int g(x) \cdot g(x) \cdot \mathcal{N}\left(x; \mu_x, \sigma_x^2\right) \mathrm{d}x \\ &= \sum_{i=0}^{n} \sum_{j=0}^{n} c_i \cdot c_j \cdot \underbrace{\int x^{i+j} \cdot \mathcal{N}\left(x; \mu_x, \sigma_x^2\right) \mathrm{d}x}_{= \mathrm{E}_{i+j}} . \end{aligned}$$

Thus, in order to calculate the variance σ_y^2, it is necessary to consider all moments up to order $2n$. Given these moments and exploiting the fact that the product of two polynomials corresponds to a discrete convolution of the polynomials' coefficients, the variance calculation can be compactly written as

$$\begin{aligned} \sigma_y^2 &= \left(\underline{c}_n * \underline{c}_n\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:2n} - \mu_y^2 + \sigma_w^2 \\ &= \left(\mathbf{T} \cdot \underline{c}_n\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:2n} - \mu_y^2 + \sigma_w^2 , \end{aligned} \tag{11}$$

where $*$ is the discrete convolution operator. The second equality indicates an efficient matrix-vector realization of the convolution by means of the matrix $\mathbf{T}$ with entries $t_{i,j} = t_{i+1,j+1} = c_{i-j}$ if $i \in [j, j+n]$ and $t_{i,j} = 0$ otherwise, where

$i = 1,2,\ldots,2n+1$ and $j = 1,2,\ldots,n+1$. Hence, $\mathbf{T}$ is special type of matrix, namely a triangular Toeplitz matrix with only the mean diagonal and n diagonals below the main diagonal being non-zero and all elements on individual diagonals being equal.

4.3 Covariance Calculation

For exploiting the ability of calculating moments of nonlinear mappings, a common assumption for performing the measurement update in Gaussian filtering is to assume that the state and the measurement are jointly Gaussian distributed. This only requires the calculation of the covariance between state and measurement, which coincides with the covariance σ_{xy} between $\boldsymbol{x}$ and $\boldsymbol{y}$ for the considered generic transformation (3). Similar to (10), the covariance can be formulated as

$$\sigma_{xy} = \mathrm{E}\{(\boldsymbol{x} - \mu_x) \cdot (\boldsymbol{y} - \mu_y)\} = \mathrm{E}\{\boldsymbol{x} \cdot g(\boldsymbol{x})\} - \mu_x \cdot \mu_y \ , \tag{12}$$

where the expected value corresponds to

$$\begin{aligned} \mathrm{E}\{\boldsymbol{x} \cdot g(\boldsymbol{x})\} &= \int x \cdot g(x) \cdot \mathcal{N}\left(x; \mu_x, \sigma_x^2\right) \mathrm{d}x \\ &= \sum_{i=0}^{n} c_i \cdot \underbrace{\int x^{i+1} \cdot \mathcal{N}\left(x; \mu_x, \sigma_x^2\right) \mathrm{d}x}_{=\mathrm{E}_{i+1}} \ . \end{aligned}$$

This is almost identical to the mean calculation in (8) except for the shift by one in the order of the involved moments. Thus, the covariance is given by

$$\sigma_{xy} = \underline{c}_n^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{1:n+1} - \mu_x \cdot \mu_y \ , \tag{13}$$

where μ_y is already known from (9).

4.4 Polynomial Kalman Filter

The results derived in the previous sections allow the formulation of a Gaussian state estimator for polynomial nonlinearities. For this purpose, the well-known structure of the Kalman filter is exploited. The resulting *polynomial Kalman filter* (PKF) is listed in Algorithm 1 and described in detail in the following paragraphs.

Algorithm 1 Polynomial Kalman Filter (PKF)

```
    ▷ Prediction
1:  Determine moment vector $\underline{\mathrm{E}}_{0:2n_p}$ of posterior state $\boldsymbol{x}_{k-1}^e$ by solving (6)
2:  Predicted mean: $\mu_k^p = \underline{c}_{n_p}^p \cdot \underline{\mathrm{E}}_{0:n_p}$
3:  Predicted variance: $\left(\sigma_k^p\right)^2 = \left(\mathbf{T} \cdot \underline{c}_{n_p}^p\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:2n_p} - \left(\mu_k^p\right)^2 + \left(\sigma_k^w\right)^2$
    ▷ Measurement Update
4:  Determine moment vector $\underline{\mathrm{E}}_{0:2n_e}$ of predicted state $\boldsymbol{x}_k^p$ by solving (6)
5:  Measurement mean: $\mu_k^z = \underline{c}_{n_e}^e \cdot \underline{\mathrm{E}}_{0:n_e}$
6:  Measurement variance: $\left(\sigma_k^z\right)^2 = \left(\mathbf{T} \cdot \underline{c}_{n_e}^e\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:2n_e} - \left(\mu_k^z\right)^2 + \left(\sigma_k^v\right)^2$
7:  Covariance: $\sigma_k^{xz} = \left(\underline{c}_{n_e}^e\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{1:n_e+1} - \mu_k^p \cdot \mu_k^z$
8:  Kalman gain: $K_k = \sigma_k^{xz} / \left(\sigma_k^z\right)^2$
9:  Calculate posterior mean $\mu_k^e$ according to (14)
10: Calculate posterior variance $\left(\sigma_k^e\right)^2$ according to (15)
```

Prediction

Given the posterior state estimate $\boldsymbol{x}_{k-1}^e \sim f_{k-1}^e\left(x_{k-1}\right) \triangleq \mathcal{N}\left(x_{k-1}; \mu_{k-1}^e, \left(\sigma_{k-1}^e\right)^2\right)$ of the previous measurement update, the prediction from the previous time step $k-1$ to the current time step k requires the calculation of the predicted mean μ_k^p and variance $\left(\sigma_k^p\right)^2$. As the system function $a_k(.)$ is assumed to be a polynomial of degree $n_p \in \mathbb{N}$ with coefficient vector $\underline{c}_{n_p}^p$, equation (9) and (11) can be directly applied in order to determine the desired predicted moments.

Measurement Update

The measurement update aims at updating the predicted state $\boldsymbol{x}_k^p \sim f_k^p\left(x_k\right) \triangleq \mathcal{N}\left(x_k; \mu_k^p, \left(\sigma_k^p\right)^2\right)$ with the latest measurement value $\hat{z}_k$. To allow for a closed-form and computationally efficient update, a common assumption in nonlinear Kalman filtering—as in the extended Kalman filter or the unscented Kalman filter—is that the state $\boldsymbol{x}_k^p$ and the measurement $\boldsymbol{z}_k$ are jointly Gaussian. The implication of this so-called *joint Gaussian assumption* is discussed in Section 4.5. It requires to compute the joint mean vector and joint covariance matrix

$$\underline{\mu}_k^{xz} = \begin{bmatrix} \mu_k^p \\ \mu_k^z \end{bmatrix} , \quad \mathbf{C}_k^{xz} = \begin{bmatrix} \left(\sigma_k^p\right)^2 & \sigma_k^{xz} \\ \sigma_k^{xz} & \left(\sigma_k^z\right)^2 \end{bmatrix} ,$$

respectively. The posterior mean and variance are then calculated according to

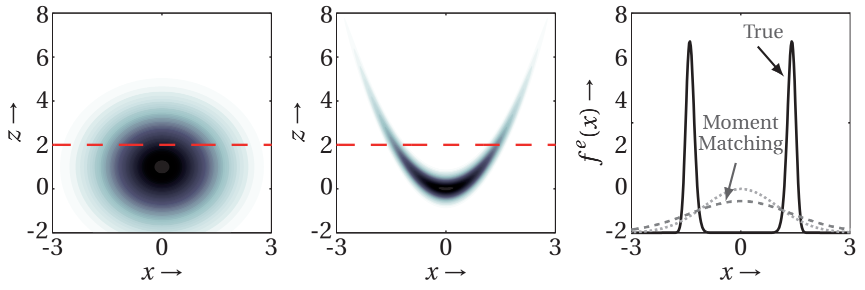
$$\mu_k^e = \mu_k^p + K_k \cdot \left(\hat{z}_k - \mu_k^z\right) , \tag{14}$$

$$\left(\sigma_k^e\right)^2 = \left(\sigma_k^p\right)^2 - K_k \cdot \sigma_k^{xz} , \tag{15}$$

which coincides with the well-known Kalman filter update step, where $K_k \triangleq \sigma_k^{xz} / \left(\sigma_k^z\right)^2$ is the Kalman gain. This measurement update requires determining the measurement mean μ_k^z and variance $\left(\sigma_k^z\right)^2$ as well as the covariance σ_k^{xz} of state and measurement. Given that the measurement function $h_k(.)$ is a polynomial of degree $n_e \in \mathbb{N}$ with coefficient vector $\underline{c}_{n_e}^e$, all three values can be calculated by means of (9), (11), and (13), respectively. Thanks to these closed-form expressions and the simple Kalman filter equations, the measurement update is straightforward to realize and computationally undemanding.

4.5 Discussion

When comparing the PKF with the approach proposed in [8], it becomes apparent that both approaches are equivalent regarding the calculated mean and variance values. However, the PKF has the following benefits. First, the involved matrices for calculating the desired moments are straightforward to determine. For instance, the matrices $\mathbf{Q}(\underline{\eta})$ and $\mathbf{R}(\underline{\eta})$ merely depend linearly on the parameters of the Gaussian density. In [8], however, the involved matrices depend on binomials coefficients, powers of the mean value, and weighted scalar products of the coefficient vector. This leads to a high computational load for determining the matrices and may cause numerical instability. Second, also the worst-case complexity is higher. While calculating the variance (13) can be performed in $\mathcal{O}(n \cdot \log n)$ if the convolution is realized via fast Fourier transform, the variance calculation in [8] scales with $\mathcal{O}\left(n^2\right)$, where $\mathcal{O}(.)$ is the big O in Landau notation. This difference is especially of importance in case of polynomials with a high degree.



(a) Gaussian approximation of the joint density. **(b)** True joint density. **(c)** True posterior density (black) and Gaussian approximations.

Figure 1: Joint density $f(x,z)$ and posterior density $f^e(x)$ for $i = 2$, i.e., for a quadratic polynomial. The red line indicates the measurement value $\hat{z} = 2$. In (c), one Gaussian approximation is obtained based on the joint Gaussian assumption (dotted) and the other via moment matching (dashed), i.e., its mean and variance coincide with the true posterior moments.

The PKF makes two different Gaussian assumptions. First, it assumes the predicted or posterior density to be Gaussian. Second, in order to perform the measurement update, it assumes that the joint density of state and measurement is Gaussian as well. If only the first Gaussian assumption would be in place, the PKF would be an exact Gaussian assumed density filter as it performs moment matching, i.e., the mean and variance calculated by PKF coincide with the true mean and variance. The additional joint Gaussian assumption, however, can result in a poor approximation of the true mean and variance, which may cause a significant loss in estimation performance or even a divergence of the estimator.

To demonstrate the effect of the joint Gaussian assumption on the estimation performance, the polynomial model

$$\boldsymbol{z} = \boldsymbol{x}^i + \boldsymbol{v} \tag{16}$$

is considered in the following, where $i > 0$ is even and the state is $\boldsymbol{x} \sim \mathcal{N}(x; 0, \sigma_x^2)$. According to (8), (10), and (12), the mean μ_z, variance σ_z^2, and covariance σ_{xz} are given by

$$\mu_z = \mathrm{E}_i \ , \quad \sigma_z^2 = \mathrm{E}_{2i} - \mathrm{E}_i + \sigma_v^2 \ , \quad \sigma_{xz} = \mathrm{E}_{i+1} \ , \tag{17}$$

respectively. Since $\boldsymbol{x}$ has zero mean, it follows that $\underline{\eta}_1 = 0$. Thus, the matrix $\mathbf{Q}(\underline{\eta})$ has only two non-zero elements $q_{11} = q_{22} = 1$, where q_{ij} is the element at row i and column j of matrix $\mathbf{Q}$. Furthermore, the matrix $\mathbf{R}(\underline{\eta})$ is zero everywhere except on the main diagonal and the second diagonal below the main diagonal. This special structure of $\mathbf{Q}(\underline{\eta})$ and $\mathbf{R}(\underline{\eta})$ leads to the conclusion that all even moments of $\boldsymbol{x}$ are non-zero and all odd moments are zero, i.e., $\mathrm{E}_i \neq 0$ and $\mathrm{E}_{i+1} = 0$ for all i being even. Hence, the covariance σ_{xz} in (17) is zero. As a result, state $\boldsymbol{x}$ and measurement $\boldsymbol{z}$ are uncorrelated and the joint Gaussian of state and measurement is axis-aligned. In Figure 1a, the joint Gaussian for $i = 2$, $\sigma_x^2 = 1$, and $\sigma_v^2 = 0.1$ is depicted.

As the covariance σ_{xz} is zero, the Kalman gain K in (14) and (15) is zero as well and no update of the predicted state occurs, i.e., the posterior state $\boldsymbol{x}^e$ is identical to the predicted state $\boldsymbol{x}^p$. In this case, a given measurement value has no impact on the estimation. This, however, is not the case, if the joint Gaussian assumption is not made. In order to demonstrate this, the measurement update is now treated from a strict Bayesian perspective. Here, the posterior state $\boldsymbol{x}^e$ is represented by the conditional density $f^e(x) \triangleq f(x|z)$ resulting from Bayes' rule

$$f^e(x) = \frac{f(z|x)\cdot f(x)}{f(z)} = \frac{f(x,z)}{f(z)} , \tag{18}$$

where $f(z|x)$ is the likelihood and $f(x)$ is the prior density of $\boldsymbol{x}$, which corresponds to the predicted Gaussian density $f^p(x) = \mathcal{N}\big(x; \mu^p, (\sigma^p)^2\big)$ in case of the considered recursive state estimation.

For the considered model (16) with a state $\boldsymbol{x}$ having zero mean, the joint Gaussian assumption leads to a factorization of the joint density $f(x,z) = f(x)\cdot f(z)$ as $\boldsymbol{x}$ and $\boldsymbol{z}$ are uncorrelated, which is equivalent to independence for Gaussian random variables. Hence, the Bayesian update in (18) degenerates to $f^e(x) = f(x) = f^p(x)$. Actually, the joint density $f(x,z)$ is an exponential density for polynomial nonlinearities. This follows from the fact that the likelihood $f(z|x)$ is defined as

$$f(z|x) \triangleq \int \delta\big(z - x^i - v\big)\cdot f(v)\,\mathrm{d}v = \mathcal{N}\big(z; x^i, \sigma_v^2\big) , \tag{19}$$

where $\delta(.)$ is the Dirac delta distribution and the second equality results from exploiting the sifting property of the Dirac delta distribution. The product of likelihood and prior density leads to the exponential density

$$\begin{aligned} f(x,z) &= f(z|x)\cdot f(x) = \mathcal{N}\big(z; x^i, \sigma_v^2\big)\cdot\mathcal{N}\big(x; 0, \sigma_x^2\big) \\ &= \exp\Big(-\log(2\pi\sigma_x\sigma_v) - \tfrac{1}{2\sigma_v^2}\cdot\big(z^2 + \tfrac{\sigma_v^2}{\sigma_x^2}x^2 - 2zx^i + x^{2i}\big)\Big)\,. \end{aligned} \tag{20}$$

This exponential joint density is depicted in Figure 1b for $i = 2$. By comparing Figure 1a with Figure 1b the difference between the true joint density and its Gaussian approximation becomes apparent. Given a measurement value $\hat{z} = 2$, Figure 1c depicts the posterior densities obtained for the Gaussian joint density and the true exponential joint density. It can be seen that the true posterior is bimodal, which only can be coarsely approximated by a Gaussian density. Furthermore, due to the joint Gaussian assumption, the Gaussian posterior does not even match the true posterior mean and variance.

The true posterior is an exponential density, since the joint density $f(x,z=\hat{z})$ is exponential and $f(\hat{z})$ is merely a normalization constant for a given measurement value $\hat{z}$. Thus, the posterior is a conjugate density of the prior density in the case of polynomial nonlinearities. Unfortunately, a general exponential density is not well suited for recursive processing for mainly two reasons: First, the prediction step as described above requires the availability of the moments $\mathrm{E}_{0:2i-1}$, but for exponential densities the calculation of these moments cannot be performed in closed form. Second, even if the moments were available, the prediction step itself merely provides the predicted moments and no analytic density representation. Determining an exponential density that matches given moments is also not possible in closed form. To overcome these limitations, a novel approach for accurately determining the true posterior mean and variance is proposed in the next section. In doing so, a computationally efficient, recursive Gaussian filter without the joint Gaussian assumption is obtained.

5 Homotopic Bayesian Measurement Update

In this section, a new method for directly calculating the moments of the posterior density will be introduced that does not require the joint Gaussian assumption and that provides a much higher estimation quality.

The key idea is to transform the known moments of the prior Gaussian density continuously into the desired posterior moments. For this purpose, *homotopy continuation* for calculating the moments of exponential densities as proposed in [10] is exploited. By means of a so-called progression parameter $\gamma \in [0;1]$ the posterior density $f^e(x)$ is parameterized in such a way that for $\gamma = 0$ the posterior density corresponds to prior Gaussian density $f^p(x)$ and for $\gamma = 1$ the posterior density corresponds to the true exponential density. For the initial value $\gamma = 0$, the moments are known as they coincide with the moments of the Gaussian prior. Incrementing the progression parameter causes moment variations described by means of a *system of ordinary differential equations* (ODEs). Solving this system of ODEs for $\gamma \in [0;1]$ gives the desired posterior moments.

5.1 Parameterization

To allow for homotopy continuation, the Bayesian measurement update in (18) is parameterized according to

$$f^e(x;\gamma) = \left(\tfrac{1}{f(\hat{z})} \cdot f(\hat{z}|x)\right)^{\gamma} \cdot f^p(x) \tag{21}$$

for a given measurement value $\hat{z}$, with likelihood $f(\hat{z}|x) = \mathcal{N}\big(\hat{z}; h(x), \sigma_v^2\big)$ according to (19) for a polynomial measurement model (2). Further, $f^e(x;\gamma)$ is a parameterized version of the posterior density. For $\gamma = 1$, this parameterized measurement update corresponds to the standard Bayes' rule, while for $\gamma = 0$, the prior density $f^p(x)$ is directly assigned to the posterior density, i.e., no measurement update is performed.

In order to simplify the following calculations, the normalization constant $1/f(\hat{z})$ in (21) is ignored, which is without any disadvantages. Since the zeroth-order moment E_0, which is reciprocal to the normalization constant, will be calculated as well, ex post division of all higher-order moments by E_0 leads to the correct results (see Section 5.3).

Due to ignoring the normalization constant, merely the proportional relation

$$f^e(x;\gamma) \propto f\big(x, \hat{z}; \underline{\eta}(\gamma)\big) \triangleq f(\hat{z}|x)^{\gamma} \cdot f^p(x) \tag{22}$$

is considered instead of (21), where the *parameterized joint density* $f\big(x, \hat{z}; \underline{\eta}(\gamma)\big) = \exp\big(\underline{\eta}(\gamma)^{\mathrm{T}} \cdot \underline{x}\big)$ is an exponential density similar to (20) with parameter vector

$$\underline{\eta}(\gamma) \triangleq \underline{\eta}^p + \gamma \cdot \underline{\eta}^l \in \mathbb{R}^{2n_e+1}$$

depending on γ. Here, $\underline{\eta}^p$ is the parameter vector of the Gaussian prior $f^p(x)$ and $\underline{\eta}^l$ is the parameter vector of the likelihood $f(\hat{z}|x)$ according to

$$\underline{\eta}^l = \begin{bmatrix} -\log(\sqrt{2\pi}\sigma_v) - \dfrac{\hat{z}^2}{2\sigma_v^2} \\ \dfrac{\hat{z}}{\sigma_v^2} \cdot \underline{c}^e_{n_e} \\ \underline{0} \end{bmatrix} - \tfrac{1}{2\sigma_v^2} \cdot \left(\underline{c}^e_{n_e} * \underline{c}^e_{n_e} \right) ,$$

with $\underline{c}^e_{n_e}$ being the coefficient vector of the measurement function $h(.)$ and $\underline{0}$ being a vector of zeros of appropriate dimension. The parameter vector $\underline{\eta}(\gamma)$ directly reflects the continuation in (22).

It is worth mentioning that the parameterized joint density always is a valid exponential density for each $\gamma \in [0;1]$. As it directly depends on the Gaussian measurement noise $\boldsymbol{v}$ and the Gaussian prior $f^p(x)$, the highest-order monomial in $\underline{x}$ is even and the last element in $\underline{\eta}^l$ is negative.

5.2 System of Ordinary Differential Equations

By a continuous modification of the progression parameter γ, a continuous variation of the parameter vector $\underline{\eta}(\gamma)$ is achieved. This in turn results in a variation of the moments $\mathrm{E}_i\big(\underline{\eta}(\gamma)\big)$, $i = 0,\ldots,2n_e - 1$, of the parameterized joint density $f(x,\hat{z};\underline{\eta}(\gamma))$. These moment variations depending on γ can be described by means of a system of ODEs by calculating the partial derivatives $\dot{\mathrm{E}}_i \triangleq \frac{\partial \mathrm{E}_i(\underline{\eta}(\gamma))}{\partial \gamma}$ for $i = 0,\ldots,2n_e - 1$. The partial derivative of the ith-order moment is given by

$$\begin{aligned}
\dot{\mathrm{E}}_i &= \frac{\partial \mathrm{E}_i\big(\underline{\eta}(\gamma)\big)}{\partial \gamma} = \left[\frac{\partial \mathrm{E}_i}{\partial \underline{\eta}} \bigg|_{\underline{\eta}=\underline{\eta}(\gamma)} \right]^{\mathrm{T}} \cdot \frac{\partial \underline{\eta}(\gamma)}{\partial \gamma} \\
&= \left(\frac{\partial \underline{\eta}(\gamma)}{\partial \gamma} \right)^{\mathrm{T}} \cdot \int x^i \cdot \frac{\partial f\big(x,\hat{z};\underline{\eta}\big)}{\partial \underline{\eta}} \bigg|_{\underline{\eta}=\underline{\eta}(\gamma)} \mathrm{d}x \\
&= \left(\frac{\partial \underline{\eta}(\gamma)}{\partial \gamma} \right)^{\mathrm{T}} \cdot \int x^i \begin{bmatrix} 1 \\ x \\ \vdots \\ x^{2n_e} \end{bmatrix} \exp\left(\underline{\eta}(\gamma)^{\mathrm{T}} \cdot \underline{x} \right) \mathrm{d}x \\
&= \left[\mathrm{E}_i\big(\underline{\eta}(\gamma)\big) \quad \mathrm{E}_{i+1}\big(\underline{\eta}(\gamma)\big) \quad \cdots \quad \mathrm{E}_{i+2n_e}\big(\underline{\eta}(\gamma)\big) \right] \cdot \underline{\eta}^l ,
\end{aligned} \tag{23}$$

which relates the variation of the ith-order moment to moments of order up to $i+2n_e$. In the following, $\mathrm{E}_i^{(\gamma)} \triangleq \mathrm{E}_i\big(\underline{\eta}(\gamma)\big)$ is used as shorthand term.

With the result in (23), the system of ODEs comprising the moment variations of all moments up to order $2n_e-1$ is

$$\underline{\dot{\mathrm{E}}}_{0:2n_e-1} = \Big(\mathbf{T}\big(\underline{\eta}^l\big)\Big)^{\mathrm{T}} \cdot \underline{\mathrm{E}}^{(\gamma)}_{0:4n_e-1} = \mathbf{T}^l \cdot \underline{\mathrm{E}}^{(\gamma)}_{0:2n_e-1} + \mathbf{T}^h \cdot \underline{\mathrm{E}}^{(\gamma)}_{2n_e:4n_e-1} \,,$$

where $\mathbf{T}(\underline{\eta}^l)$ is a Toeplitz matrix with entries $t_{i,j} = t_{i+1,j+1} = \eta^l_{i-j}$ if $i \in [j, j+2n_e]$ and $t_{i,j} = 0$ otherwise, where $i = 1,2,\dots,4n_e$ and $j = 1,2,\dots,2n_e$. The $2n_e \times 2n_e$ matrices $\mathbf{T}^l$ and $\mathbf{T}^h$ are sub-matrices of $\mathbf{T}\big(\underline{\eta}^l\big)$ according to $\mathbf{T}\big(\underline{\eta}^l\big) = \big[\mathbf{T}^l\ \mathbf{T}^h\big]^{\mathrm{T}}$. Besides the lower-order moments $\underline{\mathrm{E}}^{(\gamma)}_{0:2n_e-1}$, the system of ODEs also depends on the higher-order moments $\underline{\mathrm{E}}^{(\gamma)}_{2n_e:4n_e-1}$. Fortunately, with the result of (6), the dependence on the higher-order moments can be resolved. In doing so, the system of ODEs can be reformulated into

$$\underline{\dot{\mathrm{E}}}_{0:2n_e-1} = \Big(\mathbf{T}^l + \mathbf{T}^h \Big(\mathbf{R}\big(\underline{\eta}(\gamma)\big)\Big)^{-1} \mathbf{Q}\big(\underline{\eta}(\gamma)\big)\Big) \cdot \underline{\mathrm{E}}^{(\gamma)}_{0:2n_e-1} \tag{24}$$

with matrices $\mathbf{R}\big(\underline{\eta}(\gamma)\big)$ and $\mathbf{Q}\big(\underline{\eta}(\gamma)\big)$ according to (7), which vary with γ as they depend on the parameters of the parameterized joint density $f\big(x,\hat{z};\underline{\eta}(\gamma)\big)$.

5.3 Initialization and Solution

The system of ODEs in (24) describes the moment variations caused by homotopy continuation of the Bayesian measurement update (22) in a very elegant manner. For solving this system of ODEs, standard numerical solvers based on the Runge-Kutta method [9] can be employed. The solution describes a trajectory of the moments $\underline{\mathrm{E}}^{(\gamma)}_{0:2n_e-1}$ depending on different values of the progression parameter γ. The desired moments of the posterior density $f^e(x)$ are obtained for $\gamma = 1$, i.e., $\underline{\mathrm{E}}^{(1)}_{0:2n_e-1}$ comprises the result. As mentioned above, the moments in $\underline{\mathrm{E}}^{(1)}_{0:2n_e-1}$ are unnormalized as merely the proportional relation (22) was considered. Multiplying $\underline{\mathrm{E}}^{(1)}_{0:2n_e-1}$ with the normalization constant

$$\alpha \triangleq \frac{1}{f(\hat{z})} = \frac{1}{\mathrm{E}_0^{(1)}} \tag{25}$$

yields the actual posterior moments.

Please note that the matrix $\mathbf{R}\big(\underline{\eta}(\gamma)\big)$ in (24) is singular for $\gamma = 0$. To avoid an inversion of this matrix for $\gamma = 0$, an initialization procedure is proposed that determines an initial solution $\underline{\mathrm{E}}_{0:2n_e-1}^{(\Delta\gamma)}$ for the first solution step, with $\Delta\gamma$ being a small positive step value[2]. Based on the initial solution $\underline{\mathrm{E}}_{0:2n_e-1}^{(\Delta\gamma)}$, the system of ODEs in (24) is then solved in a standard fashion for $\gamma \in [\Delta\gamma; 1]$.

To determine the initial solution, the moment calculation (4) is expanded around $\gamma = 0$ via a first-order Taylor-series according to

$$\begin{aligned}
\mathrm{E}_i^{(\Delta\gamma)} &\approx \mathrm{E}_i^{(0)} + \Delta\gamma \cdot \frac{\partial \mathrm{E}_i^{(\gamma)}}{\partial \underline{\eta}^{\mathrm{T}}} \cdot \frac{\partial \underline{\eta}}{\partial \gamma}\bigg|_{\underline{\eta}=\underline{\eta}(\gamma),\gamma=0} \\
&= \mathrm{E}_i\left(\underline{\eta}^p\right) + \Delta\gamma \cdot \left[\frac{\partial \mathrm{E}_i\left(\underline{\eta}\right)}{\partial \underline{\eta}}\bigg|_{\underline{\eta}=\underline{\eta}^p}\right]^{\mathrm{T}} \cdot \underline{\eta}^l
\end{aligned} \tag{26}$$

for each moment i, where $\mathrm{E}_i\big(\underline{\eta}^p\big)$ are the moments of the predicted Gaussian state $\boldsymbol{x}^p \sim f^p(x) = \mathcal{N}\big(x; \mu^p, (\sigma^p)^2\big)$. The second summand in (26) is given by (23) for $\underline{\eta}(\gamma) = \underline{\eta}^p$. This derivative merely depends on the Gaussian prior density $f^p(x)$. Thus, all higher-order moments in (23) can be determined via solving the moment recursion in (6).

5.4 Homotopic Polynomial Gaussian Filter

The novel homotopic polynomial Gaussian filter (HPGF) for polynomial non-linearities is summarized in Algorithm 2. The prediction step coincides with the prediction of the PKF as proposed in Section 4. The measurement update utilizing homotopy continuation for calculating the posterior moments consists of three operations for each time step k: First, initialization as proposed in the previous section. Second, solving the system of ODEs (24). Finally, calculating the posterior mean and variance by correcting the ODE solution $\underline{\mathrm{E}}_{0:2n_e-1}^{(1)}$ with the normalization constant (25) according to

$$\mu_k^e = \alpha \cdot \mathrm{E}_1^{(1)} , \tag{27}$$

$$\left(\sigma_k^e\right)^2 = \alpha \cdot \mathrm{E}_2^{(1)} - \left(\mu_k^e\right)^2 , \tag{28}$$

which yields the desired posterior state estimate $\boldsymbol{x}_k^e \sim f_k^e(x_k) = \mathcal{N}\big(x_k; \mu_k^e, (\sigma_k^e)^2\big)$.

2 In the simulations in Section 6, $\Delta\gamma = 10^{-7}$ is used.

Algorithm 2 Homotopic Polynomial Gaussian Filter (HPGF)

```
   ▷ Prediction
1: Determine moment vector $\underline{\mathrm{E}}_{0:2n_p}$ of posterior state $\boldsymbol{x}_{k-1}^e$ by solving (6)
2: Predicted mean: $\mu_k^p = \underline{c}_{n_p}^p \cdot \underline{\mathrm{E}}_{0:n_p}$
3: Predicted variance: $\left(\sigma_k^p\right)^2 = \left(\mathbf{T} \cdot \underline{c}_{n_p}^p\right)^{\mathrm{T}} \cdot \underline{\mathrm{E}}_{0:2n_p} - \left(\mu_k^p\right)^2 + \left(\sigma_k^w\right)^2$
   ▷ Measurement Update
4: Determine initial solution according to (26)
5: Solve system of ODEs (24) for $\gamma \in [\Delta\gamma; 1]$
6: Calculate posterior mean $\mu_k^e$ according to (27)
7: Calculate posterior variance $\left(\sigma_k^e\right)^2$ according to (28)
```

6 Results

In the following, both Gaussian filters the PKF and the HPGF proposed in this paper are compared to state-of-the-art filters by means of numerical simulations.

6.1 Moment Homotopy Examples

For the first simulation, the polynomial measurement model (16) is revisited, where now merely the quadratic (order $i = 2$) and the cubic (order $i = 3$) case are considered. Furthermore, the state estimate $\boldsymbol{x} \sim \mathcal{N}(x; 0, 1)$ is standard Gaussian distributed, the measurement value is $\hat{z} = 1$, and $(\sigma_v)^2 = 0.1$ is the variance of the measurement noise.

In Table 1, the posterior moments calculated by the proposed HPGF and PKF are compared with the true moments and the results obtained by means of the extended Kalman filter (EKF). The true moments have been calculated via numerical integration. It can be seen that the results of the HPGF coincide with the true moments. Compared with numerical integration, the HPGF has the benefits of a significantly lower computational burden and that no integration interval needs to be determined. In Figure 2, the trajectories of the posterior moments resulting from the homotopy continuation are shown. It can be seen how the moments of the prior Gaussian are transformed into the true posterior moments.

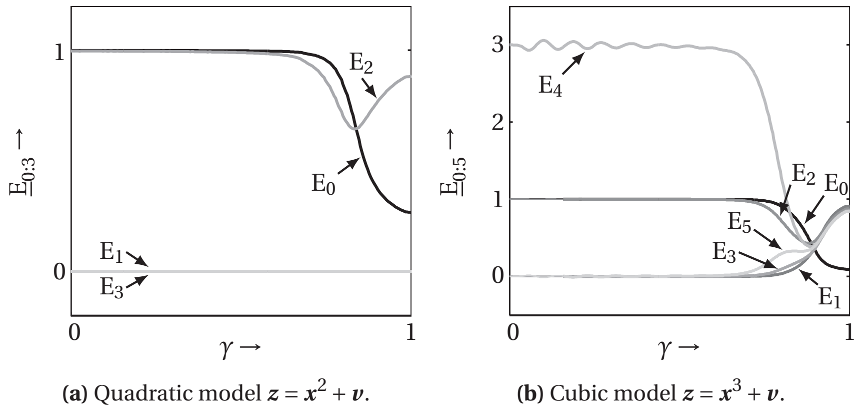


Figure 2: Trajectories of posterior moments.

Table 1: Comparison of the posterior moments calculated via different approaches for the quadratic and cubic model.

	Quadratic				Cubic			
	True	HPGF	EKF	PKF	True	HPGF	EKF	PKF
E_0	0.2664	0.2664	–	–	0.0932	0.0932	–	–
E_1	0.0	0.0	0.0	0.0	0.9113	0.9113	0.0	0.1987
E_2	0.8820	0.8820	1.0	1.0	0.8760	0.8760	1.0	0.4434
E_3	0.0	0.0	–	–	0.8526	0.8525	–	–
E_4	–	–	–	–	0.8442	0.8442	–	–
E_5	–	–	–	–	0.8457	0.8456	–	–

EKF and PKF both rely on the joint Gaussian assumption. Thus, the moments of both filters differ significantly from the true moments (see Table 1). As discussed in Section 4.5, for $i = 2$ no update of the state estimate can be performed. For the cubic case, however, an update should be available, which is true for the PKF. The EKF, however, suffers from the linearization, which results in a zero Kalman gain. Thus, no update is performed at all. This example clearly shows the benefits of avoiding the joint Gaussian assumption.

6.2 Chaotic Synchronization

In this example, the polynomial system model

$$\boldsymbol{x}_{k+1} = T_4(\boldsymbol{x}_k) + \boldsymbol{w}_k \tag{29}$$

as used in [8] is considered, where $T_i(x) = 2x \cdot T_{i-1}(x) - T_{i-2}(x)$ for $i = 2,3,\ldots$ is the ith Chebyshev polynomial, with $T_0(x) = 1$ and $T_1(x) = x$. It is known that models as in (29) generate chaotic sequences [11], which is of practical use in securing communication systems. The true initial state $\boldsymbol{x}_0$ a time step $k = 0$ is assumed to be Gaussian with mean $\mu_0^x = 0.3$ and variance $\left(\sigma_0^x\right)^2 = 0.25$.

At first, a linear measurement model

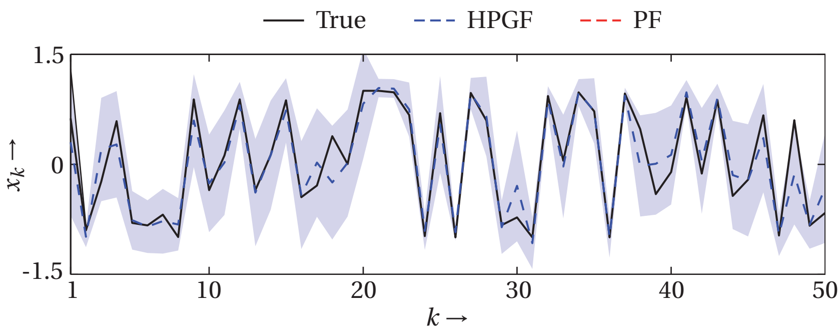
$$\boldsymbol{z}_k = \boldsymbol{x}_k + \boldsymbol{v}_k \tag{30}$$

is employed, with measurement noise variance $(\sigma_v)^2 = 10^{-2} \cdot (\sigma_w)^2$ and system noise variance being $(\sigma_w)^2 = 10^{-2}$ (high noise) or $(\sigma_w)^2 = 10^{-3}$ (low noise). As the measurement model is linear, the joint Gaussian assumption is correct. Thus, the HPGF does not need to be considered here. PKF is compared against EKF, unscented Kalman filter (UKF), and a particle filter (PF) with systematic resampling [5] and 500 samples. The latter is the only non-Gaussian filter. For all filters, 50 Monte Carlo simulation runs with identical noise sequences are performed, where the estimates are calculated for 50 time steps. As performance indicators, the root mean square error (rmse), the normalized estimation error squared (nees), and the runtime for 50 time steps are employed.

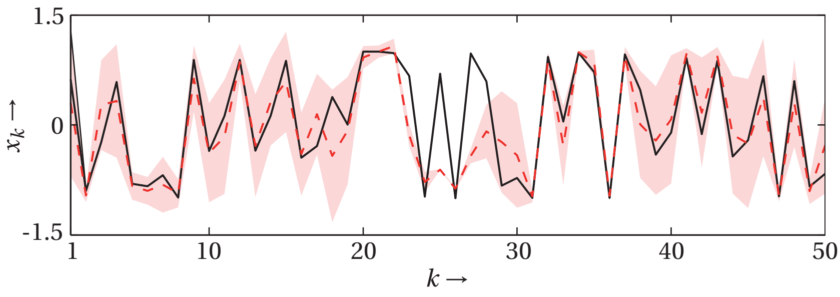
In Table 2, the average rmse, nees, and runtime over all Monte Carlo runs are listed for all filters and for both noise cases. For high noise, the proposed PKF outperforms all Gaussian filters in terms of rmse and nees, i.e., its estimates are

Table 2: Average rmse, nees, and runtime for the chaotic system model (29) and the linear measurement model (30).

	$\sigma_w^2 = 10^{-2}$				$\sigma_w^2 = 10^{-3}$			
	EKF	UKF	PF	PKF	EKF	UKF	PF	PKF
rmse	0.410	0.336	**0.292**	0.316	0.148	**0.118**	0.268	**0.118**
nees	4.737	1.550	1.168	**1.041**	7.279	**1.110**	–	1.129
time	**0.016**	0.038	0.109	0.017	**0.017**	0.037	0.102	0.018



(a) Homotopic polynomial Gaussian filter (HPGF).



(b) Particle filter (PF).

Figure 3: State trajectory (black, solid line) and the estimates of HPGF and PF together with the corresponding 2-sigma confidence regions.

closest to the true system state (low rmse) and at the same time the estimates are not overly confident (low nees). Furthermore, the matrix-vector terms proposed for the PKF allow for a runtime being close to the EKF, which is known to be the fastest Gaussian filter.

For the low noise case, UKF performs best in terms of estimation error, but PKF is very close to it. PF occasionally suffers from particle depletion, i.e., most of the particles converge towards the same state, which coincides with an overconfident estimate and thus an exceedingly high nees value. Even significantly increasing the number of particles or using different resampling techniques yields no improvement.

If a cubic measurement model $\boldsymbol{z}_k = \frac{\boldsymbol{x}_k^3}{20} + \boldsymbol{v}_k$ with measurement noise $\boldsymbol{v}_k \sim \mathcal{N}(v_k; 0, 10^{-5})$ instead of the linear model (30) is utilized, it turns out that *all* Gaussian filters relying on the joint Gaussian assumption diverge. The HPGF, however, is able to provide valid estimates. In Figure 3a, an exemplary state trajectory is depicted. The estimates of HPGF accurately follow the true state. Furthermore, the true state is always within the 2-sigma confidence region of the estimates. The result of the PF depicted in Figure 3b is less accurate and shows sample depletion from time step $k = 20$ to $k = 27$.

7 Conclusion and Future Work

Two methods for the efficient calculation of moments have been introduced in this paper. The first method named polynomial Kalman filter (PKF) efficiently calculates the moments of a polynomial mapping of a Gaussian random variable. When applied to the prediction step, the moment calculation is exact. For the filter step, this method leads to a superior estimation performance compared to existing Gaussian filters. However, the typical additional Gaussian assumption for the joint density of state and measurement is required that can cause highly inaccurate or even diverging estimates. Hence, in order to avoid this assumption, a second method named homotopic polynomial Gaussian filter (HPGF) for the almost exact calculation of the posterior moments in the filter step is introduced. This method is based on a homotopy continuation for polynomial nonlinearities. Combining both methods results in a Gaussian assumed density filter for polynomial nonlinearities that can compete even with non-Gaussian filters.

Future work is devoted to extend the proposed Gaussian filters to the multi-dimensional case. Furthermore, resolving the second limitation mentioned at the end of Section 4.5 allows an extension towards a full exponential filter.

References

[1] Ienkaran Arasaratnam and Simon Haykin. Cubature Kalman Filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, June 2009.

[2] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[3] Michael Basin. Optimal Filtering for Partially Measured Polynomial System States. In *Proceedings of the 2005 American Control Conference (ACC)*, pages 4022–4027, Portland, OR, USA, June 2005.

[4] Damiano Brigo, Bernard Hanzon, and Francois Le Gland. A Differential Geometric Approach to Nonlinear Filtering: the Projection Filter. Technical Report 2598, Insitut National De Recherche en Informatique et en Automatique, June 1995.

[5] James Carpenter, Peter Clifford, and Paul Fearnhead. Improved particle filter for nonlinear problems. In *IEE Proceedings Radar, Sonar and Navigation*, volume 146, pages 2–7, February 1999.

[6] Marco F. Huber and Uwe D. Hanebeck. Gaussian Filter based on Deterministic Sampling for High Quality Nonlinear Estimation. In *Proceedings of the 17th IFAC World Congress*, Seoul, Republic of Korea, July 2008.

[7] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*. Dover Publications, Inc., 2007.

[8] Mihai Bogdan Luca, Stéphane Azou, Gilles Burel, and Alexandru Serbanescu. On Exact Kalman Filtering of Polynomial Systems. *IEEE Transactions on Circuits and Systems—I: Regular Papers*, 53(6):1329–1340, June 2006.

[9] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*, chapter 17: Integration of Ordinary Differential Equations. Cambridge University Press, 3rd edition, 2007.

[10] Andreas Rauh and Uwe D. Hanebeck. Calculating Moments of Exponential Densities Using Differential Algebraic Equations. *IEEE Signal Processing Letters*, 10(5):144–147, May 2003.

[11] Theodore Rivlin. *Chebyshev Polynomials.* New York: Wiley, 1990.

[12] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics.* The MIT Press, 2005.

[13] Eric A Wan and Rudolph van der Merwe. The Unscented Kalman Filter. In Simon Haykin, editor, *Kalman Filtering and Neural Networks,* chapter The Unscented Kalman Filter, pages 221–280. John Wiley & Sons, Inc., 2001.

Paper E

(Semi-)Analytic Gaussian Mixture Filter

Authors: Marco F. Huber, Frederik Beutler, and Uwe D. Hanebeck

Edited version of the paper:

M. F. Huber, F. Beutler, and U. D. Hanebeck. (Semi-)Analytic Gaussian Mixture Filter. In *Proceedings of the 18th IFAC World Congress*, pages 10014–10020, Milano, Italy, August 2011.

(Semi-)Analytic Gaussian Mixture Filter

Marco F. Huber[*], Frederik Beutler[**], and Uwe D. Hanebeck[**]

[*] Variable Image Acquisition and Processing Research Group
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB
Karlsruhe, Germany
marco.huber@ieee.org

[**] Intelligent Sensor-Actuator-Systems Laboratory (ISAS)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{beutler|uwe.hanebeck}@ieee.org

Abstract

In nonlinear filtering, special types of Gaussian mixture filters are a straightforward extension of Gaussian filters, where linearizing the system model is performed individually for each Gaussian component. In this paper, two novel types of linearization are combined with Gaussian mixture filters. The first linearization is called analytic stochastic linearization, where the linearization is performed analytically and exactly, i.e., without Taylor-series expansion or approximate sample-based density representation. In cases where a full analytical linearization is not possible, the second approach decomposes the nonlinear system into a set of nonlinear subsystems that are conditionally integrable in closed form. These approaches are more accurate than fully applying classical linearization.

1 Introduction

Determining the internal state of a dynamic system is essential in many applications, e.g., target tracking or simultaneous localization and mapping (SLAM). Due to disturbances in the measurement process and imperfect system models, the Bayesian estimation framework is often applied in order to deal with these uncertainties. For linear systems affected with Gaussian noise, the Kalman filter is the optimal estimator and Bayesian estimation can be performed in closed form. In case of nonlinearities and/or non-Gaussian noise, the Kalman filter is no longer optimal or even not applicable. In order to achieve an estimate for the internal state of the system, approximate state estimators have to be employed.

A common restriction of approximate state estimators is to assume a (jointly) Gaussian representation of the system state and the measurement. Estimators corresponding to these so-called Gaussian filters are for example the Extended Kalman Filter (EKF), the Unscented Kalman Filter (UKF, [8]), analytic approaches (AGF, [3]), and semi-analytic approaches (SAGF, [5]). Due to the Gaussian assumption, a linearization of the nonlinear system models is performed, which can be explicit (EKF) or implicit (UKF, AGF, SAGF) as discussed in Section 4.1. Obviously, in problems with strong nonlinearities that cause non-Gaussian densities, e.g., multi-modal or heavily skewed densities, the estimation accuracy of Gaussian filters is limited. But on the other hand, the computational complexity is only polynomial with the dimension of the state space.

Estimators especially designed for nonlinear non-Gaussian problems are for instance the grid filter or the particle filters [2], which make no specific assumptions about the shape of the density function. Hence, these approaches can be arbitrarily accurate, but their complexity is exponential.

A trade-off between both worlds are the so-called Gaussian mixture filters (GMFs, see Section 3), which are often applied to SLAM [9, 11]. Gaussian mixtures are very convenient for filtering purposes as with an increasing number of mixture components, a Gaussian mixture can approximate any density function with arbitrary accuracy [12]. GMFs utilize a weighted sum of Gaussian densities and typically rely on individual linearizations for each Gaussian component. By applying a bank of EKFs, individual linearization is achieved via component-wise first-order Taylor-series expansion [1]. A bank of UKFs corresponds to a sample-based stochastic linearization [13] for each component.

In this paper, two novel types of linearization are utilized for GMFs. For some classes of nonlinear functions, e.g., polynomial or trigonometric functions or combinations of them, stochastic linearization can be performed analytically and exactly (see Section 4.2). Instead of a sample point representation, the whole Gaussian density is propagated through the nonlinear function for linearization. The resulting filter is named the analytic Gaussian mixture filter (A-GMF). For arbitrary nonlinearities, the semi-analytic Gaussian mixture filter (SA-GMF) introduced in Section 4.3 utilizes a linearization approach that relies on a decomposition of the nonlinear system into integrable substructures such that one part of the problem can be solved analytically and for the remaining part a sample-based stochastic linearization is employed. This decomposition is much more general than the usual decomposition into conditional linear subsystems via Rao-Blackwellization. Since both GMFs at least partly rely on analytic linearization, much better local approximations and thus, more accurate estimation results are achieved compared to GMFs proposed by [1] or [13]. This is demonstrated via simulations in Section 5, where a robot with tricycle kinematics is considered that measures distances to landmarks.

2 Problem Formulation

A nonlinear dynamic system is described by its system and measurement equation according to

$$\underline{\boldsymbol{x}}_{k+1} = \underline{a}_k\left(\underline{\boldsymbol{x}}_k, \underline{u}_k, \underline{\boldsymbol{w}}_k\right) , \tag{1}$$

$$\underline{\boldsymbol{y}}_k = \underline{h}_k\left(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{v}}_k\right) , \tag{2}$$

where $\underline{\boldsymbol{x}}_k$ is the internal state, $\underline{\boldsymbol{w}}_k$ and $\underline{\boldsymbol{v}}_k$ are the white system and measurement noise processes, $\underline{\boldsymbol{y}}_k$ is the measurement process, and $\underline{u}_k$ a known control input. The known function $\underline{a}_k(\cdot,\cdot,\cdot)$ describes the evolution of the system over time and the known function $\underline{h}_k(\cdot,\cdot)$ the mapping between the internal state and the output of the system. Please note that actual measurement values $\underline{\hat{y}}_k$ are realizations of $\underline{\boldsymbol{y}}_k$.

The random variables, e.g., $\underline{\boldsymbol{x}}_k$, are denoted be bold font letters and are described by probability density functions $f(\cdot)$. For recursive estimation, the Bayesian estimation framework is applied, which consists of the prediction and filter step. In the following, a short overview of the Bayesian estimation framework is given.

2.1 Prediction Step

In the prediction step, the estimated density of the state $f^e\left(\underline{x}_k\right)$ is propagated to the next time step $k+1$ by means of the Chapman-Kolmogorov equation

$$f^p\left(\underline{x}_{k+1}\right) = \iint \underbrace{f\left(\underline{x}_{k+1}|\underline{x}_k, \underline{w}_k\right)}_{\delta\left(\underline{x}_{k+1}-\underline{a}_k(\underline{x}_k,\underline{u}_k,\underline{w}_k)\right)} f^e\left(\underline{x}_k\right) f\left(\underline{w}_k\right) \mathrm{d}\underline{x}_k \, \mathrm{d}\underline{w}_k , \tag{3}$$

where $\delta(\cdot)$ is the Dirac delta distribution and $f^p\left(\underline{x}_{k+1}\right)$ is the predicted density of the state.

2.2 Filter Step

In the filter step, the predicted density $f^p(\underline{x}_k)$ from the last prediction step is updated based on Bayes' rule

$$f^e\left(\underline{x}_k\right) = \tfrac{1}{c_k} f\left(\underline{\hat{y}}_k | \underline{x}_k\right) f^p\left(\underline{x}_k\right) \ , \tag{4}$$

where $c_k = \int f\left(\underline{\hat{y}}_k | \underline{x}_k\right) f^p\left(\underline{x}_k\right) \mathrm{d}\underline{x}_k$ is a normalization constant and $f\left(\underline{\hat{y}}_k | \underline{x}_k\right)$ is the Likelihood function given by

$$f(\underline{\hat{y}}_k | \underline{x}_k) = \int \underbrace{f\left(\underline{\hat{y}}_k | \underline{x}_k, \underline{v}_k\right)}_{\delta\left(\underline{\hat{y}}_k - \underline{h}_k(\underline{x}_k, \underline{v}_k)\right)} f\left(\underline{v}_k\right) \mathrm{d}\underline{v}_k \ .$$

The function $f\left(\underline{\hat{y}}_k | \underline{x}_k, \underline{v}_k\right)$ depends on the nonlinear function in (2) and the current measurement value $\underline{\hat{y}}_k$.

For simplicity and brevity, the time index k is omitted and for the prediction step, the variables $\underline{x}_k$ and $\underline{x}_{k+1}$ are replaced by $\underline{x}^e$ and $\underline{x}^p$, respectively.

3 Gaussian Mixture Filter

The basic idea behind a GMF is to apply linearization individually to each Gaussian component of a given Gaussian mixture for approximate prediction and filtering.

3.1 Prediction Step

In the prediction step, it is assumed that the result of the previous filter step is represented by means of the Gaussian mixture

$$f^e\left(\underline{x}^e\right) = \sum_{i=1}^{L} \omega_i^e \cdot \mathcal{N}\left(\underline{x}^e; \underline{\mu}_i^e, \mathbf{C}_i^e\right) \ , \tag{5}$$

where ω_i^e are non-negative weighting factors summing up to one and $\mathcal{N}(\underline{x};\underline{\mu},\mathbf{C})$ is a Gaussian density with mean vector $\underline{\mu}$ and covariance matrix $\mathbf{C}$. L is the number of components. The mixture in (5) is used in (3), which results in

$$f^p\left(\underline{x}^p\right) \approx \sum_{i=1}^{L} \omega_i^p \cdot f_i^p\left(\underline{x}^p\right), \tag{6}$$

where the weighting factors[1] are $\omega_i^p = \omega_i^e$ and $f_i^p\left(\underline{x}^p\right)$ is the predicted density

$$f_i^p\left(\underline{x}^p\right) = \iint f\left(\underline{x}^p|\underline{x}^e,\underline{w}\right) \cdot \mathcal{N}\left(\underline{x}^e;\underline{\mu}_i^e,\mathbf{C}_i^e\right) \cdot f(\underline{w})\,\mathrm{d}\underline{x}^e\,\mathrm{d}\underline{w} \tag{7}$$

of the ith component. In general, the integral in (7) cannot be solved in closed form. To simplify this problem, the predicted density of each component is individually approximated by a Gaussian density $f_i^p\left(\underline{x}^p\right) \approx \mathcal{N}\left(\underline{x}^p;\underline{\mu}_i^p,\mathbf{C}_i^p\right)$. Thus, it remains to calculate the moments $\underline{\mu}_i^p$ and $\mathbf{C}_i^p$ via moment matching. The mean $\underline{\mu}_i^p$ of the ith component for instance is

$$\underline{\mu}_i^p = \int \underline{x}^p \cdot f_i^p\left(\underline{x}^p\right)\mathrm{d}\underline{x}^p\,. \tag{8}$$

Using (7) in (8) and exploiting the sifting property of the Dirac delta distribution, the mean results in

$$\underline{\mu}_i^p = \iint \underline{a}(\underline{x}^e,\underline{u},\underline{w}) \cdot \mathcal{N}\left(\underline{x}^e;\underline{\mu}_i^e,\mathbf{C}_i^e\right) \cdot f(\underline{w})\,\mathrm{d}\underline{x}^e\,\mathrm{d}\underline{w}\,. \tag{9}$$

Similar to the mean, the covariance matrix is given by

$$\mathbf{C}_i^p = \iint \left(\underline{a}(\underline{x}^e,\underline{u},\underline{w}) - \underline{\mu}_i^p\right)\left(\underline{a}(\underline{x}^e,\underline{u},\underline{w}) - \underline{\mu}_i^p\right)^{\mathrm{T}} \cdot \mathcal{N}\left(\underline{x}^e;\underline{\mu}_i^e,\mathbf{C}_i^e\right) \cdot f(\underline{w})\,\mathrm{d}\underline{x}^e\,\mathrm{d}\underline{w}\,. \tag{10}$$

But still, the integrals in (9) and (10) cannot be solved analytically in general. A famous exception is the case, where $\underline{a}(\cdot,\cdot,\cdot)$ is linear and thus, the Kalman predictor can be applied individually for each component.

1 This weight update is exact only for linear system models, otherwise it is an approximation. For an improved weight update, see for example [14].

3.2 Filter Step

In the filter step, the Gaussian mixture (6) of the predicted state is used in (4), which results in

$$f^e(\underline{x}) = \frac{1}{c} \cdot f\big(\hat{\underline{y}}|\underline{x}\big) \cdot \sum_{i=1}^{L} \omega_i^p \cdot \mathcal{N}\big(\underline{x}; \underline{\mu}_i^p, \mathbf{C}_i^p\big) \,. \tag{11}$$

Due to the nonlinear measurement equation (2), the filter step cannot be solved analytically. For applying individual approximations in (11), the equation is extended with $\mathcal{N}(\hat{\underline{y}}; \underline{\mu}_i^y, \mathbf{C}_i^y)$, which results in

$$\begin{aligned} f^e(\underline{x}) &= \frac{1}{c} \sum_{i=1}^{L} w_i^p \cdot \mathcal{N}\left(\hat{\underline{y}}; \underline{\mu}_i^y, \mathbf{C}_i^y\right) \cdot \frac{f\big(\hat{\underline{y}}|\underline{x}^p\big) \cdot \mathcal{N}\big(\underline{x}^p; \underline{\mu}_i^p, \mathbf{C}_i^p\big)}{\mathcal{N}\left(\hat{\underline{y}}; \underline{\mu}_i^y, \mathbf{C}_i^y\right)} \\ &\approx \sum_{i=1}^{L} w_i^e \cdot \mathcal{N}\big(\underline{x}; \underline{\mu}_i^e, \mathbf{C}_i^e\big) \,, \end{aligned}$$

where the fraction is approximated with a Gaussian distribution $\mathcal{N}\big(\underline{x}; \underline{\mu}_i^e, \mathbf{C}_i^e\big)$ and the weights for the estimated density are given by

$$w_i^e = \frac{w_i^p \cdot \mathcal{N}\left(\hat{\underline{y}}; \underline{\mu}_i^y, \mathbf{C}_i^y\right)}{\sum_{i=1}^{L} w_i^p \cdot \mathcal{N}\left(\hat{\underline{y}}; \underline{\mu}_i^y, \mathbf{C}_i^y\right)} \,.$$

For calculating the estimated mean $\underline{\mu}_i^e$ and covariance $\mathbf{C}_i^e$ for each component i, it is assumed that the state and the measurement are jointly Gaussian. This assumption is typical for Gaussian filters and is only true for linear systems affected with Gaussian noise. Otherwise, it is an approximation. In doing so, the desired moments are

$$\begin{aligned} \underline{\mu}_i^e &= \underline{\mu}_i^p + \mathbf{C}_i^{x,y} \left(\mathbf{C}_i^y\right)^{-1} \left(\hat{\underline{y}} - \underline{\mu}_i^y\right) \,, \\ \mathbf{C}_i^e &= \mathbf{C}_i^p - \mathbf{C}_i^{x,y} \left(\mathbf{C}_i^y\right)^{-1} \left(\mathbf{C}_i^{x,y}\right)^{\mathrm{T}} \,, \end{aligned} \tag{12}$$

which depend on the current measurement value $\hat{\underline{y}}$. To calculate the required parameters $\mathbf{C}_i^{x,y}, \mathbf{C}_i^y$, and $\underline{\mu}_i^y$ in (12), moment calculations similar to the prediction step are applied. The predicted measurement $\underline{\mu}_i^y$, the covariance of the measurement process $\mathbf{C}_i^y$, and the cross-covariance $\mathbf{C}_i^{x,y}$ between state and measurement can then be calculated via

$$\underline{\mu}_i^y = \iint \underline{h}(\underline{x},\underline{v}) \cdot \mathcal{N}\left(\underline{x};\underline{\mu}_i^p,\mathbf{C}_i^p\right) \cdot f(\underline{v})\,\mathrm{d}\underline{x}\,\mathrm{d}\underline{v}\,, \tag{13}$$

$$\mathbf{C}_i^y = \iint \left(\underline{h}(\underline{x},\underline{v}) - \underline{\mu}_i^y\right) \cdot \left(\underline{h}(\underline{x},\underline{v}) - \underline{\mu}_i^y\right)^{\mathrm{T}} \cdot \mathcal{N}\left(\underline{x};\underline{\mu}_i^p,\mathbf{C}_i^p\right) \cdot f(\underline{v})\,\mathrm{d}\underline{x}\,\mathrm{d}\underline{v}\,, \tag{14}$$

$$\mathbf{C}_i^{x,y} = \iint \left(\underline{x} - \underline{\mu}_i^p\right) \cdot \left(\underline{h}(\underline{x},\underline{v}) - \underline{\mu}_i^y\right)^{\mathrm{T}} \cdot \mathcal{N}\left(\underline{x};\underline{\mu}_i^p,\mathbf{C}_i^p\right) \cdot f(\underline{v})\,\mathrm{d}\underline{x}\,\mathrm{d}\underline{v}\,. \tag{15}$$

Unfortunately, these integrals cannot be solved in closed form in general. Thus, approximations in form of linearizations have to be applied, which are described in Section 4.

3.3 Gaussian Mixture Noise

So far, not much attention has been paid to the noise densities $f(\underline{v})$ and $f(\underline{w})$. If the noise is Gaussian, it can be processed jointly with the individual Gaussian components of the state density. One way to deal with arbitrary noise densities is to approximate them by means of a Gaussian mixture. In doing so, the noise can still be processed component-wise and the overall estimation procedure remains similar to the single Gaussian noise case. However, the calculation of the weighting factors differs, since the weighting factors of the noise mixture have to be considered as well. Furthermore, the number of components representing the Gaussian mixture of the state increases exponentially with the time. To bound this growth, Gaussian mixture reduction algorithms such as the one proposed in [7] have to be employed.

4 Types of Linearization

For an approximate solution of the moment integrals in (9), (10) and (13)–(15), GMFs rely on individually linearizing the nonlinear functions $\underline{a}(\cdot,\cdot,\cdot)$ and $\underline{h}(\cdot,\cdot)$ for each component. Thus, the estimation performance of a GMF significantly depends on the quality of these linearizations. In the following, common types of linearization are briefly described. Then, two novel linearization approaches are proposed, namely the analytic stochastic linearization and the semi-analytic stochastic linearization.

For introducing the different linearization approaches, it is sufficient to restrict the focus on an abstract nonlinear transformation

$$\underline{\boldsymbol{y}} = \underline{g}(\underline{\boldsymbol{x}}) \ . \tag{16}$$

Here, the Gaussian random vector $\underline{\boldsymbol{x}} \sim \mathcal{N}\left(\underline{x}; \underline{\mu}^x, \mathbf{C}^x\right)$ is mapped to the random vector $\underline{\boldsymbol{y}}$. The nonlinear transformation $\underline{g}(\cdot)$ can be replaced by $\underline{a}(\cdot,\cdot,\cdot)$ in the prediction step and by $\underline{h}(\cdot,\cdot)$ in the filter step.

4.1 Classical Linearization

Generally, linearization approaches can be separated into explicit and implicit linearization. Classical linearizations that fall into these two classes are as follows.

Explicit Linearization

For an explicit linearization, the nonlinear function $\underline{g}(\cdot)$ is linearized by applying a first-order Taylor-series expansion around the mean $\underline{\mu}^x$ as in the EKF. Thus, the linearized function is given by

$$\underline{g}(\underline{x}) \approx \underline{g}\left(\underline{\mu}^x\right) + \nabla_x \underline{g}\left(\underline{\mu}^x\right) \cdot \left(\underline{x} - \underline{\mu}^x\right) ,$$

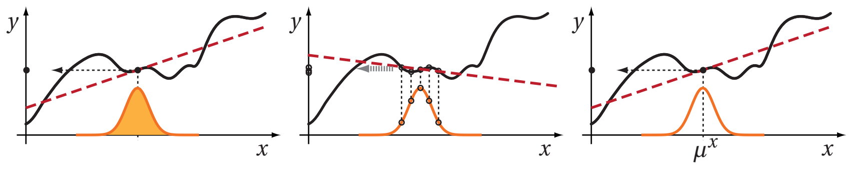
where ∇_x is the gradient with respect to $\underline{x}$. Employing this type of linearization in a GMF corresponds to a bank of EKFs and the resulting GMF is called the Gaussian sum filter proposed in [1].

Implicit Linearization

In case of the implicit linearization, the nonlinear function in (16) remains untouched. Here, the word implicit indicates that if the mean $\underline{\mu}^y$ and covariance $\mathbf{C}^y$ of $\underline{\boldsymbol{y}}$ can be calculated, there exists an equivalent linear transformation

$$\underline{\boldsymbol{y}} = \underline{g}(\underline{\boldsymbol{x}}) \approx \mathbf{A} \cdot \underline{\boldsymbol{x}} + \underline{b} \tag{17}$$

that produces exactly the same mean and covariance of $\underline{\boldsymbol{y}}$ given a Gaussian random vector $\underline{\boldsymbol{x}}$.



(a) Analytic stochastic linearization **(b)** Sample-based linearization **(c)** Linearization via Taylor-series expansion

Figure 1: Illustration of the different linearization approaches: the nonlinear function (black) and its linearized version (red dashed). (a) For the analytic stochastic linearization the entire Gaussian is propagated. (b) Sample-based linearization utilizes a sample representation of the Gaussian. (c) Taylor-series expansion linearizes the nonlinear function around a single point.

A straightforward approach for implicit linearization is to approximate the Gaussian $\underline{\boldsymbol{x}}$ via a sample-based representation

$$\mathcal{N}\left(\underline{x};\underline{\mu}^x,\mathbf{C}^x\right) \approx \sum_{j=1}^{N} w_j \cdot \delta\left(\underline{x}-\underline{\mu}_j\right) , \tag{18}$$

which exactly captures the mean $\underline{\mu}^x$ and covariance $\mathbf{C}^x$. Here, w_j is the weight of the jth sample point located at $\underline{\mu}_j$. In the following, the index j is used for indicating sample points, while the index i is used for components of a Gaussian mixture. The samples can be easily propagated through (16). Calculating the weighted sample mean and sample covariance for the propagated samples then allows approximately solving the moment integrals (9), (10) and (13)–(15). The linear transformation in (17)—if desired—can be obtained via least squares optimization as described in [10].

Several methods exist for calculating the sample representation in (18), e.g., the unscented transform [8] or the deterministic sampling scheme [6]. Combining the unscented transform with a GMF for instance, corresponds to a bank of UKFs and is described in detail in [13].

4.2 Analytic Stochastic Linearization

Special nonlinear functions for $\underline{g}(\cdot)$ facilitate to solve the moment integrals analytically and exactly, e.g., polynomials, trigonometric functions, and their combinations. In this case, the results of the moment integrals can be derived in an analytic form, as it is shown in the simulation example in Section 5. Thus, linearization is performed implicitly under the consideration of the entire Gaussian density of $\underline{\boldsymbol{x}}$. This is in contrast to the previously described sample-based linearization, which merely propagates a sample-based approximation of the Gaussian. Even more extreme is the explicit linearization used in the EKF, where linearization is only performed on the basis of the single point, that is the mean of the Gaussian. These differences are depicted in Figure 1.

In the following, the combination of analytic stochastic linearization with a GMF is named analytic Gaussian mixture filter (A-GMF).

4.3 Semi-Analytic Stochastic Linearization

In order to extend the principle of analytic stochastic linearization to a wider class of nonlinear functions, the semi-analytic stochastic linearization approach is proposed. Here, sampled-based linearization is combined with the analytic stochastic linearization such that only some dimensions of the random vector $\underline{\boldsymbol{x}}$ are discretized by means of a sample representation. Thus, only some parts of the nonlinear transformation (16) have to be evaluated approximately for moment calculation.

For this purpose, the nonlinear equation (16) is rearranged according to

$$\underline{\boldsymbol{y}} = \underline{g}\left(\underline{\boldsymbol{x}}^a, \underline{\boldsymbol{x}}^b\right) , \tag{19}$$

where the Gaussian random vector $\underline{\boldsymbol{x}}^{\mathrm{T}} = \left[\left(\underline{\boldsymbol{x}}^a\right)^{\mathrm{T}}, \left(\underline{\boldsymbol{x}}^b\right)^{\mathrm{T}}\right]$ consists of the substates $\underline{\boldsymbol{x}}^a$, $\underline{\boldsymbol{x}}^b$ with mean and covariance

$$\underline{\mu}^x = \begin{bmatrix} \underline{\mu}^a \\ \underline{\mu}^b \end{bmatrix} , \quad \mathbf{C}^x = \begin{bmatrix} \mathbf{C}^a & \mathbf{C}^{a,b} \\ \mathbf{C}^{b,a} & \mathbf{C}^b \end{bmatrix} .$$

As mentioned above, there exists no closed-form expression for the desired moments in general. However, the decomposition into $\underline{\boldsymbol{x}}^a$ and $\underline{\boldsymbol{x}}^b$ is chosen in such a way that the moment integrals can be calculated in closed form for any given fixed value of $\underline{\boldsymbol{x}}^b$. Hence, $\underline{g}(\cdot,\cdot)$ is denoted to be conditionally integrable if such a decomposition exists. For determining a sample-based representation of $\underline{\boldsymbol{x}}^b$, the sampling techniques mentioned in Section 4.1 are applied.

The analytic stochastic linearization and sampled-based linearization are extreme cases of the semi-analytic stochastic linearization: if $\underline{\boldsymbol{x}}^b$ has no entries, the semi-analytic stochastic linearization becomes an analytic stochastic linearization and if $\underline{\boldsymbol{x}}^a$ has no entries, the semi-analytic stochastic linearization degenerates into a sample-based linearization.

General Solution

For the general transformation in (19), the desired moments of $\underline{\boldsymbol{y}}$ can be calculated as follows. At first, the joint density $f(\underline{x}, \underline{y})$ is separated by employing Bayes' rule

$$f(\underline{x}, \underline{y}) = \underbrace{\delta\left(\underline{y} - \underline{g}(\underline{x}^a, \underline{x}^b)\right)}_{= f\left(\underline{y}|\underline{x}\right)} \cdot \underbrace{f(\underline{x}^a|\underline{x}^b) \cdot f(\underline{x}^b)}_{= f(\underline{x})} \, .$$

The conditional density $f(\underline{x}^a|\underline{x}^b) = \mathcal{N}(\underline{x}^a; \underline{\mu}^{a|b}, \mathbf{C}^{a|b})$ is (conditionally) Gaussian with mean and covariance

$$\begin{aligned} \underline{\mu}^{a|b} &= \underline{\mu}^a + \mathbf{C}^{a,b} \cdot \left(\mathbf{C}^b\right)^{-1} \cdot \left(\underline{x}^b - \underline{\mu}^b\right) \, , \\ \mathbf{C}^{a|b} &= \mathbf{C}^a - \mathbf{C}^{a,b} \cdot \left(\mathbf{C}^b\right)^{-1} \cdot \mathbf{C}^{b,a} \, . \end{aligned} \tag{20}$$

To determine the mean $\underline{\mu}^y$, the Gaussian density $f(\underline{x}^b)$ of the substate $\underline{\boldsymbol{x}}^b$ is represented by means of a sample density as in (18), which allows a sample-based linearization. For integrating over $\underline{x}^b$, the sifting property of the Dirac delta distribution is exploited. This gives rise to

$$\underline{\mu}^y \approx \sum_{j=1}^{N} w_j \cdot \underline{\mu}_j^y \quad \text{with} \quad \underline{\mu}_j^y = \int \underline{g}\left(\underline{x}^a, \underline{\mu}_j^b\right) \cdot f\left(\underline{x}^a|\underline{\mu}_j^b\right) \mathrm{d}\underline{x}^a \tag{21}$$

for the mean of $\underline{\boldsymbol{y}}$ and analogously, the covariance of $\underline{\boldsymbol{y}}$ is approximated via

$$\begin{aligned} \mathbf{C}^y &\approx \sum_{j=1}^{N} w_j \cdot \left(\mathbf{C}_j^y - \underline{\mu}_j^y (\underline{\mu}^y)^{\mathrm{T}} - \underline{\mu}^y (\underline{\mu}_j^y)^{\mathrm{T}} + \underline{\mu}^y (\underline{\mu}^y)^{\mathrm{T}}\right) \, , \\ \mathbf{C}_j^y &= \int \underline{g}\left(\underline{x}^a, \underline{\mu}_j^b\right) \cdot \underline{g}\left(\underline{x}^a, \underline{\mu}_j^b\right)^{\mathrm{T}} \cdot f\left(\underline{x}^a|\underline{\mu}_j^b\right) \mathrm{d}\underline{x}^a \, . \end{aligned} \tag{22}$$

It is worth mentioning that the integrals in (21) and (22) can be evaluated analytically as the function $\underline{g}(\cdot,\cdot)$ is chosen to be conditionally integrable. Solving these integrals is an off-line task and the solution is represented in parametric form for an efficient on-line evaluation.

Gaussian Estimation

By putting it all together, prediction and filtering for the ith Gaussian component of the Gaussian mixture can now be derived.

In the prediction step, the predicted mean $\underline{\mu}_i^p$ (9) and covariance $\mathbf{C}_i^p$ (10) of $f_i^p(\underline{x}^p)$ have to be calculated. For this purpose, the system function (1) can be directly mapped to the nonlinear transformation (19) according to

$$\underline{\boldsymbol{x}}^p = \underline{a}\left(\underline{\boldsymbol{x}}^e, \underline{u}, \underline{\boldsymbol{w}}\right) = \underline{g}\left(\underline{\boldsymbol{x}}^a, \underline{\boldsymbol{x}}^b\right) .$$

Here, the system input $\underline{u}$ becomes a part of the function $\underline{g}(\cdot,\cdot)$ and the substates $\underline{\boldsymbol{x}}^a$ and $\underline{\boldsymbol{x}}^b$ are augmented with the noise variables $\underline{\boldsymbol{w}}^a$ and $\underline{\boldsymbol{w}}^b$, where $\underline{\boldsymbol{w}}^{\mathrm{T}} = \left[(\underline{\boldsymbol{w}}^a)^{\mathrm{T}}, (\underline{\boldsymbol{w}}^b)^{\mathrm{T}}\right]$, in order to consider additive and/or multiplicative noise. For calculating the mean $\underline{\mu}_i^p$ and covariance $\mathbf{C}_i^p$, (21) and (22) are used, respectively.

In the filter step, the measurement equation (2) is mapped to the nonlinear transformation (19) according to

$$\underline{\boldsymbol{y}} = \underline{h}(\underline{\boldsymbol{x}}, \underline{\boldsymbol{v}}) = \underline{g}\left(\underline{\boldsymbol{x}}^a, \underline{\boldsymbol{x}}^b\right) ,$$

where the measurement noise $\underline{\boldsymbol{v}}$ is spread across the substates $\underline{\boldsymbol{x}}^a, \underline{\boldsymbol{x}}^b$. It is worth mentioning that the decomposition of $\underline{\boldsymbol{x}}$ into the substates for the filter step is independent of the decomposition of the prediction step.

For determining the mean $\underline{\mu}_i^e$ and covariance $\mathbf{C}_i^e$ in (12) for the ith Gaussian component, the moments $\underline{\mu}_i^y$, $\mathbf{C}_i^y$ and $\mathbf{C}_i^{x,y}$ in (13)–(15) are required. According to (18), the density of the substate $\underline{\boldsymbol{x}}^b$ is approximated by the sample density

$$\mathcal{N}\left(\underline{x}^b; \underline{\mu}_i^b, \mathbf{C}_i^b\right) \approx \sum_{j=1}^{N} w_{ij} \cdot \delta\left(\underline{x}^b - \underline{\mu}_{ij}^b\right) .$$

By means of this sample representation, $\underline{\mu}_i^y$ and $\mathbf{C}_i^y$ can be calculated as in (21) and (22), respectively. The cross-covariance $\mathbf{C}_i^{x,y}$ needs further derivations. The cross-covariance $\mathbf{C}_i^{x,y} = \left[\mathbf{C}_i^{a,y}, \mathbf{C}_i^{b,y}\right]^{\mathrm{T}}$ consists of

$$\mathbf{C}_i^{a,y} = \sum_{j=1}^{N} w_{ij} \cdot \left(\mathbf{C}_{ij}^{a,y} - \underline{\mu}_{ij}^{a|b} \left(\underline{\mu}_i^y\right)^{\mathrm{T}} + \underline{\mu}_i^a \left(\underline{\mu}_i^y - \underline{\mu}_{ij}^y\right)^{\mathrm{T}}\right) ,$$

$$\mathbf{C}_i^{b,y} = \sum_{j=1}^{N} w_{ij} \cdot \left(\underline{\mu}_{ij}^b - \underline{\mu}_i^b\right) \cdot \left(\underline{\mu}_{ij}^y - \underline{\mu}_i^y\right)^{\mathrm{T}} ,$$

with

$$\mathbf{C}_{ij}^{a,y} = \int \underline{x}^a \cdot \underline{g}\left(\underline{x}^a, \underline{\mu}_{ij}^b\right)^{\mathrm{T}} \cdot f\left(\underline{x}^a \middle| \underline{\mu}_{ij}^b\right) \mathrm{d}\underline{x}^a ,$$

where $\underline{\mu}_{ij}^{a|b}$ is calculated according to (20) with $\underline{x}^b$ replaced by $\underline{\mu}_{ij}^b$ and $\underline{\mu}_{ij}^y$ results from solving the integral in (21).

The combination of semi-analytic stochastic linearization with a GMF is named semi-analytic Gaussian mixture filter (SA-GMF) in the following.

5 Simulation Results

In the simulations, a localization scenario is considered. A robot with tricycle kinematics measures the distance to one out of four landmarks per time step. The landmark considered for measurement is selected randomly with equal probability. The proposed A-GMF and SA-GMF are compared with the EKF-GMF (a.k.a. Gaussian sum filter, [1]) and the UKF-GMF [13] for different numbers of components and measurement noise levels.

5.1 System and Measurement Model

The nonlinear kinematics model of the robot is given by

$$\begin{aligned} \boldsymbol{p}_{k+1}^x &= \boldsymbol{p}_k^x + \left(u_k^v + \underline{\boldsymbol{w}}_k^v\right) \cdot \cos\left(\boldsymbol{\phi}_k + u_k^\alpha\right) , \\ \boldsymbol{p}_{k+1}^y &= \boldsymbol{p}_k^y + \left(u_k^v + \underline{\boldsymbol{w}}_k^v\right) \cdot \sin\left(\boldsymbol{\phi}_k + u_k^\alpha\right) , \\ \boldsymbol{\phi}_{k+1} &= \boldsymbol{\phi}_k + \left(u_k^\alpha + \underline{\boldsymbol{w}}_k^\alpha\right) , \end{aligned} \tag{23}$$

with state $\underline{\boldsymbol{x}}_k = \left[\boldsymbol{p}_k^x \;\; \boldsymbol{p}_k^y \;\; \boldsymbol{\phi}_k\right]^{\mathrm{T}}$, where $\boldsymbol{p}_k^x$ and $\boldsymbol{p}_k^y$ describe the Cartesian position of the robot and $\boldsymbol{\phi}_k$ its orientation. The known control inputs are the velocity u_k^v and the turning angle u_k^α. $\underline{\boldsymbol{w}}_k^v$ and $\underline{\boldsymbol{w}}_k^\alpha$ are noise processes affecting the corresponding control inputs. They are assumed to be zero-mean Gaussian with variances Q^v and Q^α, respectively.

The measured range $\boldsymbol{r}_k$ is given by the nonlinear measurement model

$$\boldsymbol{r}_k = \sqrt{\left(\boldsymbol{p}_k^x - L^x + \boldsymbol{v}_k^x\right)^2 + \left(\boldsymbol{p}_k^y - L^y + \boldsymbol{v}_k^y\right)^2} \,,$$

where $\underline{L} = \left[L^x \; L^y\right]^{\mathrm{T}}$ is the position of the landmark, and $\underline{\boldsymbol{v}}_k = \left[\boldsymbol{v}_k^x \; \boldsymbol{v}_k^y\right]^{\mathrm{T}}$ is the measurement noise. The noise is assumed to be zero-mean Gaussian with covariance $\mathbf{C}^v$.

5.2 Estimator: A-GMF

The prediction step can be solved analytically as the model (23) consists of linear, bilinear, and polynomial functions. The predicted mean of the ith Gaussian component is

$$\begin{aligned}
\underline{\mu}_i^p(1) &= \underline{\mu}_i^e(1) + u^v \cdot \mathrm{e} \cdot \cos(\beta) \,,\\
\underline{\mu}_i^p(2) &= \underline{\mu}_i^e(2) + u^v \cdot \mathrm{e} \cdot \sin(\beta) \,,\\
\underline{\mu}_i^p(3) &= \underline{\mu}_i^e(3) + u^\alpha \,,
\end{aligned}$$

where $\underline{\mu}_i^p(j)$ is the jth element of the vector $\underline{\mu}_i^p$ and $\mathbf{C}_i^e(m,n)$ is the element of the matrix $\mathbf{C}_i^e$ at row m and column n. The variables β and e are given by

$$\beta = u^\alpha + \underline{\mu}_i^e(3) \quad \text{and} \quad \mathrm{e} = \exp\left(-\tfrac{1}{2} \cdot \mathbf{C}_i^e(3,3)\right) \,,$$

respectively. The analytic expressions of the predicted covariances can be found in Appendix A.

For an analytic moment calculation in the filter step, the measurement model needs to be squared, where the resulting new measurement is denoted by $\boldsymbol{y}_k \triangleq (\boldsymbol{r}_k)^2$. Based on the new squared measurement model, which now is polynominal, the required quantities in (13)-(15) are

$$\mu_i^y = \underline{A}^{\mathrm{T}} \cdot \underline{A} + \mathrm{trace}(\mathbf{T}) \,, \quad \mathbf{C}_i^{x,y} = -2 \cdot \mathbf{C}_i^p \cdot \mathbf{P}^{\mathrm{T}} \cdot \underline{A} \,,$$

$$C_i^y = \underline{1}_2^{\mathrm{T}} \cdot (4 \cdot (\underline{A} \cdot \underline{A}^{\mathrm{T}}) \circ \mathbf{T} + 2\mathbf{T} \circ \mathbf{T}) \cdot \underline{1}_2 ,$$

where $\circ$ is the element-wise product, $\underline{1}_2 = [1\ 1]^{\mathrm{T}}$, and

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} , \quad \mathbf{T} = \mathbf{C}^v + \mathbf{P} \cdot \mathbf{C}_i^p \cdot \mathbf{P}^{\mathrm{T}} , \quad \underline{A} = \underline{L} - \underline{\mu}_i^p .$$

The measurement value $\hat{y}$ is replaced by $(\hat{r})^2$.

5.3 Estimator: SA-GMF

Even if the prediction step can be performed in closed form, the system model is written in conditionally linear form according to

$$\underline{\boldsymbol{x}}_{k+1} = \begin{bmatrix} u_k^v \cdot \cos\big(\boldsymbol{\phi}_k + u_k^\alpha\big) \\ u_k^v \cdot \sin\big(\boldsymbol{\phi}_k + u_k^\alpha\big) \\ \boldsymbol{\phi}_k + u_k^\alpha \end{bmatrix} + \begin{bmatrix} 1 & 0 & \cos\big(\boldsymbol{\phi}_k + u_k^\alpha\big) & 0 \\ 0 & 1 & \sin\big(\boldsymbol{\phi}_k + u_k^\alpha\big) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \underline{\boldsymbol{x}}_k^a ,$$

in order to demonstrate to performace of the SA-GMF. Here, $\underline{\boldsymbol{x}}^a = \left[\boldsymbol{p}^x\ \boldsymbol{p}^y\ \underline{\boldsymbol{w}}^v\ \underline{\boldsymbol{w}}^\alpha\right]^{\mathrm{T}}$ and $\underline{\boldsymbol{x}}^b = \boldsymbol{\phi}$. By approximating the orientation $\boldsymbol{\phi}$ by means of the sample-based representation, the nonlinear system model becomes a conditionally linear one. Thus, given a sample point $\underline{\mu}_j^b$, the prediction step with respect to $\underline{\boldsymbol{x}}^a$ for each Gaussian component can be solved via the Kalman predictor. The filter step is performed as in the A-GMF.

5.4 Setup

The initial position of the robot at time step $k = 0$ is $\underline{x}_0 = [5\ 3\ 0.2]^{\mathrm{T}}$. Furthermore, the known control inputs are constant and selected as $u^v = 0.1$ and $u^\alpha = 0.1$. The variances of the noise processes are $Q^v = 0.1$ and $Q^\alpha = 0.01$.

The measurement noise is isotropic, i.e., the covariance is $\mathbf{C}^v = (\sigma^v)^2 \cdot \mathbf{I}_2$ with $\mathbf{I}_2$ being the 2×2 identity matrix. For the variance $(\sigma^v)^2$, the three noise levels 0.5, 1, and 2 are considered. The positions $\underline{L}_i$, $i = 1, \ldots, 4$ of the four landmarks are

$$\begin{bmatrix} \underline{L}_1 & \underline{L}_3 & \underline{L}_3 & \underline{L}_4 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 5 & 10 \\ 0 & 2 & 5 & 10 \end{bmatrix} .$$

Table 1: Average rmse and standard deviation for the different estimators at different noise levels and numbers of components.

Noise 0.5	64	8	1
A-GMF	**2.02 ± 1.34**	**3.17 ± 1.98**	**3.76 ± 2.01**
SA-GMF	2.03 ± 1.34	**3.16 ± 1.99**	3.76 ± 2.09
UKF-GMF	2.41 ± 1.79	4.08 ± 3.02	4.31 ± 3.58
EKF-GMF	2.64 ± 1.86	4.08 ± 4.99	6.40 ± 9.64

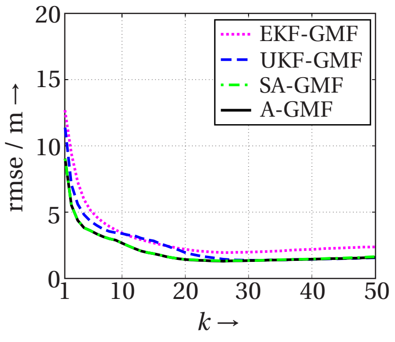
Noise 1.0	64	8	1
A-GMF	**2.05 ± 1.36**	3.16 ± 1.97	**3.70 ± 2.07**
SA-GMF	**2.06 ± 1.35**	**3.15 ± 1.97**	**3.70 ± 2.07**
UKF-GMF	2.44 ± 1.80	4.04 ± 3.05	4.25 ± 3.61
EKF-GMF	2.95 ± 1.73	4.40 ± 4.43	6.72 ± 8.71

Noise 2.0	64	8	1
A-GMF	**2.16 ± 1.35**	**3.22 ± 2.02**	**3.78 ± 2.16**
SA-GMF	2.16 ± 1.36	**3.22 ± 2.02**	**3.78 ± 2.16**
UKF-GMF	2.61 ± 1.77	4.11 ± 3.08	4.35 ± 3.65
EKF-GMF	3.42 ± 1.55	4.95 ± 4.14	7.21 ± 7.60

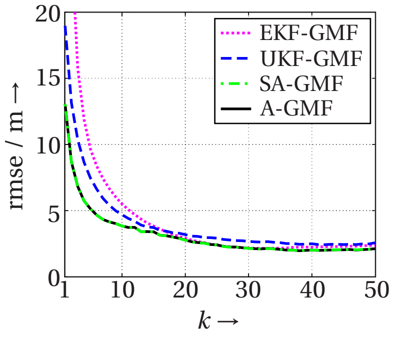
The GMFs are initialized with different numbers of components, namely 1, 8, and 64 components. This initialization is performed via the splitting procedure proposed in [4], where the Gaussian used for splitting has the covariance matrix $\mathbf{C} = \text{diag}([1000\ 1000\ 30])$ and a mean sampled from the Gaussian $\mathcal{N}(\underline{x}; \underline{x}_0, \mathbf{C})$. For each combination of noise level and number of components, 1000 simulation runs are performed, where each run consists of 50 time steps.

5.5 Results

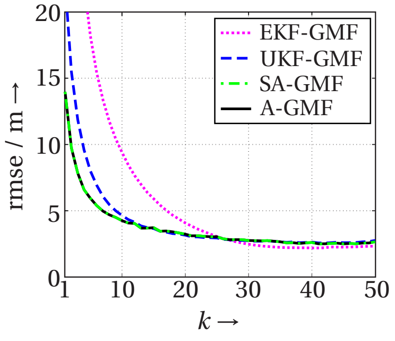
In Table 1, the average root mean square error (rmse) with respect to the robot's position and the standard deviation of the average rmse of all estimators for all noise levels and all numbers of components are listed. The A-GMF and the SA-GMF outperform the UKF-GMF and the EKF-GMF under all conditions. The rmse for the noise level $(\sigma^v)^2 = 1$ is shown in Figure 2. A-GMF and SA-GMF converge significantly faster compared to other two estimators and provide the lowest estimation errors.



(a) 64 components



(b) 8 components



(c) 1 component

Figure 2: The rmse and its standard deviation over the 1000 trajectories at noise level 1 for different numbers of Gaussian components.

In this simulation, A-GMF and SA-GMF perform almost identical as both use the same filter step and a sample-based representation in case of the SA-GMF is merely necessary for one dimension.

The EKF-GMF provides the worst estimation results in this scenario. This can be explained by the fact that the underlying linearization does not consider any uncertainty information and thus, the linearization error is neglected. Thanks to the sample-based linearization of the UKF-GMF, uncertainty information can be incorporated, which leads to a superior linearization and estimation compared to the EKF-GMF. However, only a finite number of sample points is used for moment calculation. In the A-GMF instead, the linearization is performed implicitly under the consideration of the entire Gaussian density. This further improves estimation performance.

Furthermore, the computational complexity of the A-GMF and SA-GMF is lower compared to the UKF-GMF. This can be explained by the necessity of calculating matrix square roots for determining a sample-based representation of a Gaussian. This operation has cubic complexity and has to be applied in the UKF-GMF on the whole covariance matrix of each Gaussian component. It is not required for the A-GMF and in case of the SA-GMF, the covariance matrix is of reduced dimension as some dimensions of the state space are processed analytically. A further reason for the higher computational load of the UKF-GMF is the high number of on-line evaluations of the functions (1) and (2). In case of the A-GMF, all function evaluations are performed off-line, while for the SA-GMF the number of function evaluations is reduced due to the small dimension of the substate $\underline{\boldsymbol{x}}^b$.

6 Conclusions

This paper introduces two novel Gaussian mixture filters, which combine analytic stochastic linearization techniques with a Gaussian mixture density representation. The purely analytic stochastic linearization approach is designed for nonlinear filtering problems, where the integrals for calculating the mean and covariance can be solved in closed form. Semi-analytic linearization extends this approach to a wider class of filtering problems, where only some parts of the system model need to be analytically integrable. Thanks to the combination of these linearization approaches with a Gaussian mixture representation of the state density, the estimation performance is improved compared to Gaussian filters, especially in problems with severe nonlinearities. In the simulation it is shown that the proposed filter outperforms Gaussian mixture filters employing standard linearization techniques concerning estimation error and convergence.

A Analytic Expressions for A-GMF

The analytic expressions of the elements of the predicted covariance matrix of the A-GMF are

$$
\begin{aligned}
\mathbf{C}_i^p(1{,}1) &= \tfrac{1}{2}(u^v)^2 + \mathbf{C}_i^e(1{,}1) - \left(u^v \mathrm{e} \cos(\beta)\right)^2 \\
&\quad + \tfrac{1}{2}\left(u^v\right)^2 \mathrm{e}^4 \cos(2\beta) - 2u^v \mathrm{e} \sin(\beta) \cdot \mathbf{C}_i^e(1{,}3) \\
&\quad + Q^v \tfrac{1}{2}\left(1 + \mathrm{e}^4 \cos(2\beta)\right) , \\
\mathbf{C}_i^p(2{,}2) &= \tfrac{1}{2}(u^v)^2 + \mathbf{C}_i^e(2{,}2) - \left(u^v \mathrm{e} \sin(\beta)\right)^2 \\
&\quad - \tfrac{1}{2}\left(u^v\right)^2 \mathrm{e}^4 \cos(2\beta) + 2u^v \mathrm{e} \cos(\beta) \cdot \mathbf{C}_i^e(2{,}3) \\
&\quad + Q^v \tfrac{1}{2}\left(1 - \mathrm{e}^4 \cos(2\beta)\right) , \\
\mathbf{C}_i^p(3{,}3) &= \mathbf{C}_i^e(3{,}3) + Q^\alpha \\
\mathbf{C}_i^p(1{,}2) &= \mathbf{C}_i^e(1{,}2) - u^v \cdot \mathrm{e} \cdot \mathbf{C}_i^e(1{,}3) \cos(\beta) - u^v \cdot \mathrm{e} \cdot \mathbf{C}_i^e(2{,}3) \cdot \sin(\beta) \\
&\quad + (u^v \cdot \mathrm{e})^2 \cos(\beta) \sin(\beta) (\mathrm{e}^2 - 1) + Q^v \tfrac{\mathrm{e}^4}{2} \sin(2\beta) , \\
\mathbf{C}_i^p(1{,}3) &= \mathbf{C}_i^e(1{,}3) - u^v \cdot \mathbf{C}_i^e(3{,}3) \cdot \mathrm{e} \cdot \sin(\beta) , \\
\mathbf{C}_i^p(2{,}3) &= \mathbf{C}_i^e(2{,}3) + u^v \cdot \mathbf{C}_i^e(3{,}3) \cdot \mathrm{e} \cdot \cos(\beta) , \\
\mathbf{C}_i^p(2{,}1) &= \mathbf{C}_i^p(1{,}2) , \quad \mathbf{C}_i^p(3{,}1) = \mathbf{C}_i^p(1{,}3) , \quad \mathbf{C}_i^p(3{,}2) = \mathbf{C}_i^p(2{,}3) .
\end{aligned}
$$

References

[1] Daniel L. Alspach and Harold W. Sorenson. Nonlinear Bayesian Estimation using Gaussian Sum Approximation. *IEEE Transactions on Automatic Control*, 17(4):439–448, August 1972.

[2] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[3] Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck. Optimal Stochastic Linearization for Range-based Localization. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010.

[4] Marco F. Huber, Tim Bailey, Hugh Durrant-Whyte, and Uwe D. Hanebeck. On Entropy Approximation for Gaussian Mixture Random Vectors. In *Proceedings of the 2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 181–188, Seoul, Republic of Korea, August 2008.

[5] Marco F. Huber, Frederik Beutler, and Uwe D. Hanebeck. Semi-Analytic Gaussian Assumed Density Filter. In *Proceedings of the 2011 American Control Conference (ACC)*, San Francisco, California, June 2011.

[6] Marco F. Huber and Uwe D. Hanebeck. Gaussian Filter based on Deterministic Sampling for High Quality Nonlinear Estimation. In *Proceedings of the 17th IFAC World Congress*, Seoul, Republic of Korea, July 2008.

[7] Marco F. Huber and Uwe D. Hanebeck. Progressive Gaussian Mixture Reduction. In *Proceedings of the 11th International Conference on Information Fusion (Fusion)*, Cologne, Germany, July 2008.

[8] Simon Julier, Jeffrey Uhlmann, and Hugh F. Durrant-Whyte. A New Method for the Nonlinear Trannsformation of Means and Covariances in Filters and Estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, 2000.

[9] Ngai M. Kwok, Gamini Dissanayake, and Quang P. Ha. Bearing-only SLAM Using a SPRT Based Gaussian Sum Filter. In *Proceedings of the 2005 IEEE*

International Conference on Robotics and Automation (ICRA), pages 1109–1114, Barcelona, Spain, April 2005.

[10] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter. *Nonlinear Kalman Filtering for Force-Controlled Robot Tasks.* Springer Berlin, 2005.

[11] Thomas Lemaire, Cyrille Berger, Il-Kyun Jung, and Simon Lacroix. Vision-Based SLAM: Stereo and Monocular Approaches. *International Journal of Computer Vision*, 74(3):343–364, 2007.

[12] Vladimir Maz'ya and Gunther Schmidt. On approximate approximations using gaussian kernels. *IMA J. Numer. Anal.*, 16:13–29, 1996.

[13] Miroslav Simandl and Jindrich Duník. Sigma Point Gaussian Sum Filter Design using Square Root Unscented Filters. In *Proceedings of the 16th IFAC World Congress*, pages 1000–1005, Prague, Czech Republic, 2005.

[14] Gabriel Terejanu, Puneet Singla, Tarunraj Singh, and Peter D. Scott. A Novel Gaussian Sum Filter Method for Accurate Solution to the Nonlinear Filtering Problem. In *Proceedings of the 11th International Conference on Information Fusion (Fusion)*, Cologne, Germany, July 2008.

Paper F

Adaptive Gaussian Mixture Filter Based on Statistical Linearization

Authors: Marco F. Huber

Edited version of the paper:

M. F. Huber. Adaptive Gaussian Mixture Filter Based on Statistical Linearization. In *Proceedings of the 14th International Conference on Information Fusion (Fusion)*, Chicago, Illinois, July 2011.

Adaptive Gaussian Mixture Filter Based on Statistical Linearization

Marco F. Huber

Variable Image Acquisition and Processing Research Group
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB
Karlsruhe, Germany
`marco.huber@ieee.org`

Abstract

Gaussian mixtures are a common density representation in nonlinear, non-Gaussian Bayesian state estimation. Selecting an appropriate number of Gaussian components, however, is difficult as one has to trade of computational complexity against estimation accuracy. In this paper, an adaptive Gaussian mixture filter based on statistical linearization is proposed. Depending on the nonlinearity of the considered estimation problem, this filter dynamically increases the number of components via splitting. For this purpose, a measure is introduced that allows for quantifying the locally induced linearization error at each Gaussian mixture component. The deviation between the nonlinear and the linearized state space model is evaluated for determining the splitting direction. The proposed approach is not restricted to a specific statistical linearization method. Simulations show the superior estimation performance compared to related approaches and common filtering algorithms.

1 Introduction

Bayesian state estimation for nonlinear systems requires an efficient approximation for practical applications as closed-form solutions are not available in general. A common approximation technique is the discretization of the state space as done in grid filters or particle filters [3]. Theoretically, these techniques facilitate to approach the true statistics of the state with arbitrary accuracy. But they are only applicable to low-dimensional problems since their computational complexity increases exponentially with the dimension of the state space.

A famous exception that exhibits an analytic solution is the linear Gaussian case. Here, the famous Kalman filter provides optimal results in an efficient manner [12]. So-called Gaussian filters try to adapted the Kalman filter equations to nonlinear problems by assuming that the density function of the state can be represented by a Gaussian density. The extended Kalman filter [22] applies first-order Taylor series expansion for linearization. The unscented Kalman filter [11, 24] or the Gaussian estimator [8] offer higher order accuracy by employing statistical linearization. But in general a single Gaussian density is typically not a sufficient representation for the true density function, which may be skew or multimodal. Thanks to their universal approximator property, Gaussian mixtures [15] are a much better approach for approximating complex density functions. Examples for Gaussian mixture filters applied to nonlinear estimation are in [1, 21].

The estimation accuracy of Gaussian mixture filters significantly depend on the number of Gaussian components used. This number is typically defined by the user. In this paper, a novel Gaussian mixture filter is proposed, which adapts the number of components dynamically and on-line. The nonlinear system and measurement models are linearized locally by means of statistical linearization at each component of the Gaussian mixture. The induced linearization error is quantified by means of the linearization error covariance matrix. Based on this error, a novel moment-preserving splitting procedure is proposed for introducing new mixture components. The component causing the highest linearization error is selected, while splitting is performed in direction of the strongest nonlinearity, i.e., the strongest deviation between the nonlinear model and its linearized version. Both linearization and splitting are independent of the used statistical linearization method, which makes the proposed filter versatilely applicable.

The paper is structured as follows: The Bayesian state estimation problem is formulated in the next section. In Section 3, a brief introduction in statistical linearization is given. The novel splitting scheme is derived in Section 4. Based on this, Section 5 describes the complete adaptive Gaussian mixture filter with all major components. Numerical evaluation by means of simulations is part of Section 6. The paper closes with concluding remarks.

2 Problem Formulation

In this paper, discrete-time nonlinear dynamic systems

$$\underline{\boldsymbol{x}}_{k+1} = \underline{a}_k\left(\underline{\boldsymbol{x}}_k, \underline{u}_k, \underline{\boldsymbol{w}}_k\right) , \tag{1}$$

$$\underline{\boldsymbol{z}}_k = \underline{h}_k\left(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{v}}_k\right) \tag{2}$$

are considered. Here, (1) is the dynamics model with the known time-variant nonlinear system function $\underline{a}_k(\cdot)$, which propagates the system state $\underline{\boldsymbol{x}}_k \in \mathbb{R}^{n_x}$ at time step k to time step $k+1$, given the current system input $\underline{u}_k \in \mathbb{R}^{n_u}$ and the process noise $\underline{\boldsymbol{w}}_k \in \mathbb{R}^{n_w}$. The measurement model is given by (2), where $\underline{h}_k(\cdot)$ is the known time-variant nonlinear measurement function, $\underline{\boldsymbol{z}}_k \in \mathbb{R}^{n_z}$ is the measurement vector, and $\underline{\boldsymbol{v}}_k \in \mathbb{R}^{n_v}$ is the measurement noise. Note that an actual measurement value $\underline{z}_k$ is a realization of the random vector $\underline{\boldsymbol{z}}_k$ in (2).

Both noise processes $\underline{\boldsymbol{w}}_k$ and $\underline{\boldsymbol{v}}_k$ are assumed to be independent and white. The probability density functions of $\underline{\boldsymbol{w}}_k$ and $\underline{\boldsymbol{v}}_k$ are denoted by $f_k^w\left(\underline{w}_k\right)$ and $f_k^v\left(\underline{v}_k\right)$, respectively. It is assumed that these density functions are *Gaussian mixtures*

$$f_k^w\left(\underline{w}_k\right) = \sum_{i=1}^{L_k^w} \omega_{k,i}^w \cdot \mathcal{N}\left(\underline{w}_k; \underline{\hat{w}}_{k,i}, \mathbf{C}_{k,i}^w\right) , \tag{3}$$

$$f_k^v\left(\underline{v}_k\right) = \sum_{i=1}^{L_k^v} \omega_{k,i}^v \cdot \mathcal{N}\left(\underline{v}_k; \underline{\hat{v}}_{k,i}, \mathbf{C}_{k,i}^v\right) , \tag{4}$$

where L_k^w, L_k^v are the numbers of mixture components, $\omega_{k,i}^w$, $\omega_{k,i}^v$ are non-negative weights that sum up to one, and $\mathcal{N}\left(\underline{w}; \underline{\hat{w}}, \mathbf{C}^w\right)$ is a Gaussian density with mean vector $\underline{\hat{w}}$ and covariance matrix $\mathbf{C}^w$. The initial density function $f_0^x(\underline{x}_0)$ of the system state at time step $k=0$ is also assumed to be given as a Gaussian mixture.

Estimating the system state from noisy measurements is done according to the Bayesian framework. Here, two steps are performed alternately, namely the prediction step and the filtering step. In the prediction step, the density $f_k^e\left(\underline{x}_k\right) := f_k^x\left(\underline{x}_k | \underline{u}_{0:k}, \underline{z}_{0:k}\right)$ of the previous filtering step is propagated to the next time step according to

$$\begin{aligned} f_{k+1}^p\left(\underline{x}_{k+1}\right) &:= f_{k+1}^x\left(\underline{x}_{k+1} | \underline{u}_{0:k}, \underline{z}_{0:k}\right) \\ &= \int \underbrace{f\left(\underline{x}_{k+1} | \underline{x}_k, \underline{u}_k, \underline{w}_k\right)}_{\delta\left(\underline{x}_{k+1} - \underline{a}_k(\underline{x}_k, \underline{u}_k, \underline{w}_k)\right)} \cdot f_k^e\left(\underline{x}_k\right) \cdot f_k^w\left(\underline{w}_k\right) \mathrm{d}\underline{x}_k \, \mathrm{d}\underline{w}_k , \end{aligned} \tag{5}$$

where $\underline{z}_{0:k} = \left(\underline{z}_0, \underline{z}_1, \ldots, \underline{z}_k\right)$ denotes the measurements up to and including time step k, $f\left(\underline{x}_{k+1} | \underline{x}_k, \underline{u}_k, \underline{w}_k\right)$ is the transition density depending on the dynamics model (1), and $\delta(\cdot)$ is the Dirac delta distribution.

The filtering step determines the *posterior density* $f_k^e\left(\underline{x}_k\right)$ of the system state $\underline{x}_k$ based on all acquired measurement values according to *Bayes' law*

$$f_k^e\left(\underline{x}_k\right) = c_k \cdot f\left(\underline{z}_k | \underline{x}_k\right) \cdot f_k^p\left(\underline{x}_k\right) ,$$

where c_k is a normalization constant and $f\left(\underline{z}_k | \underline{x}_k\right)$ is the likelihood function given by

$$f\left(\underline{z}_k | \underline{x}_k\right) = \int \delta\left(\underline{z}_k - \underline{h}_k\left(\underline{x}_k, \underline{v}_k\right)\right) \cdot f_k^v\left(\underline{v}_k\right) \mathrm{d}\underline{v}_k$$

and the measurement model (2).

In general, for arbitrary nonlinear systems with arbitrarily distributed random vectors, there exist no analytical solutions of the prediction step and filtering step. Thus, for efficient estimation, it is inevitable to apply an approximate solution. In the following, an adaptive approximation scheme is proposed, where the predicted and posterior state densities are represented by means of Gaussian mixtures

$$f_k^\bullet\left(\underline{x}_k\right) = \sum_{i=1}^{L_k^\bullet} \omega_{k,i}^\bullet \cdot \mathcal{N}\left(\underline{x}_k; \underline{\hat{x}}_{k,i}^\bullet, \mathbf{C}_{k,i}^\bullet\right) , \text{ with } \bullet \in \{e, p\} , \tag{6}$$

where the number $L_k^\bullet$ of mixture components is variable and adapted on-line by the proposed Gaussian mixture filter.

3 Statistical Linearization

Substituting the Gaussian mixtures representing the noise and the state density into the prediction step and the filtering step, it can be easily seen that estimation can be performed component-wise. For example in case of the prediction, using (3) and (6) with $\bullet = e$ in (5) gives rise to

$$f_{k+1}^p\left(\underline{x}_{k+1}\right) = \sum_{i=1}^{L_k^e} \sum_{j=1}^{L_k^w} \omega_{k,i}^e \cdot \omega_{k,j}^w \cdot \left(\iint f\left(\underline{x}_{k+1} | \underline{x}_k, \underline{u}_k, \underline{w}_k\right) \cdot \right.$$
$$\left. \mathcal{N}\left(\underline{x}_k; \underline{\hat{x}}_{k,i}^e, \mathbf{C}_{k,i}^e\right) \cdot \mathcal{N}\left(\underline{w}_k; \underline{\hat{w}}_{k,j}, \mathbf{C}_{k,j}^w\right) \mathrm{d}\underline{x}_k \, \mathrm{d}\underline{w}_k\right) . \tag{7}$$

Thus, it is sufficient to focus in the following on the simplified nonlinear transformation

$$\underline{\boldsymbol{y}} = \underline{g}(\underline{\boldsymbol{x}}) \,, \tag{8}$$

which maps the Gaussian random vector $\underline{\boldsymbol{x}}$ with density $\mathcal{N}\left(\underline{x}; \hat{\underline{x}}, \mathbf{C}^x\right)$ to the random vector $\underline{\boldsymbol{y}}$. This nonlinear transformation can be replaced by $\underline{a}_k(\cdot)$ in the prediction step and by $\underline{h}_k(\cdot)$ in the filtering step, while the Gaussian random vector $\underline{\boldsymbol{x}}$ in (8) represents the joint Gaussian of the state and noise.

3.1 Classical Linearization

Calculating the density or the statistics of $\underline{\boldsymbol{y}}$ cannot be carried out in closed form. Hence, *directly* processing the density or the moments is computationally demanding and imprecise, or even impossible. An exception are linear transformations, where the Kalman filter [12] provides analytic expressions of the Bayesian estimation problem. To apply the Kalman filter equations to nonlinear transformations, a typical way is to linearize the nonlinear transformation, which results in the extended Kalman filter [22]. Here, it is assumed that the nonlinear transformation can be approximated by a linear transformation through a first-order Taylor series expansion around the mean $\hat{\underline{x}}$. In case of mild nonlinearities the linearization error of this approximation is acceptable. However, for this type of linearization the spread of $\underline{\boldsymbol{x}}$, i.e., the covariance matrix $\mathbf{C}^x$ is not taken into account and there is no measure which allows to quantify the linearization error.

3.2 Statistical Linear Regression

To overcome these flaws, deterministic sampling techniques are employed instead, which allow for propagating the mean and the covariance of $\underline{\boldsymbol{x}}$ through the nonlinear transformation (8). In doing so, linearizing the transformation by so-called statistical linear regression or *statistical linearization* is possible [14, 23]. More precisely, statistical linearization calculates a matrix $\mathbf{G}$ and a vector $\underline{b}$ such that

$$\underline{\boldsymbol{y}} = \underline{g}(\underline{\boldsymbol{x}}) \approx \mathbf{G} \cdot \underline{\boldsymbol{x}} + \underline{b} \,, \tag{9}$$

where the error term

$$\underline{\boldsymbol{e}} = \underline{g}(\underline{\boldsymbol{x}}) - \mathbf{G} \cdot \underline{\boldsymbol{x}} - \underline{b} \tag{10}$$

describes the deviation of the nonlinear transformation and its linear approximation. To determine $\mathbf{G}$ and $\underline{b}$, the nonlinear transformation $\underline{g}(\cdot)$ is evaluated at a set of weighted *regression points* $\{\alpha_i, \underline{x}_i\}_{i=1\ldots L}$ with non-negative weights α_i with $\sum_i \alpha_i = 1$, which results in points $\underline{y}_i = \underline{g}(\underline{x}_i)$ for $i = 1\ldots L$. This set of points is chosen in such a way that the mean $\underline{\hat{x}}$ and covariance $\mathbf{C}^x$ of $\underline{\boldsymbol{x}}$ are captured exactly, that is

$$\underline{\hat{x}} = \sum_{i=1}^{L} \alpha_i \cdot \underline{x}_i \text{ and } \mathbf{C}^x = \sum_{i=1}^{L} \alpha_i \cdot (\underline{x}_i - \underline{\hat{x}}) \cdot (\underline{x}_i - \underline{\hat{x}})^{\mathrm{T}} .$$

Then $\mathbf{G}$ and $\underline{b}$ are determined by minimizing the weighted sum of squared errors

$$\{\mathbf{G}, \underline{b}\} = \arg\min_{\mathbf{G},\underline{b}} \left(\sum_{i=1}^{L} \alpha_i \cdot \underline{e}_i^{\mathrm{T}} \cdot \underline{e}_i \right) \tag{11}$$

with $\underline{e}_i = \underline{y}_i - (\mathbf{G} \cdot \underline{x}_i + \underline{b})$. The solution of (11) is given by

$$\mathbf{G} = \left(\mathbf{C}^{xy}\right)^{\mathrm{T}} \left(\mathbf{C}^{x}\right)^{-1} \text{ and } \underline{b} = \underline{\hat{y}} - \mathbf{G} \cdot \underline{\hat{x}} , \tag{12}$$

where the set of propagated points $\{\alpha_i, \underline{y}_i\}_{i=1\ldots L}$ is used to approximate the mean, covariance, and cross-covariance of $\underline{\boldsymbol{y}}$ according to

$$\underline{\hat{y}} \approx \sum_{i=1}^{L} \alpha_i \cdot \underline{y}_i , \quad \mathbf{C}^y \approx \sum_{i=1}^{L} \alpha_i \cdot (\underline{y}_i - \underline{\hat{y}}) \cdot (\underline{y}_i - \underline{\hat{y}})^{\mathrm{T}} ,$$
$$\mathbf{C}^{xy} \approx \sum_{i=1}^{L} \alpha_i \cdot (\underline{x}_i - \underline{\hat{x}}) \cdot (\underline{y}_i - \underline{\hat{y}})^{\mathrm{T}} .$$

The linearization error is characterized by the error term (10) and has zero-mean and the covariance matrix

$$\mathbf{C}^e = \mathbf{C}^y - \mathbf{G}\mathbf{C}^x\mathbf{G}^{\mathrm{T}} . \tag{13}$$

Thus, by means of the covariance $\mathbf{C}^e$ it is possible to quantify the linearization error. If $\mathbf{C}^e$ is a zero matrix, the density of the error $\underline{\boldsymbol{e}}$ corresponds to a Dirac delta distribution[16] and the transformation $\underline{g}(\cdot)$ is affine with $\underline{g}(\underline{\boldsymbol{x}}) = \mathbf{G} \cdot \underline{\boldsymbol{x}} + \underline{b}$.

3.3 Calculating the Regression Points

Many approaches for calculating the set of regression points have been proposed in the recent years. They differ in the number of regression points L and the way these points are chosen. In the following example, both selection schemes used in this paper are briefly introduced.

Example 1: Regression Points

In the simulations described in Section 6 the famous *unscented transform* [11] and the *Gaussian estimator* [8] are considered. For both, the calculation of the sigma points $\underline{x}_i \in \mathbb{R}^{n_x}$ can be summarized as

$$\begin{aligned}\underline{x}_1 &= \hat{\underline{x}} \,, \\ \underline{x}_i &= \hat{\underline{x}} + \nu_j \cdot \mathbf{P}_l \quad , \quad i = l + 1 + (j-1)\cdot n_x \,,\end{aligned}$$

where $\mathbf{P}_l$, $l = 1 \ldots n_x$ is the lth column of the matrix $\mathbf{P} = \sqrt{\mathbf{C}^x}$ and ν_j, $j = 1 \ldots N$ are scaling factors. This results in a number of $L = n_x \cdot N + 1$ regression points. The type of the matrix root, the scaling factors, and the weights α_i of the regression points depend on the considered selection scheme.

In case of the unscented transform, the Cholesky decomposition is chosen as matrix root. For the scaling factors holds $N = 2$ and $\nu_1 = \sqrt{n_x + \kappa} = -\nu_2$, where κ is a scaling parameter. The weights are $\alpha_1 = \frac{\kappa}{n_x+\kappa}$ and $\alpha_i = \frac{1}{2(n_x+\kappa)}$, for $i > 1$.

The Gaussian estimator utilizes the eigenvalue decomposition for calculating the matrix root. The number of scaling factors N and thus the total number L of regression points can be varied. Since the scaling factors result from solving a optimization problem, there is no closed-form expression. For $N = 2$ and $N = 4$, the scaling factors can be calculated to

$$\begin{aligned} N = 2: &\qquad \nu_j \in \{-1.2245, 1.2245\} \,, \\ N = 4: &\qquad \nu_j \in \{-1.4795, -0.5578, 0.5578, 1.4795\} \,. \end{aligned} \tag{14}$$

The regression points are equally weighted with $\forall i : \alpha_i = 1/L$.

It is important to note that the proposed adaptive Gaussian mixture filter is not restricted to these two selection schemes. In fact, any selection scheme for statistical linearization including those described in [2, 10, 20] can be used, depending on the considered application as well as the desired estimation performance and computational demand.

4 Splitting Scheme

Given a random vector $\underline{\boldsymbol{x}}$, whose density function $f^x(\underline{x})$ is a Gaussian mixture with L^x components according to (6), it is possible to linearize the nonlinear transformation $\underline{g}(\cdot)$ for each component of $f^x(\underline{x})$. This kind of component-wise or *local linearization* leads to an improved approximation of the true density function $f^y(\underline{y})$ of $\underline{\boldsymbol{y}}$ compared to a single, global linearization. To further improve the approximation, especially in case of strong nonlinearities and/or large variances of some components, the idea is to select a component of $f^x(\underline{x})$ and split it into several components with reduced weights and covariances. It was demonstrated for example in [1] that the filtering accuracy of local linearization approaches benefits from this decrease of the covariances and simultaneous increase of the number of Gaussians.

4.1 Component Selection

A straightforward way to select a Gaussian component for splitting is to consider the weights ω_i^x, $i = 1 \dots L^x$. The component with the highest weight is then split. This however does not take the nonlinearity of $\underline{g}(\cdot)$ in the support of the selected component into account. Since linearization is performed component-wise and locally, a more reasonable selection would be to consider also the induced linearization error of each component. For this purpose, statistical linearization already provides an appropriate measure for the linearization error in form of the covariance matrix $\mathbf{C}^e$ in (13).

In order to easily assess the linearization error in the multi-dimensional case, the trace operator is applied to $\mathbf{C}^e$, which gives the measure

$$\varepsilon = \mathrm{Tr}\left(\mathbf{C}^e\right) \in [0, \infty) \ . \tag{15}$$

Geometrically speaking, the trace is proportional to circumference of the covariance ellipsoid corresponding to $\mathbf{C}^e$. The larger $\mathbf{C}^e$ and thus the linearization error, the larger is ε. Conversely, the trace is zero, if and only if $\mathbf{C}^e$ is the zero matrix, i.e., $\varepsilon = 0 \Leftrightarrow \mathbf{C}^e = \mathbf{0}$. Hence, (15) is only zero, when there is no linearization error, that is, the nonlinear transformation $\underline{g}(\cdot)$ is affine in the support of the considered Gaussian component.

Besides the linearization error, the contribution of a component to the nonlinear transformation is important as well. That is, the probability mass of the component, which is given by its weight ω_i^x, has also to be taken into account. This avoids splitting irrelevant components. Putting all together the criterion for selecting a component i for splitting is defined as

$$s_i = \left(\omega_i^x\right)^\gamma \cdot \left(1 - \exp\left(-\varepsilon_i\right)\right)^{1-\gamma} \in [0,1] \tag{16}$$

for $i = 1 \ldots L^x$, where $1 - \exp(-\varepsilon_i)$ normalizes the linearization measure (15) into the interval [0,1]. For a geometric interpolation between weight and linearization error of component i, the parameter $\gamma \in [0,1]$ used. With $\gamma = 0$, selecting a component for splitting only focuses on the linearization error, while $\gamma = 1$ considers the weight only.

Component selection criteria for splitting have also been proposed in [5, 17]. The criterion in [5] is designed for the unscented transform only, while the criterion in [17] can only be calculated analytically in some special cases. The proposed criterion instead is generally applicable.

4.2 Splitting a Gaussian

Assume that according to the selection criterion (16), the Gaussian component $\omega \cdot \mathcal{N}\left(\underline{x}; \hat{\underline{x}}, \mathbf{C}^x\right)$ is chosen. Splitting this Gaussian into many can be formulated as replacing the Gaussian by a Gaussian mixture according to

$$\omega \cdot \mathcal{N}\left(\underline{x}; \hat{\underline{x}}, \mathbf{C}^x\right) \approx \sum_{j=1}^{L} \omega_j \cdot \mathcal{N}\left(\underline{x}; \hat{\underline{x}}_j, \mathbf{C}_j\right) . \tag{17}$$

It can be easily verified that for $L > 1$, the number of free parameters, i.e., weights, mean vectors and covariance matrices, is larger than the number of given parameters. More precisely, splitting a Gaussian is an ill-posed problem. In order to reduce the degrees of freedom and to not introduce errors concerning the mean and covariance, splitting is performed in a moment-preserving fashion. Thus, it must hold that

$$\omega = \sum_{j=1}^{L} \omega_j \, , \quad \hat{\underline{x}} = \sum_{j=1}^{L} \frac{\omega_j}{\omega} \cdot \hat{\underline{x}}_j \, , \quad \mathbf{C}^x = \sum_{j=1}^{L} \frac{\omega_j}{\omega} \cdot \left(\mathbf{C}_j + \hat{\underline{x}}_j \hat{\underline{x}}_j^{\mathrm{T}}\right) - \hat{\underline{x}} \hat{\underline{x}}^{\mathrm{T}} . \tag{18}$$

To further simplify the problem, splitting is restricted in direction of the eigenvectors of $\mathbf{C}^x$, which is computationally cheap and numerically stable. Furthermore, it reduces the problem to splitting a univariate standard Gaussian.

Univariate standard Gaussian

A moment-preserving split of a univariate standard Gaussian $\mathcal{N}(x;0,1)$ into a mixture with L components requires to determine $3 \cdot L$ free parameters. By forcing symmetry, i.e., the means $\hat{x}_j$ are placed symmetrically around the mean $\hat{x}$ with symmetrically chosen weights and for the variances holds $\forall j : \sigma_j^2 = \sigma$, the number of free parameters is reduced to $L+1$. In [7], a splitting library with symmetric components is proposed. Unfortunately, preserving the moments is not guaranteed. Instead, the following split into two components is used throughout this paper.

Example 2: Splitting into Two Components

Following the approach proposed in [6], the univariate standard Gaussian is split into the mixture $1/2 \cdot \mathcal{N}(x;\hat{x},\sigma^2) + 1/2 \cdot \mathcal{N}(x;-\hat{x},\sigma^2)$. The moment-preserving constraints of splitting (18) lead to the dependency $\sigma^2 = 1 - \hat{x}^2$ between $\hat{x}$ and σ, where $\hat{x}$ is now the only free parameter. This equation is valid for $\hat{x} \in [-1,1]$ and contains the trivial solution $\hat{x} = 0$. Generally, $\hat{x}$ may be determined dynamically by minimizing the resulting linearization error. But throughout this paper, $\hat{x}$ is set to 0.5 for simplicity.

To determine the parameters of more than two components, additional constraints, e.g., capturing higher order moments like the skewness or the kurtosis have to be considered additionally. Since splitting is performed recursively in this paper (see Section 5), the new introduced components can be split in the subsequent splitting step if the local linearization error may not be reduced sufficiently. Splitting into two components is a good compromise between reducing the linearization error on the one hand and controlling the growth of the number of components and the computational load on the other hand.

Multivariate Gaussian

Applying univariate splitting to the multivariate case requires the eigenvalue decomposition of the covariance matrix $\mathbf{C}^x = \mathbf{V}\mathbf{D}\mathbf{V}^\mathrm{T}$, with $\mathbf{V}$ being the matrix of eigenvectors and $\mathbf{D}$ being the diagonal matrix of eigenvalues according to

$$\mathbf{V} = \begin{bmatrix} \underline{v}_1 & \underline{v}_2 & \dots & \underline{v}_{n_x} \end{bmatrix} , \quad \mathbf{D} = \operatorname{diag}\left(\lambda_1, \lambda_2, \dots, \lambda_{n_x}\right) ,$$

where $\underline{v}_i \in \mathbb{R}^{n_x}$ are the (orthonormal) eigenvectors and λ_i are the eigenvalues. $\mathbf{V}$ is a rotation matrix and the eigenvalue decomposition of $\mathbf{C}^x$ corresponds to the transformation

$$\underline{\boldsymbol{x}} = \mathbf{V} \cdot \underline{\boldsymbol{z}} \tag{19}$$

of a Gaussian random vector $\underline{\boldsymbol{z}}$ with density

$$f^z(\underline{z}) = \mathcal{N}(\underline{z}; \hat{\underline{z}}, \mathbf{D}) = \prod_{i=1}^{n_x} \mathcal{N}\left(z_i; \hat{z}_i, \lambda_i\right) , \tag{20}$$

where $\hat{\underline{z}} = \mathbf{V}^{\mathrm{T}} \cdot \hat{\underline{x}}$, to a Gaussian random vector $\underline{\boldsymbol{x}}$ with density $\mathcal{N}\left(\underline{x}; \hat{\underline{x}}, \mathbf{C}^x\right)$. Since the Gaussian $f^z(\underline{z})$ has a diagonal covariance matrix, the eigenvectors are parallel to the axes of the coordinate system. Thus, univariate splitting can be easily applied along the eigenvectors by replacing a univariate Gaussian on the right-hand side of (20) by a Gaussian mixture.

Assume that eigenvector $\underline{v}_l$ is chosen for splitting and let $\sum_{j=1}^{L} \omega'_j \cdot \mathcal{N}\left(z_l; \hat{z}'_j, \sigma_j^2\right)$ be the Gaussian mixture that approximates a univariate standard Gaussian as described above. As this mixture approximates a standard Gaussian, its components have to be shifted by adding $\hat{z}_l$ and scaled by multiplying with $\sqrt{\lambda_l}$ in order to match the mean $\hat{z}_l$ and the variance λ_l, respectively. These operations result in

$$\mathcal{N}(z_l; \hat{z}_l, \lambda_l) \approx \sum_{j=1}^{L} \omega'_j \cdot \mathcal{N}\left(z_l; \hat{z}_l + \sqrt{\lambda_l} \hat{z}'_j, \lambda_l \sigma_j^2\right) . \tag{21}$$

Plugging (21) into (20) leads to

$$\mathcal{N}(\underline{z}; \hat{\underline{z}}, \mathbf{D}) \approx \sum_{j=1}^{L} \omega'_j \cdot \mathcal{N}\left(z_l; \hat{z}_l + \sqrt{\lambda_l} \hat{z}'_j, \lambda_l \sigma_j^2\right) \cdot \prod_{\substack{i=1 \\ i \neq l}}^{n_x} \mathcal{N}\left(z_i; \hat{z}_i, \lambda_i\right) .$$

Transforming this mixture via (19) gives the desired splitting result (17) with the weights, means, and covariance matrices

$$\omega_j = \omega \cdot \omega'_j , \quad \hat{\underline{x}}_j = \hat{\underline{x}} + \sqrt{\lambda_l} \cdot \hat{z}'_j \cdot \underline{v}_l , \quad \mathbf{C}_j = \mathbf{C}^x + \lambda_l \cdot (\sigma_j^2 - 1) \cdot \underline{v}_l \underline{v}_l^{\mathrm{T}} , \tag{22}$$

respectively, for $j = 1 \ldots L$.

It is worth mentioning that the calculation of the parameters in (22) is independent of the number of components L and does not necessarily require a symmetric, moment-preserving splitting. Thus, arbitrary splitting methods of univariate standard Gaussians besides those described in this paper, can be used with these formulae.

4.3 Splitting Direction

So far, no criterion for selecting an appropriate eigenvector for splitting is defined. A straightforward criterion may be the eigenvector with the largest eigenvalue as in [6, 7]. But since (16) determines the Gaussian component that causes the largest linearization error, merely splitting along the eigenvector with the largest eigenvalue does not take this error into account.

The key idea of the proposed criterion is to evaluate the deviation between the nonlinear transformation (8) and its linearized version (9) along each eigenvector. The eigenvector with the largest deviation is then considered for splitting, i.e., the Gaussian is split in direction of the largest deviation in order to cover this direction with more Gaussians, which will reduce the error in subsequent linearization steps.

By means of the error term (10), the desired criterion for the splitting direction is defined as

$$d_l \triangleq \int_{\mathbb{R}} \underline{\boldsymbol{e}}\big(\underline{x}_l(v)\big)^{\mathrm{T}} \cdot \underline{\boldsymbol{e}}\big(\underline{x}_l(v)\big) \cdot \mathcal{N}\big(\underline{x}_l(v); \hat{\underline{x}}, \mathbf{C}^x\big)\,\mathrm{d}v \tag{23}$$

with $\underline{x}_l(v) := \hat{\underline{x}} + v \cdot \underline{v}_l$, $l = 1 \ldots n_x$, and $\underline{v}_l$ being the lth eigenvector $\mathbf{C}^x$. The integral in (23) cumulates the squared deviations along the lth eigenvector under the consideration of the probability at each point $\underline{x}_l(v)$. The eigenvector that maximizes (23) is then chosen for splitting. Unfortunately, due to the nonlinear transformation $\underline{g}(\cdot)$ in (10), this integral cannot be solved in closed-form in general. For an efficient and approximate solution, the regression point calculation schemes described in Section 3.3 are employed to approximate the Gaussian in (23) in direction of $\underline{v}_l$ by means of a Dirac mixture. This automatically leads to a discretization of the integral at a few but carefully chosen points.

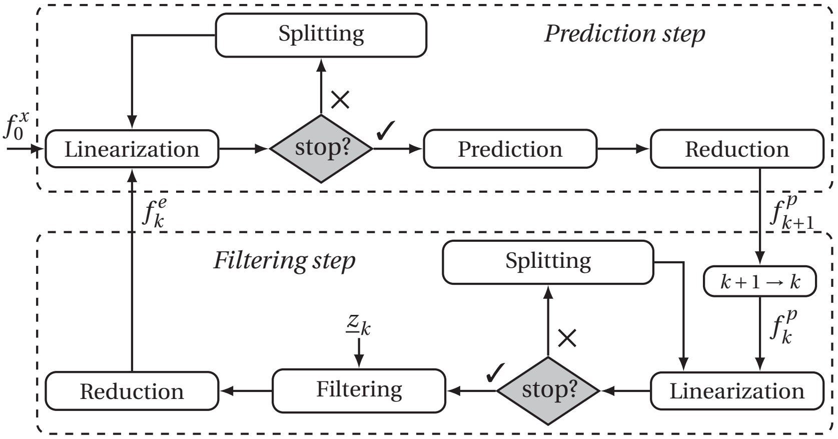


Figure 1: Flow chart of the proposed adaptive Gaussian mixture filter. Both the prediction and filtering step employ splitting and reduction for adapting the number of mixture components.

5 Adaptive Gaussian Mixture Filter

Based on the statistical linearization described in Section 3 and the splitting procedure proposed in Section 4, the complete adaptive Gaussian mixture filter (AGMF) is now derived. The key idea of AGMF is to dynamically increase the number of Gaussians of a given mixture at regions with large linearization errors. The number is reduced after each prediction and filtering in order to limit the computational and memory demand.

5.1 Prediction Step

The major operations to be performed in the prediction step are illustrated in the upper part of Figure 1. The following paragraphs provide detailed descriptions of these operations.

Linearization

As shown in (7) the prediction step can be performed component-wise. Therefore, the nonlinear system function is linearized statistically at the weighted joint Gaussian $\omega_{k,s} \cdot \mathcal{N}\big(\underline{X}_k; \hat{\underline{X}}_{k,s}, \mathbf{C}^X_{k,s}\big)$ with $\underline{\boldsymbol{X}}_k = \big[\underline{\boldsymbol{x}}_k^{\mathrm{T}}, \underline{\boldsymbol{w}}_k^{\mathrm{T}}\big]^{\mathrm{T}}$, $s = (i-1) \cdot L^w_k + j = 1 \ldots L^e_k \cdot L^w_k$, $\omega_{k,s} = \omega^e_{k,i} \cdot \omega^w_{k,j}$, and

$$\hat{\underline{X}}_{k,s} = \begin{bmatrix} \hat{\underline{x}}^e_{k,i} \\ \hat{\underline{w}}_{k,j} \end{bmatrix} , \quad \mathbf{C}^X_{k,s} = \begin{bmatrix} \mathbf{C}^e_{k,i} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}^w_{k,j} \end{bmatrix} .$$

With (12), the linearization results in

$$\underline{\boldsymbol{x}}_{k+1} = \underbrace{\begin{bmatrix} \mathbf{A}_{k,s} & \mathbf{B}_{k,s} \end{bmatrix}}_{=\mathbf{G}_{k,s}} \cdot \underline{\boldsymbol{X}}_k + \underline{b}_{k,s} . \tag{24}$$

Splitting

Due to the nonlinearity of the system function $\underline{a}_k(\cdot)$, some of the mixture components $\omega_{k,s} \cdot \mathcal{N}\big(\underline{X}_k; \hat{\underline{X}}_{k,s}, \mathbf{C}^X_{k,s}\big)$ may locally cause severe linearization errors. These errors are quantified by means of the selection criterion (16). The component maximizing (16) will be split in direction of the largest deviation between the nonlinear function $\underline{a}_k(\cdot)$ and its linearized version (24) as described in Section 4.3. After splitting this Gaussian, linearization is performed for the newly introduced mixture components. The linearization need not to be repeated for the remaining mixture components as they are not affected by the splitting.

Splitting Gaussians and the subsequent linearization is repeated until a stopping criterion is satisfied. This stopping criterion combines three user-defined thresholds:

1. For *each* component the value of the selection criterion (16) shall drop below the *error threshold* $\varepsilon_{\max} \in [0,1]$.
2. The number of Gaussians shall not grow beyond the *component threshold* $L_{\max}$.
3. The deviation between the original Gaussian mixture $f(\underline{x})$ and the mixture obtained via splitting $\tilde{f}(\underline{x})$ shall remain below a *deviation threshold* $d_{\max} \in [0,1]$.

In the latter case, the deviation is determined by means of the normalized integral squared distance measure [9]

$$D\big(f(\underline{x}), \tilde{f}(\underline{x})\big) = \frac{\int \big(f(\underline{x}) - \tilde{f}(\underline{x})\big)^2 \, \mathrm{d}\underline{x}}{\int f(\underline{x})^2 \, \mathrm{d}\underline{x} + \int \tilde{f}(\underline{x})^2 \, \mathrm{d}\underline{x}} \in [0,1] .$$

Since splitting always introduces an approximation error to the original mixture $f(\underline{x})$, tracking the deviation during splitting and keeping the deviation below the threshold $d_{\max}$ avoids that errors introduced by splitting neutralize the gain in linearization. Splitting stops, if at least one threshold is reached.

Prediction

Let $\omega_{k,\tilde{s}} \cdot \mathcal{N}\big(\underline{X}_k; \hat{\underline{X}}_{k,\tilde{s}}, \mathbf{C}^X_{k,\tilde{s}}\big)$ be the Gaussians resulting from the splitting step, with $\tilde{s} = 1 \ldots \tilde{L}^p_k$ and $\tilde{L}^p_k \gg L^e_k \cdot L^w_k$. Based on these Gaussians and their corresponding locally linearized system models (24), the parameters of each component of the predicted Gaussian mixture $f^p_{k+1}\big(\underline{x}_{k+1}\big)$ can be calculated by means of the Kalman predictor according to

$$\begin{aligned}
\omega^p_{k+1,\tilde{s}} &= \omega_{k,\tilde{s}} \ , \\
\hat{\underline{x}}^p_{k+1,\tilde{s}} &= \mathbf{A}_{k,\tilde{s}} \cdot \hat{\underline{x}}^e_{k,\tilde{s}} + \mathbf{B}_{k,\tilde{s}} \cdot \hat{\underline{w}}_{k,\tilde{s}} + \underline{b}_{k,\tilde{s}} \ , \\
\mathbf{C}^p_{k+1,\tilde{s}} &= \mathbf{A}_{k,\tilde{s}} \mathbf{C}^e_{k,\tilde{s}} \mathbf{A}^{\mathrm{T}}_{k,\tilde{s}} + \mathbf{B}_{k,\tilde{s}} \mathbf{C}^w_{k,\tilde{s}} \mathbf{B}^{\mathrm{T}}_{k,\tilde{s}} + \mathbf{C}_{k,\tilde{s}} \ ,
\end{aligned}$$

where $\mathbf{C}_{k,\tilde{s}}$ is the linearization error covariance (13).

Reduction

The number of components $\tilde{L}^p_k$ in $f^p_{k+1}\big(\underline{x}_{k+1}\big)$ grows due to the multiplication of the Gaussian mixtures $f^e_k\big(\underline{x}_k\big)$ and $f^w_k\big(\underline{w}_k\big)$ for prediction and due to splitting. It is necessary to bound this growth in order to reduce the computational and memory demand of subsequent prediction and filtering steps. For this purpose, one can exploit the redundancy and similarity of Gaussian components. Furthermore, many components will have weights close to zero, thus they can be removed without introducing significant errors. To reduce a Gaussian mixture, many algorithms have been proposed in the recent years (see for example [9, 18, 19, 25]). Most of these algorithms require a *reduction threshold* L^p_{k+1}—typically much smaller than $L_{\max}$—to which the number of components of the given Gaussian mixture has to be reduced. In the simulations, Runnalls' reduction algorithm [18] is employed as it provides a good trade-off between computational demand and reduction errors.

With the reduction to L^p_{k+1} components, the calculation of the predicted Gaussian mixture $f^p_{k+1}\big(\underline{x}_{k+1}\big)$ in (6) is finished.

5.2 Filtering Step

The operations to be performed for the filtering step are almost identical to the prediction step (see Figure 1). Thus, only linearization and filtering are described in the following. Splitting and reduction coincide with the prediction step.

Linearization

Linearization and filtering are also performed component-wise. In the following, let $\omega_{k,s} \cdot \mathcal{N}\left(\underline{X}_k, \underline{\hat{X}}_{k,s}, \mathbf{C}^X_{k,s}\right)$ be the joint Gaussian comprising the ith component of the predicted mixture $f^p_k\left(\underline{x}_k\right)$ and the jth component of the measurement noise mixture (4), where $s = (i-1) \cdot L^v_k + j = 1 \dots L^p_k \cdot L^v_k$. The corresponding linearized measurement model is

$$\underline{\boldsymbol{z}}_k = \begin{bmatrix} \mathbf{H}_{k,s} & \mathbf{D}_{k,s} \end{bmatrix} \cdot \underline{\boldsymbol{X}}_k + \underline{b}_{k,s} \tag{25}$$

with joint state $\underline{\boldsymbol{X}}_k = \left[\underline{\boldsymbol{x}}^{\mathrm{T}}_k, \underline{\boldsymbol{v}}^{\mathrm{T}}_k\right]^{\mathrm{T}}$.

Filtering

Let $\omega_{k,\tilde{s}} \cdot \mathcal{N}(\underline{X}_k; \underline{\hat{X}}_{k,\tilde{s}}, \mathbf{C}^X_{k,\tilde{s}})$ be the Gaussians resulting from splitting, with $\tilde{s} = 1 \dots \tilde{L}^e_k$ and $\tilde{L}^e_k \gg L^p_k \cdot L^v_k$. Given the current measurement value $\underline{z}_k$, the Kalman filter update equations applied on these Gaussians and their corresponding locally linearized measurement models (25) give rise to the parameters of each component of the posterior Gaussian mixture $f^e_k\left(\underline{x}_k\right)$

$$\begin{aligned}
\omega^e_{k,\tilde{s}} &= c_k \cdot \omega_{k,\tilde{s}} \cdot \mathcal{N}\left(\underline{z}_k; \underline{\hat{z}}_{k,\tilde{s}}, \mathbf{S}_{k,\tilde{s}}\right) , \\
\underline{\hat{x}}^e_{k,\tilde{s}} &= \underline{\hat{x}}^p_{k,\tilde{s}} + \mathbf{K}_{k,\tilde{s}}\left(\underline{z}_k - \underline{\hat{z}}_{k,\tilde{s}}\right) , \\
\mathbf{C}^e_{k,\tilde{s}} &= \mathbf{C}^p_{k,\tilde{s}} - \mathbf{K}_{k,\tilde{s}}\mathbf{H}_{k,\tilde{s}}\mathbf{C}^p_{k,\tilde{s}} ,
\end{aligned} \tag{26}$$

with predicted measurement $\underline{\hat{z}}_{k,\tilde{s}} = \mathbf{H}_{k,\tilde{s}} \cdot \underline{\hat{x}}^p_{k,\tilde{s}} + \mathbf{D}_{k,\tilde{s}} \cdot \underline{\hat{v}}_{k,\tilde{s}} + \underline{b}_{k,\tilde{s}}$, Kalman gain $\mathbf{K}_{k,\tilde{s}} = \mathbf{C}^p_{k,\tilde{s}}\mathbf{H}^{\mathrm{T}}_{k,\tilde{s}}\mathbf{S}^{-1}_{k,\tilde{s}}$, innovation covariance $\mathbf{S}_{k,\tilde{s}} = \mathbf{H}_{k,\tilde{s}}\mathbf{C}^p_{k,\tilde{s}}\mathbf{H}^{\mathrm{T}}_{k,\tilde{s}} + \mathbf{D}_{k,\tilde{s}}\mathbf{C}^v_{k,\tilde{s}}\mathbf{D}^{\mathrm{T}}_{k,\tilde{s}} + \mathbf{C}_{k,\tilde{s}}$, and $\mathbf{C}_{k,\tilde{s}}$ being the linearization error covariance (13). The calculation of the weight $\omega^e_{k,\tilde{s}}$ in (26) is adapted from [1, 21], where $c_k = 1/\sum_{\tilde{s}} \omega_{k,\tilde{s}} \cdot \mathcal{N}\left(\underline{z}_k; \underline{\hat{z}}_{k,\tilde{s}}, \mathbf{S}_{k,\tilde{s}}\right)$ is a normalization constant.

After the reduction to L^e_k components, the posterior Gaussian mixture $f^e_k\left(\underline{x}_k\right)$ in (6) is completely determined.

6 Simulation Results

Two numerical simulations are conducted in order to demonstrate the performance of the proposed AGMF.

6.1 Shape Approximation

In the first simulation, the nonlinear growth process

$$\boldsymbol{y} = g(\underline{\boldsymbol{x}}) = \frac{\boldsymbol{\xi}}{2} + 5 \cdot \frac{\boldsymbol{\xi}}{1+\boldsymbol{\xi}^2} + \boldsymbol{w}$$

adapted from [13] is considered, where $\underline{\boldsymbol{x}} = [\boldsymbol{\xi}, \boldsymbol{w}]^{\mathrm{T}} \sim f^x(\underline{x}) = \mathcal{N}\left(\underline{x}; [1,0]^{\mathrm{T}}, \mathbf{I}_2\right)$ with $\mathbf{I}_n$ being the $n \times n$ identity matrix. To approximate the density of $\boldsymbol{y}$, the Gaussian $f^x(\underline{x})$ is split recursively into a Gaussian mixture, where the number of components is always doubled until a maximum of 64 components is reached. No mixture reduction and no thresholds $\varepsilon_{\max}$, $d_{\max}$ are used. The Gaussian estimator with 4 scaling factors according to (14) is employed for statistical linearization. The true density of $\boldsymbol{y}$ is calculated via numerical integration.

Two different values for the parameter γ of the selection criterion (16) are used: $\gamma = 0.5$, which makes no preference between the component weight and the linearization error and $\gamma = 1$, which considers the weight only. Furthermore, a rather simple selection criterion is considered for comparison, where selecting a Gaussian for splitting is based on the weights only (as it is the case for $\gamma = 1$), while the splitting is performed in direction of the eigenvector with the largest eigenvalue.

Table 1 shows the Kullback-Leibler divergence (KLD, [4]) between the true density of $\boldsymbol{y}$ and the approximations obtained by splitting. The approximations of the proposed splitting scheme are significantly better than the approximations of the largest eigenvalue scheme. This follows from the fact that the proposed scheme not only considers the spread of a component. It also takes the linearization errors into account. In doing so, the Gaussians are always split along the eigenvector that is closest to $\boldsymbol{\xi}$, since this variable is transformed nonlinearly, while $\boldsymbol{w}$ is not. This is different for the largest eigenvalue scheme, which wastes nearly half of the splits on $\boldsymbol{w}$.

The inferior approximation quality for $\gamma = 1$ compared to $\gamma = 0.5$ results from splitting components, which may have a high importance due to their weight but

Table 1: Approximation error (KLD × 10) for different splitting schemes and numbers of components.

splitting scheme	number of Gaussians						
	1	2	4	8	16	32	64
max. eigenvalue	2.01	0.77	0.64	0.47	0.39	0.21	0.26
$\gamma = 1$	2.01	0.77	0.59	0.34	0.20	0.12	0.07
$\gamma = 0.5$	2.01	0.77	0.40	0.22	0.07	0.03	0.02

which do not cause severe linearization errors. Thus, splitting these components will not improve the approximation quality much.

In Figure 2, the approximate density of $\boldsymbol{y}$ is depicted for different numbers of mixture components for $\gamma = 0.5$. With an increasing number of components, the approximation approaches the true density very well.

6.2 Object Tracking

For the second simulation example, a object tracking scenario is considered. The kinematics of the mobile object are modeled by means of the bicycle model

$$\underline{\boldsymbol{x}}_{k+1} \triangleq \begin{bmatrix} \boldsymbol{x}_{k+1} \\ \boldsymbol{y}_{k+1} \\ \boldsymbol{\phi}_{k+1} \end{bmatrix} = \underline{\boldsymbol{x}}_k + \begin{bmatrix} \cos(\boldsymbol{\phi}_k) \\ \sin(\boldsymbol{\phi}_k) \\ u_k \end{bmatrix} + \underline{\boldsymbol{w}}_k \,,$$

where the system state $\underline{\boldsymbol{x}}_k$ comprises the position $[\boldsymbol{x}_k, \boldsymbol{y}_k]^{\mathrm{T}}$ and the orientation $\boldsymbol{\phi}_k$ of the bicycle. At time step $k = 0$, the initial estimate of the state $\underline{\boldsymbol{x}}_0$ is represented by a Gaussian density with mean $\hat{\underline{x}}_0 = [100\,\mathrm{m}, 100\,\mathrm{m}, 0\,\mathrm{rad}]^{\mathrm{T}}$ and covariance matrix $\mathbf{C}_0^x = \operatorname{diag}(10^2, 10^2, \pi^2)$. The system input $u_k := \tan(\alpha_k)$ with α_k being the steering angle, is chosen randomly and uniformly distributed from the interval $[-0.2, 0.2]$ at each time step. The system noise $\underline{\boldsymbol{w}}_k$ is zero-mean Gaussian with covariance matrix $\mathbf{C}_k^w = \operatorname{diag}(0.1^2, 0.1^2, 0.01^2)$.

A radar sensor with measurement model

$$\underline{\boldsymbol{z}}_k = \begin{bmatrix} \sqrt{\boldsymbol{x}_k^2 + \boldsymbol{y}_k^2} \\ \arctan\left(\boldsymbol{y}_k / \boldsymbol{x}_k\right) \end{bmatrix} + \underline{\boldsymbol{v}}_k$$

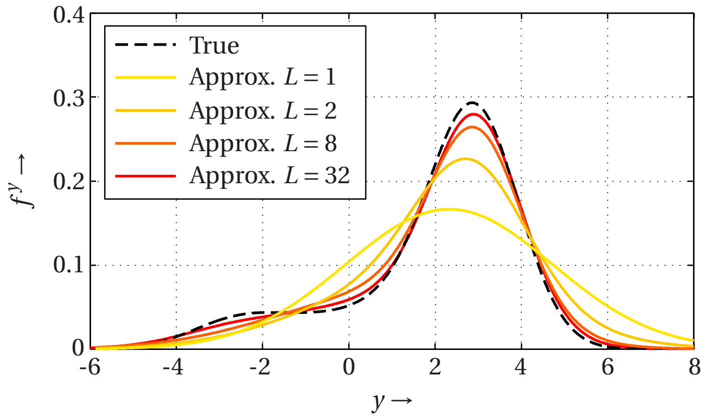
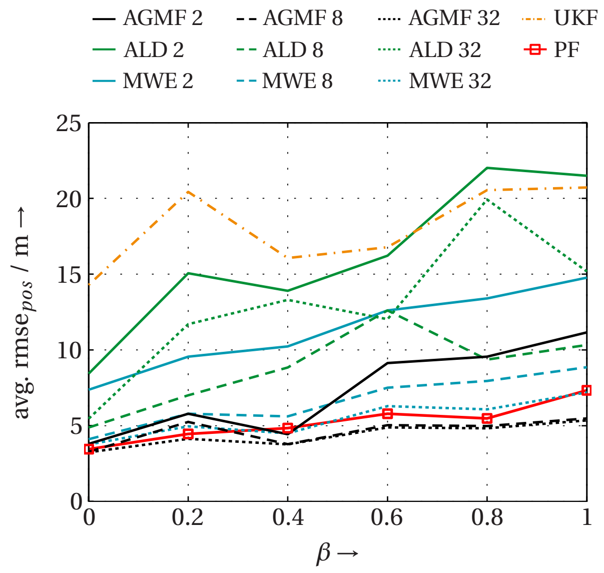


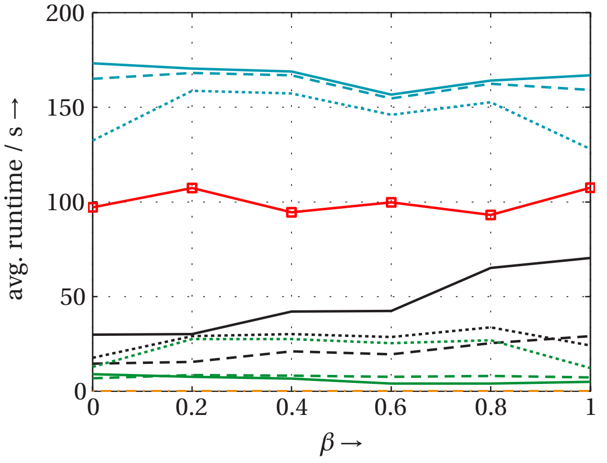
Figure 2: True density function of $\boldsymbol{y}$ (black, dashed) and approximations with an increasing number of mixture components.

is employed for observing the object, where the measurement noise $\underline{\boldsymbol{v}}_k$ is modeled as unimodal glint noise [26] with density $f_k^v(\underline{v}_k) = (1-\beta)\cdot\mathcal{N}(\underline{v}_k; \underline{0}, \mathbf{C}_{k,1}^v) + \beta\cdot\mathcal{N}(\underline{v}_k; \underline{0}, \mathbf{C}_{k,2}^v)$ with covariances $\mathbf{C}_{k,1}^v = \text{diag}(1^2, 0.1^2)$ and $\mathbf{C}_{k,2}^v = \text{diag}(2^2, 0.2^2)$. The parameter β refers to the glint noise probability. Six probability values $\beta = \{0, 0.2, \ldots, 1\}$ are exploited for simulation. By increasing β it is possible to investigate the performance of the filters for stronger noise, which is also heavily tailed for $\beta \neq 0$ and $\beta \neq 1$.

For this simulation setup, AGMF is applied with parameter $\gamma = 0.5$, error threshold $\varepsilon_{\max} = 0.05$, deviation threshold $d_{\max} = 1$, and component threshold $L_{\max} = 128$ for both prediction and filtering. Three values of reduction thresholds are used, $L_k^p = L_k^e = 2, 8, 32$. For comparison, a Gaussian mixture filter (denoted as MWE) employing the simple largest-weight-largest-eigenvalue-criterion as described in the previous section is considered. Further, the *adaptive level of detail* (ALD) Gaussian mixture filter proposed in [6] is employed as well. Since ALD is only designed for the unscented transform (see Example 1), this statistical linearization method is also used for AGMF to allow a fair comparison. The scaling parameter κ of the unscented transform is set to 0.5, i.e., all regression points are equally weighted. MWE and ALD use the same parameters as AGMF, except that MWE always splits until $L_{\max}$ is reached since it exploits no linearization errors.



(a) Average rmse over all simulation runs.



(b) Average runtime per simulation run.

Figure 3: Average rmseand runtime for different β values.

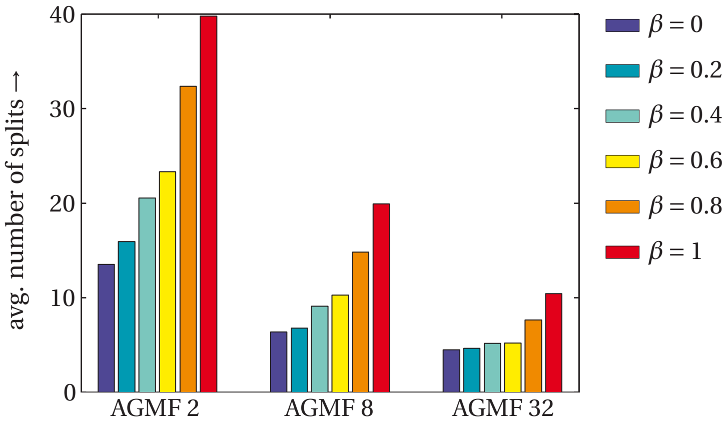


Figure 4: Average number of splits performed per prediction or filtering step of the AGMF for each β value.

Besides these Gaussian mixture filters, a particle filter (PF) with residual resampling and 10,000 samples as well as the unscented Kalman filter (UKF, [24]) with $\kappa = 0.5$ are also applied.

For each glint probability and each reduction threshold, 50 Monte Carlo simulation runs are performed, where the object is observed for 100 time steps. In Figure 3, the average root means square error (rmse) of the position and the average runtime per simulation run are depicted. The AGMFs with 8 and 32 components provide the best tracking performance. The PF is close to AGMF, but with a significantly higher runtime. Conversely, the UKF is by far the fastest algorithm, but leads to diverging estimates.

The splitting criterion used for ALD selects components that exhibit a high degree of nonlinearity. But splitting is performed merely in direction of the largest eigenvalue. This explains the relative poor tracking performance of ALD.

Even if MWE is allowed to split until $L_{\max}$ is reached, the performance of MWE is always inferior to AGMF. This is due to wasting many splits, e.g., in the prediction step only one quarter and less of the splits is used for $\boldsymbol{\phi}_k$, which is the only nonlinearly transformed variable. Here, AGMF is much more effective thanks to the novel splitting criterion. Besides splitting mainly in direction of the nonlinearity, it does not require all available splits as shown in Figure 4. The maximum number of splits is $L_{\max} - L_k^p$ in the prediction step and analogously in the filtering step. But at most 40 splits are performed in case of the strongest noise and when the state mixture is reduced to two components. If more components are allowed to represent the state density, the number of splits decreases as the approximation before splitting is already of high quality. This also reduces the runtime as can be seen when comparing for example AGMF 32 with AGMF 2. Here, the time consuming splitting operation has to be performed less often and the reduction operation has to reduce a mixture with an already low number of components.

7 Conclusions

In this paper, a novel adaptive Gaussian mixture filter has been proposed. It is based on statistical linearization, which allows quantifying the induced linearization errors in terms of a linearization error covariance matrix. A criterion based on this covariance matrix is used for selecting Gaussian components for splitting, while the direction of the split is performed in direction of the eigenvalue with the strongest linearization errors. Compared to other splitting criteria, the proposed one reliably detects strong nonlinearities and keeps the number of splits on a low level. Furthermore, arbitrary approaches for statistical linearization can be employed.

References

[1] Daniel L. Alspach and Harold W. Sorenson. Nonlinear Bayesian Estimation using Gaussian Sum Approximation. *IEEE Transactions on Automatic Control*, 17(4):439–448, August 1972.

[2] Ienkaran Arasaratnam and Simon Haykin. Cubature Kalman Filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, June 2009.

[3] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[4] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.

[5] Friedrich Faubel and Dietrich Klakow. An Adaptive Level of Detail Approach to Nonlinear Estimation. In *Proceedings of the 2010 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3958–3961, 2010.

[6] Friedrich Faubel, John McDonough, and Dietrich Klakow. The Split and Merge Unscented Gaussian Mixture Filter. *IEEE Signal Processing Letters*, 16(9):786–789, September 2009.

[7] Marco F. Huber, Tim Bailey, Hugh Durrant-Whyte, and Uwe D. Hanebeck. On Entropy Approximation for Gaussian Mixture Random Vectors. In *Proceedings of the 2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 181–188, Seoul, Republic of Korea, August 2008.

[8] Marco F. Huber and Uwe D. Hanebeck. Gaussian Filter based on Deterministic Sampling for High Quality Nonlinear Estimation. In *Proceedings of the 17th IFAC World Congress*, Seoul, Republic of Korea, July 2008.

[9] Marco F. Huber and Uwe D. Hanebeck. Progressive Gaussian Mixture Reduction. In *Proceedings of the 11th International Conference on Information Fusion (Fusion)*, Cologne, Germany, July 2008.

[10] Kazufumi Ito and Kaiqi Xiong. Gaussian Filters for Nonlinear Filtering Problems. *IEEE Transactions on Automatic Control*, 45(5):910–927, May 2000.

[11] Simon J. Julier and J. K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[12] Rudolf E. Kalman. A new Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME, Journal of Basic Engineering*, 82 (Series D)(1):35–45, 1960.

[13] Genshiro Kitagawa. Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[14] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter. *Nonlinear Kalman Filtering for Force-Controlled Robot Tasks*. Springer Berlin, 2005.

[15] Vladimir Maz'ya and Gunther Schmidt. On approximate approximations using gaussian kernels. *IMA J. Numer. Anal.*, 16:13–29, 1996.

[16] Kalyanapuram R. Parthasarathy. *Probability Measures on Metric Spaces*. American Mathematical Society, new edition, 2005.

[17] Andreas Rauh, Kai Briechle, and Uwe D. Hanebeck. Nonlinear Measurement Update and Prediction: Prior Density Splitting Mixture Estimator. In *Proceedings of the 2009 IEEE International Conference on Control Applications (CCA)*, July 2009.

[18] Andrew R. Runnalls. Kullback-Leibler Approach to Gaussian Mixture Reduction. *IEEE Transactions on Aerospace and Electronic Systems*, 43(3):989–999, July 2007.

[19] David J. Salmond. Mixture reduction algorithms for target tracking in clutter. In *Proceedings of SPIE Signal and Data Processing of Small Targets*, volume 1305, pages 434–445, October 1990.

[20] Tor S. Schei. A finite difference method for linearizing in nonlinear estimation algorithms. *Automatica*, 33(11):2051–2058, November 1997.

[21] Miroslav Simandl and Jindrich Duník. Sigma point gaussian sum filter design using square root unscented filters. In *Proceedings of the 16th IFAC World Congress*, Prague, Czech Republic, July 2005.

[22] Dan Simon. *Optimal State Estimation: Kalman, H-Infinity, and Nonlinear Approaches*. Wiley & Sons, 1 edition, 2006.

[23] Tom Vercauteren and Xiaodong Wang. Decentralized Sigma-Point Information Filters for Target Tracking in Collaborative Sensor Networks. *IEEE Transactions on Signal Processing*, 53(8):2997–3009, 2005.

[24] Eric A. Wan and Rudolph van der Merwe. The Unscented Kalman Filter for Nonlinear Estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000*, pages 153–158, 2000.

[25] Mike West. Approximating Posterior Distributions by Mixtures. *Journal of the Royal Statistical Society: Series B*, 55(2):409–422, 1993.

[26] Weng-Rong Wu. Target Tracking with Glint Noise. *IEEE Transactions on Aerospace and Electronic Systems*, 29(1):174–185, 1993.

Paper G

Superficial Gaussian Mixture Reduction

Authors: Marco F. Huber, Peter Krauthausen, and Uwe D. Hanebeck

Edited version of the paper:

M. F. Huber, P. Krauthausen, and U. D. Hanebeck. Superficial Gaussian Mixture Reduction. In *INFORMATIK 2011 - the 41th Annual Conference of the Gesellschaft für Informatik e.V. (GI), 6th Workshop Sensor Data Fusion: Trends, Solutions, Applications (SDF),* Berlin, Germany, October 2011.

Superficial Gaussian Mixture Reduction

Marco F. Huber[*], Peter Krauthausen[**], and Uwe D. Hanebeck[**]

[*]AGT International
Darmstadt, Germany
marco.huber@ieee.org

[**] Intelligent Sensor-Actuator-Systems
Laboratory (ISAS)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
peter.krauthausen@kit.edu,
uwe.hanebeck@ieee.org

Abstract

Many information fusion tasks involve the processing of Gaussian mixtures with simple underlying shape, but many components. This paper addresses the problem of reducing the number of components, allowing for faster density processing. The proposed approach is based on identifying components irrelevant for the overall density's shape by means of the curvature of the density's surface. The key idea is to minimize an upper bound of the curvature while maintaining a low global reduction error by optimizing the weights of the original Gaussian mixture only. The mixture is reduced by assigning zero weights to reducible components. The main advantages are an alleviation of the model selection problem, as the number of components is chosen by the algorithm automatically, the derivation of simple curvature-based penalty terms, and an easy, efficient implementation. A series of experiments shows the approach to provide a good trade-off between quality and sparsity.

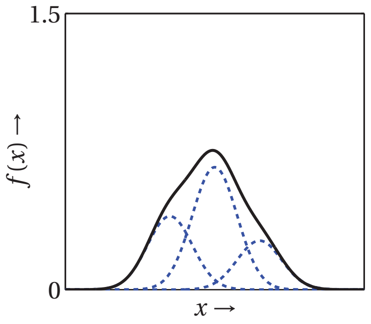
1 Introduction

Gaussian mixtures as a weighted sum of Gaussian densities are an often used function system in various information fusion applications, such as Bayesian filtering [1, 8], multi-target tracking [2], density estimation [16], or machine learning [5], just to name a few. Since the space of Gaussian densities forms a complete basis system, Gaussian mixtures can approximate every function with arbitrary accuracy [10]. Unfortunately, the number of Gaussian components of a Gaussian mixture tends to grow exponentially when processed recursively. To control this growth and thus, to bound computational and memory demands, Gaussian mixture reduction algorithms have to be applied continually.

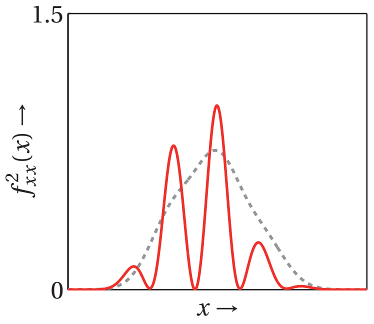
In recent years, many reduction algorithms have been proposed. Most of them employ a top-down approach: Two or more components of the Gaussian mixture with strong similarity are merged or components that do not contribute much to the mixture are deleted. These operations are performed recursively in a greedy fashion. The reduction stops as soon as an user-defined threshold on the number of components is reached. To quantify the similarity between components, local distance measures such as the Mahalanobis distance as in [13, 17] or global distance measures such as the integral squared distance as in [14, 18] and the Kullback-Leibler divergence as in [12] are used. Compared to local approaches, the deviation from the original Gaussian mixture when applying global optimization is typically much lower, at the expense of a higher computational load. Merging components is performed in a moment-preserving way, i.e., the mean and the covariance of the mixture remain unchanged. Independent of the used distance measure, top-down approaches have two severe drawbacks. First, merging and deleting components is performed greedily without considering the global effect in successive reduction steps. The second drawback results from the user-defined threshold on the final number of components. From a statistical point of view, the choice of this threshold corresponds to a model selection. The reduction algorithm has to reduce the mixture even if the resulting model is inappropriate, which for example leads to oversmoothed modes.

To overcome these drawbacks, a bottom-up approach named progressive Gaussian mixture reduction has recently been proposed [9]. Here, a Gaussian mixture is successively constructed to approximate the original mixture with far less components. Starting with a single Gaussian, new components are added at regions of strong deviation. Thus, the number of components is chosen automatically by the algorithm, which avoids the bias introduced by a predefined threshold. Unfortunately, PGMR is only able to construct mixtures with axis-aligned Gaussian components. It further requires a complex implementation.

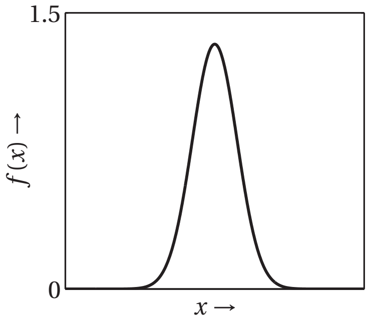
In this paper, a global optimization approach named *superficial Gaussian mixture reduction* (SGMR) is proposed, which is based on minimizing the curvature of the reduced Gaussian mixture while keeping the integral squared distance (ISD) low. This top-down approach allows for the reduction of arbitrary Gaussian mixtures while preserving the mean. As illustrated in Figure 1, the curvature



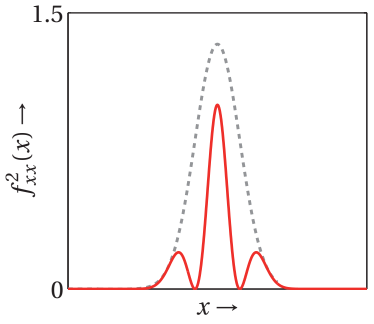
(a) Gaussian mixture with three components.



(b) Pointwise upper bound of the squared curvature (normalized) of the mixture in (a).



(c) The mixture in (a) reduced to the middle Gaussian.



(d) Pointwise upper bound of the squared curvature (normalized) for the single Gaussian.

Figure 1: Two different Gaussian mixtures and their curvatures.

captures “small bumps" in densities with an overall simple shape, i.e., with clear modes. Using the curvature it is possible to identify similar components globally and remove these components from the density. Carrying the idea that similar components may be dropped a step further, the trade-off between curvature and approximation error is minimized by merely optimizing weights, thus assigning zeros weights to reduced components. This approach is computationally feasible and allows a simple and efficient implementation based on standard quadratic program (QP) solvers. Additionally, this weight-only optimization alleviates the model selection problem, as the final number of components is automatically derived from setting the trade-off between error and roughness.

The next section gives a brief introduction to the Gaussian mixture reduction problem. The rest of the paper is structured as follows: The theoretical background of the proposed SGMR algorithm is described in Section 3. Here, an upper bound of the curvature is derived, which acts as a roughness penalty. In Section 4, limitations of the proposed approach are discussed. By means of numerical experiments, SGMR is compared to state-of-the-art mixture reduction algorithms in Section 5.

2 Problem Statement

Given is a random vector $\underline{x} \in \mathbb{R}^{\mathrm{N}}$ with probability density function $\tilde{f}(\underline{x})$. This density function is assumed to be represented as a Gaussian mixture

$$\tilde{f}(\underline{x}) = \underline{\tilde{\alpha}}^{\mathrm{T}} \cdot \underline{f}(\underline{x}) = \sum_{i=1}^{L} \tilde{\alpha}_i \cdot \mathcal{N}\left(\underline{x}; \underline{\tilde{\mu}}_i, \tilde{\mathbf{C}}_i\right), \tag{1}$$

with $\underline{\tilde{\alpha}} = \left[\tilde{\alpha}_1 \ \ldots \ \tilde{\alpha}_L\right]^{\mathrm{T}}$ and $\underline{f}(\underline{x}) = \left[\mathcal{N}\left(\underline{x}; \underline{\tilde{\mu}}_1, \tilde{\mathbf{C}}_1\right) \ \ldots \ \mathcal{N}\left(\underline{x}; \underline{\tilde{\mu}}_L, \tilde{\mathbf{C}}_L\right)\right]^{\mathrm{T}}$. Here, L is the number of mixture components, $\tilde{\alpha}_i \geq 0$, $i = 1 \ldots L$ are weights that sum up to one, and $\mathcal{N}\left(\underline{x}; \underline{\mu}, \mathbf{C}\right)$ is a Gaussian density with mean vector $\underline{\mu}$ and covariance matrix $\mathbf{C}$.

In typical fusion tasks such as recursive filtering, the number of mixture components L grows exponentially with the number of processing steps. To keep this growth bounded, it is necessary to compute a reduced Gaussian mixture with the number of components being significantly lower than L. Additionally, the deviation between the reduced mixture and the original mixture has to be as small as possible.

To solve this apparent conflict of goals, one can make use of the following observation: Although the number of components is large, the shape of the density functions in typical estimation tasks is often rather simple, i.e., the number of modes is low and the smoothness is high. Especially in recursive filtering tasks, the density function of the hidden state often becomes unimodal or even Gaussian-like after a transient phase. Thus, a Gaussian mixture with a considerably smaller number of components can typically be found without causing a strong deviation from the original mixture.

In the following, reducing the number of components is achieved by adapting the weights $\tilde{\alpha}_j$ only, while the remaining parameters of the mixture, i.e., mean vectors and covariance matrices, remain untouched. The Gaussian mixture to be adapted is given by

$$f_{\underline{\alpha}}(\underline{x}) = \underline{\alpha}^{\mathrm{T}} \cdot \underline{f}(\underline{x}) = \sum_{j=1}^{L} \alpha_j \cdot \mathcal{N}\left(\underline{x}; \underline{\tilde{\mu}}_j, \tilde{\mathbf{C}}_j\right) , \tag{2}$$

with the vector of weights $\underline{\alpha} = \left[\alpha_1 \ \dots \ \alpha_L\right]^{\mathrm{T}}$. The proposed mixture reduction method will assign weights close to zero to redundant mixture components. These components can be easily removed in a subsequent processing step. The remaining components, however, compensate this loss in representing the shape of $\tilde{f}(\underline{x})$ just by weight adaption. For the sake of brevity and clarity, only univariate and bivariate mixtures are considered from now on.

3 Superficial Gaussian Mixture Reduction

The Gaussian mixture reduction problem is formulated as a weight optimization problem

$$\begin{aligned} \min_{\underline{\alpha}} \quad & D\left(\tilde{f}, f_{\underline{\alpha}}\right) + \lambda \cdot R\left(f_{\underline{\alpha}}\right) \\ \text{s.t.} \quad & \underline{1}^{\mathrm{T}} \cdot \underline{\alpha} = 1 \, , \\ & \underline{0} \preceq \underline{\alpha} \, , \\ & \underline{0} = \textstyle\sum_{i=1}^{L} \underline{\tilde{\mu}}_i (\alpha_i - \tilde{\alpha}_i) \, , \end{aligned} \tag{3}$$

where the parameter λ governs the trade-off between a distance $D\left(\tilde{f}, f_{\underline{\alpha}}\right)$ of the true density $\tilde{f}$ to its reduction $f_{\underline{\alpha}}$ and a roughness penalty $R\left(f_{\underline{\alpha}}\right)$, measuring the curvature of $f_{\underline{\alpha}}$. The constraints in the optimization problem assert the integration of the probability mass to one, the positivity of the density, and that $\tilde{f}$ and $f_{\underline{\alpha}}$ have identical means.

3.1 Distance Measure

A key requirement for any reduction algorithm is that the distance of the reduced $f_{\underline{\alpha}}$ to the true density $\tilde{f}$ is small over the entire state space. Therefore, the ISD is employed and reformulated as a function of the mixture weights $\underline{\alpha}$

$$D\big(\tilde{f}, f_{\underline{\alpha}}\big) = \tfrac{1}{2} \int_{\mathbb{R}^2} \big(\tilde{f}(\underline{x}) - f_{\underline{\alpha}}(\underline{x})\big)^2 \,\mathrm{d}\underline{x} = \underline{\alpha}^{\mathrm{T}}\, \mathbf{D}\, \underline{\alpha} - 2\, \underline{d}^{\mathrm{T}} \underline{\alpha} + c\,, \tag{4}$$

with matrix $\mathbf{D} = \int \underline{f}(\underline{x}) \cdot \underline{f}(\underline{x})^{\mathrm{T}} \,\mathrm{d}\underline{x}$ corresponding to the self-similarity of $f_{\underline{\alpha}}$, the vector $\underline{d}^{\mathrm{T}} = \underline{\tilde{\alpha}}^{\mathrm{T}}\mathbf{D}$ encoding the cross-similarity between $f_{\underline{\alpha}}$ and $\tilde{f}$, as well as constant $c = \underline{\tilde{\alpha}}^{\mathrm{T}}\,\mathbf{D}\,\underline{\tilde{\alpha}}$ corresponding to the self-similarity of $\tilde{f}$. The quadratic form in (4) is obtained by expanding the binomial, exploiting the linearity, and solving the obtained integrals. Note that the same vector of Gaussians $\underline{f}(x)$ is used for both $f_{\underline{\alpha}}$ and $\tilde{f}$.

3.2 Upper bound of Curvature

The roughness of $f_{\underline{\alpha}}$ is interpreted as the curvature κ of the probability density function's surface. Since the curvature (see e.g. [4]) is signed and a function of the position on the surface, a quantification in terms of the integral squared curvature (ISC) is sought. For the sake of brevity, the notation $f_m \triangleq \frac{\partial}{\partial m} f(\underline{x})$ and $f_{xy} := \frac{\partial^2}{\partial x \partial y} f(\underline{x})$ is used for the derivatives at point $\underline{x}$. For a Gaussian mixture density $f(\underline{x}) = \underline{\alpha}^{\mathrm{T}} \cdot \underline{f}(\underline{x})$, $f_m^{(i)} = \frac{\partial}{\partial m} \mathcal{N}\big(\underline{x}; \underline{\mu}_i, \mathbf{C}_i\big)$ denotes the i-th component's partial derivative w.r.t. m and $\underline{f}_m$ the vector of all components' partial derivatives. The key idea is to derive an (approximate) upper bound of the squared curvature of the mixture density function. The derivation is based on the pointwise squared curvature $\kappa(\underline{x})$ for a probability density function f, for which an upper bound $\breve{\kappa}(\underline{x})$ is determined, integrated over the entire domain of $\underline{x}$, i.e., $R_{\underline{x}} = \int \breve{\kappa}(\underline{x})^2 \,\mathrm{d}\underline{x}$. For the 1D and 2D case, the upper bounds are

$$\breve{\kappa}(x)^2 \triangleq \Big(\underline{\alpha}^{\mathrm{T}} \underline{f}_{xx}\Big)^2, \qquad \breve{\kappa}(\underline{x})^2 \triangleq \Big(\underline{\alpha}^{\mathrm{T}} \underline{f}_{xx} - 2\, \underline{\alpha}^{\mathrm{T}} \underline{f}_{x}\, \underline{\alpha}^{\mathrm{T}} \underline{f}_{y}\, \underline{\alpha}^{\mathrm{T}} \underline{f}_{xy} + \underline{\alpha}^{\mathrm{T}} \underline{f}_{yy}\Big)^2 .$$

For the weight optimization, the upper bound of the curvature is formulated as a quadratic form $R(\underline{\alpha}) = \underline{\alpha}^{\mathrm{T}}\, \mathbf{R}^{\underline{x}}\, \underline{\alpha}$. The elements of $\mathbf{R}^{\underline{x}}$ may be obtained as follows.

For the 1D case, one may simplify the upper bound of the squared curvature [11]

$$\int_{\mathbb{R}} \left(\underline{\alpha}^{\mathrm{T}} \underline{f}_{xx}\right)^2 \mathrm{d}x = \int_{\mathbb{R}} \underline{\alpha}^{\mathrm{T}} \underline{f}_{xx} \underline{f}_{xx}^{\mathrm{T}} \underline{\alpha} \, \mathrm{d}x$$

and use the linearity of the integral to obtain the expression for the elements of $\mathbf{R}^{\underline{x}}$

$$\mathbf{R}^{\underline{x}}_{ij} = \int_{\mathbb{R}} f^{(i)}_{xx} f^{(j)}_{xx} \, \mathrm{d}x \, .$$

For the 2D case, the following approximation is used

$$\left(\underline{\alpha}^{\mathrm{T}} \underline{f}_{xx} - 2\,\underline{\alpha}^{\mathrm{T}} \underline{f}_{x} \underline{\alpha}^{\mathrm{T}} \underline{f}_{y} \underline{\alpha}^{\mathrm{T}} \underline{f}_{xy} + \underline{\alpha}^{\mathrm{T}} \underline{f}_{yy}\right)^2 \approx c \cdot \left(\underline{\alpha}^{\mathrm{T}} \left[\underline{f}_{xx} - 2\,\underline{f}_{x} \underline{f}_{y}^{\mathrm{T}} \underline{f}_{xy} + \underline{f}_{yy}\right]\right)^2 ,$$

where c is neglected, as it is considered independent of $\underline{\alpha}$. One obtains

$$\mathbf{R}^{\underline{x}}_{ij} = \int_{\mathbb{R}^2} \left(f^{(i)}_{xx} - 2\, f^{(i)}_{x} f^{(i)}_{y} f^{(i)}_{xy} + f^{(i)}_{yy}\right) \cdot \left(f^{(j)}_{xx} - 2\, f^{(j)}_{x} f^{(j)}_{y} f^{(j)}_{xy} + f^{(j)}_{yy}\right) \mathrm{d}\underline{x} \, . \tag{5}$$

For the 1D curvature, the $\mathbf{R}^{\underline{x}}_{ij}$ may be calculated in closed form. Note for a 2D probability density function, the curvature is not unique, as it is calculated from the minimum and maximum curvature in the principal directions at each point $\underline{x}$, which may be multiplied (Gaussian curvature) or averaged (mean curvature) [4]. The above upper bound of the integral squared mean curvature was obtained by dropping the denominator and positive summands. For arbitrary Gaussian mixture densities, the terms in (5) may only be calculated numerically or need to be approximated further. In the following algorithm, the property that the exact upper bound of the 1D curvature and the approximate upper bound of the 2D curvature of the probability density functions as well as the distance measure may be represented as quadratic forms will be exploited.

3.3 Algorithm

The overall algorithm of the SGMR comprises three parts: the *pre-processing* of the components of the quadratic forms and the hyperparameter, the weight optimization by the solution of (3) in form of a *quadratic program (QP)*, and a fast *post-optimization* of the already reduced set of weights from (3). The *pre-processing* consists of calculating the matrix $\mathbf{D}$ and vector $\underline{d}$ corresponding to the distance of the densities in $D(\tilde{f}, f_{\underline{\alpha}})$. The matrix $\mathbf{R}$ describing the curvature of the surface has to be calculated depending on the type of density. Subsequently, the QP may be composed, i.e.,

$$\underline{\alpha}^{\mathrm{T}} \mathbf{D} \underline{\alpha} - 2 \underline{d}^{\mathrm{T}} \underline{\alpha} + \lambda \underline{\alpha}^{\mathrm{T}} \mathbf{R} \underline{\alpha} = \underline{\alpha}^{\mathrm{T}} \mathbf{Q} \underline{\alpha} - \underline{q}^{\mathrm{T}} \underline{\alpha} \,,$$

with

$$\mathbf{Q} \triangleq \mathbf{D} + \lambda \mathbf{R} \,, \quad \underline{q} \triangleq 2 \underline{d} \,.$$

The matrix **Q** of the quadratic form is symmetric. Furthermore, the matrices **D** and **R** are positive semi-definite since the inequalities

$$\underline{\alpha}^{\mathrm{T}} \mathbf{D} \underline{\alpha} = \underline{\alpha}^{\mathrm{T}} \left(\int \underline{f}\, \underline{f}^{\mathrm{T}} \,\mathrm{d}\underline{x} \right) \underline{\alpha} = \int \left(\underline{\alpha}^{\mathrm{T}} \underline{f} \right)^2 \mathrm{d}\underline{x} \geq 0 \tag{6}$$

and

$$\underline{\alpha}^{\mathrm{T}} \mathbf{R} \underline{\alpha} = \underline{\alpha}^{\mathrm{T}} \left(\int T(\underline{f})\, T(\underline{f})^{\mathrm{T}} \,\mathrm{d}\underline{x} \right) \underline{\alpha} = \int \left[\underline{\alpha}^{\mathrm{T}} T(\underline{f}) \right]^2 \mathrm{d}\underline{x} \geq 0 \tag{7}$$

hold for all $\underline{\alpha} \in \mathbb{R}^L$. The differential operator T depends on the respective upper bound $\breve{\kappa}(\underline{x})^2$ used. Thus, **Q** is positive semi-definite and the optimization problem is a convex QP. Using the mass and positivity constraints, one obtains a QP that may be solved by any standard solver. For the experiments in this paper, the freely available `CVX` optimization library [6] was used.

The purpose of the *post-optimization* is an adaptation of the already reduced weights components $\underline{\alpha}^+$ aimed at improving the accuracy, by neglecting the curvature and only minimizing the ISD w.r.t. $\underline{\alpha}^+$, where, e.g., $\alpha_i^+ \geq \epsilon$ with $\epsilon = 1e^{-4}$, in the experiments presented in Sec. 5. The resulting QP is

$$\begin{aligned} \min_{\underline{\alpha}^+} \quad & \left(\underline{\alpha}^+\right)^{\mathrm{T}} \left(\mathbf{D}^+\right) \left(\underline{\alpha}^+\right) - 2 \left(\underline{d}^+\right)^{\mathrm{T}} \left(\underline{\alpha}^+\right) \\ \text{s.t.} \quad & \underline{0} \preceq \underline{\alpha}^+ \,, \\ & \underline{1}^{\mathrm{T}} \underline{\alpha}^+ = 1 \,, \\ & \textstyle\sum_i \alpha_i^+ \underline{\mu}_i^+ = \sum_i \tilde{\alpha}_i \underline{\tilde{\mu}}_i \,. \end{aligned}$$

This optimization problem for the reduced weights consists of the quadratic form of the ISD as a target function and the positivity, mass, and mean constraint w.r.t. to the reduced mixture's components. This QP may be solved with any standard solver. The obtained weights $\underline{\alpha}^*$ will be reduced again by removing components with almost zero weights, i.e., $\alpha_i < \epsilon$. The overall algorithm is

Algorithm 1 Superficial Gaussian Mixture Reduction

```
 1: Input: $\tilde{f}$
    ▷ Preprocessing
 2: Calculate distance terms $\mathbf{D}, \underline{d}$, roughness penalty matrix $\mathbf{R}$, and $\lambda$
 3: Compose Quadratic Program QP($\mathbf{D}, \underline{d}, \mathbf{R}, \lambda$)
 4: $\underline{\alpha}^{+} \leftarrow$ REDUCE( SOLVE QP($\mathbf{D}, \underline{d}, \mathbf{R}, \lambda$) )      ▷ Quadratic Program
 5: $\underline{\alpha}^{*} \leftarrow$ REDUCE( OPTIMIZEWEIGHTS($\mathbf{D}^{+}, \underline{d}^{+}, \underline{\alpha}^{+}$) )    ▷ Post-Optimization
 6: function OPTIMIZEWEIGHTS($\mathbf{D}^{+}, \underline{d}^{+}, \alpha^{+}$)
 7:     Compose Quadratic Program QP($\mathbf{D}^{+}, \underline{d}^{+}$)
 8:     $\underline{\alpha}^{++} \leftarrow$ REDUCE( SOLVE QP($\mathbf{D}^{+}, \underline{d}^{+}$) )
 9: end function
10: function REDUCE($\underline{\alpha}'$)
11:     $\underline{\alpha}'' \leftarrow \underline{\alpha}' \geq \epsilon$
12: end function
13: Output: $f \sim \text{GMM}\left\{\underline{\alpha}^{*}, \left\{\underline{\mu}_i, \mathbf{C}_i\right\}^{*}\right\}$      ▷ $L^{*} \ll L$
```

given in Algorithm 1. Note, that the hyperparameter λ in Algorithm 1, which governs the trade-off between the distance $D(\tilde{f}, f_{\underline{\alpha}})$ and the curvature term $R(f_{\underline{\alpha}})$, needs to be determined by means of generic model selection algorithms, cf. [15]. For small values of λ, the ISD will be weighted relatively higher than the curvature. This results in more components in the reduced mixture f and less approximation error. For large values of λ, the curvature will be weighted higher enforcing more reduction and approximation error.

4 Limitations

The computational complexity for both optimization steps is polynomial in the number of mixture components. The cost for the post-optimization is smaller, as only components with $\alpha_i^{+} \geq \epsilon$ are considered, which may be significantly fewer. The underlying major assumption of this approach is that $\tilde{f}$ consists of a large number of components. As will be shown in the experiments, the quality of the results depends on the number of components to be reduced. Therefore, the reduction of $\tilde{f}$ with very few components–actually not needing a reduction–will result in a low quality reduction $f_{\underline{\alpha}}$ as only $\underline{\alpha}$ is optimized, but no mixture means or covariances. The initially given set of means is identical to the set the means used in $f_{\underline{\alpha}}$. Note that even though this reduces the theoretical reduction capability, any $\tilde{f}$ with an insufficient number of components will not require a reduction at all.

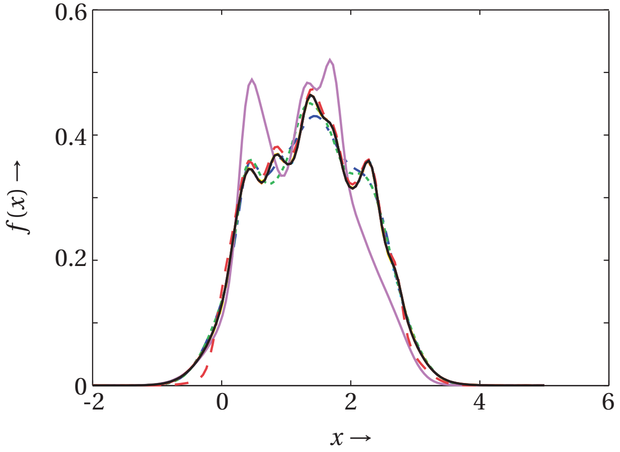
5 Experiments

In the following, SGMR is compared to six established reduction methods: The simplest reduction is a *pruning* of all but the components with the highest weights, [3]. Top-down and local reduction algorithm, denoted by *West*, employ the Mahalanobis distance and merge two components at each reduction step [17]. *Salmond*'s approach is similar to West's approach, but merges complete clusters of mixture components of size two and more [13]. A top-down and global reduction algorithm based on the ISD (4) is proposed by *Williams* [18], where two components are merged per step and irrelevant components are additionally deleted. *Runnalls*' algorithm offers a compromise of local and global reduction algorithms, as it considers a localized upper bound of the (global) Kullback-Leibler divergence [12]. Merging is performed for two components. *PGMR* is a bottom-up approach employing the ISD (4) [9].

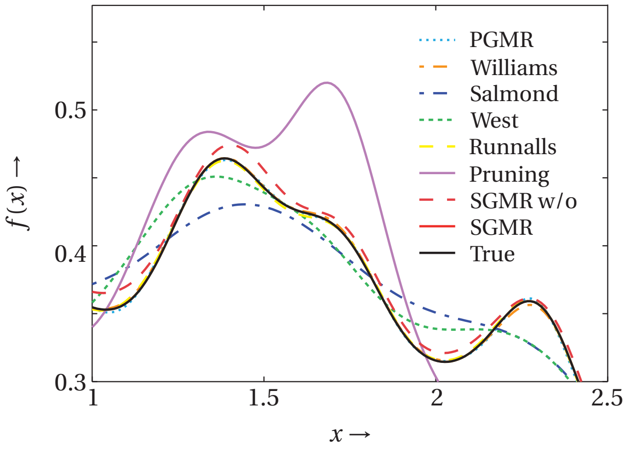
For SGMR, two variants are considered, one with post-optimization and one without post-optimization. The first four top-down approaches and the pruning method require a user-defined threshold on the number of components to which the given Gaussian mixture has to be reduced. Since SGMR reduces a Gaussian mixture in a completely different fashion and thus, to ensure a fair comparison, the number of components resulting from SGMR with post-optimization is used as threshold for these approaches. In order to quantify the reduction error, the normalized ISD

$$\overline{D}\big(\tilde{f}, f_{\underline{\alpha}}\big) = \sqrt{\frac{\int_{\mathbb{R}^N} \big(\tilde{f}(\underline{x}) - f_{\underline{\alpha}}(\underline{x})\big)^2 \, \mathrm{d}\underline{x}}{\int_{\mathbb{R}^N} \tilde{f}(\underline{x})^2 \, \mathrm{d}\underline{x} + \int_{\mathbb{R}^N} \tilde{f}_{\underline{\alpha}}(\underline{x})^2 \, \mathrm{d}\underline{x}}} \in [0,1] \tag{8}$$

is employed [7]. It ranges between zero, which is the case if $\tilde{f}(\underline{x})$ and $f_{\underline{\alpha}}(\underline{x})$ are identical, and one, when both mixtures are absolutely non-overlapping. The algorithms are implemented in Matlab 7.8.0 (R2009a) and run on an office PC (Intel Core2 Duo P9600).



(a) True Gaussian mixture (black) and the reduced Gaussian mixture for the different reduction algorithms.



(b) Zoom of the Gaussian mixture in (a) and the respective reduced mixtures.

Figure 2: Result of several reduction algorithms for a exemplary Gaussian mixture. Note, that the results of PGMR, Williams, Runnalls, and SGMR are almost identical to the true Gaussian mixture.

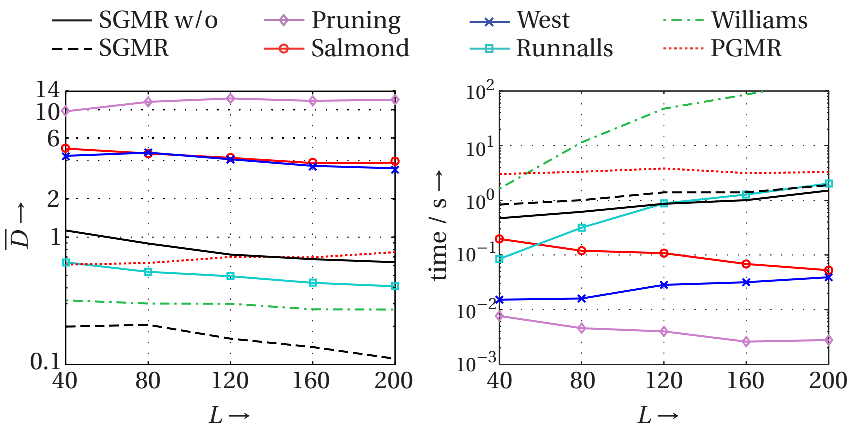
5.1 1D Experiment

At first, univariate Gaussian mixtures with $L \in \{40, 80, 120, 160, 200\}$ components are used for evaluation. The mixture parameters are drawn uniformly at random from the intervals $\tilde{\alpha} \in [0.05, 0.5]$, $\tilde{\mu} \in [0, 3]$, and $\tilde{\sigma} \in [0.09, 0.5]$. For each number of components L, 50 Monte Carlo simulation runs are performed. For SGMR, the hyperparameter λ is set to 500 and the deletion threshold ε is $1e^{-4}$. The maximum error threshold of PGMR is set to 1%.

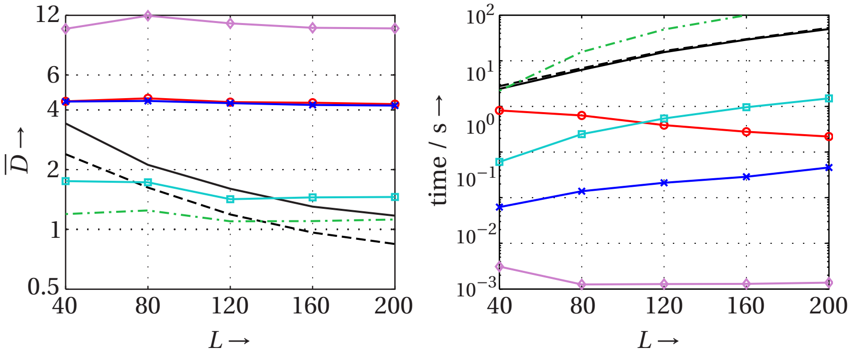
In Figure 3a, the average reduction errors and the average computation times for all L are shown. It can be seen that SGMR provides the lowest reduction error. Closest to SGMR is Williams' algorithm, but this algorithm clearly suffers from its high computational demand. Salmond's and West's methods perform similarly. Both are very fast, but their approximation quality is the worst except for pruning. In terms of the reduction error, the results of Runnalls' method are in between of SGMR with and without post-optimization. But for an increasing number of components L in the original mixture it becomes computationally more expensive than both SGMR methods. Overall, SGMR provides the best trade-off between reduction error and computation time.

The reduction performance of SGMR improves with a larger number of components in the original mixture. As listed in Table 1, SGMR reduces to about 50% if the number of components of the original mixture is $L = 40$, while for $L = 200$ only 22% of the components remain. Since SGMR merely adapts the weights $\underline{\tilde{\alpha}}$, a larger number of components is advantageous for SGMR for a better exploitation of redundancies. This leads to a stronger reduction by a simultaneously lower reduction error. Furthermore, the comparison between SGMR and SGMR without post-optimization shows that the post-optimization always lowers both the reduction error and the number of components.

In Figure 2, the reduction results for an exemplary Gaussian mixture with $L = 120$ components is depicted. SGMR, PGMR, Runnalls', and Williams' algorithm are capable of almost exactly capturing the shape of the original mixture, while West's and Salmond's algorithm show the tendency to oversmooth modes. The result of SGMR without post-optimization is in between. There is an obvious deviation to the original mixture, but the shape–especially the single modes–are captured very well. The inferior results of pruning can be explained by its simplicity. Components are only deleted on the basis of the value of their weights. No distance measure is used to quantify the loss of a component.



(a) Results for 1D Gaussian mixtures.



(b) Results for 2D Gaussian mixtures.

Figure 3: Reduction error (left) and runtime (right) of different reduction algorithms for increasing number of components. The results are averages over 50 Monte Carlo runs. The average reduction error is multiplied by 100 for better readability.

For 1D mixtures, PGMR clearly is the best reduction algorithm. The reduction error is close to SGMR without post-optimization, but the number of components in the reduced mixture is significantly lower (see Table 1). However, a straightforward extension to multivariate mixtures is not possible as only axis-aligned Gaussian components can be utilized for representing the reduced mixture. For this reason, PGMR is not considered in the following 2D experiment.

5.2 2D Experiment

In this experiment, randomly generated bivariate Gaussian mixtures are considered. The weights $\underline{\tilde{\alpha}}$ and the elements c of the covariance matrices of the original mixture are drawn from the intervals $\tilde{\alpha} \in [0.05, 0.5]$ and $c \in [0.1, 1]$, respectively. 25% of the mean vectors are drawn from $\underline{\tilde{\mu}} \in [0, 0.75] \times [0, 1.5]$ and the remaining mean vectors are sampled from $\underline{\tilde{\mu}} \in [1.5, 3] \times [0, 1.5]$. Due to this placement of the Gaussian components, bimodality is forced in the true mixture. Again, 50 Monte Carlo simulation runs are performed for each number of components $L \in \{40, 80, 120, 160, 200\}$. The hyperparameter λ of SGMR is set to 0.04 and $\varepsilon = 1e^{-4}$.

Figure 3b gives the reduction error and the average computation time. In comparison to the 1D experiment, it becomes more obvious that SGMR is the ideal method for reducing a Gaussian mixture with a large number of components. From $L = 160$ on, SGMR outperforms all other algorithms with respect to the reduction error. Furthermore, the benefit of the post-optimization is more significant than in the 1D case. Besides a lower reduction error, the number of components can be reduced much stronger as shown in Table 1. This benefit comes with a low computational overhead.

In contrast to the 1D experiment, the computation time of SGMR now significantly increases with the number of components. Most of the time is used for calculating the roughness penalty matrix $\mathbf{R}$, which requires numerical integration in the 2D case. It is expected that an improved problem-adequate implementation of the numerical integration will reduce the computational demand drastically.

Table 1: Number of components in the reduced Gaussian mixture for different reduction algorithms. The results are averages over 50 Monte Carlo runs.

1D -Experiment

L	SGMR	PGMR	SGMR &
	w/o		all others
40	21.66	7.34	20.54
80	30.88	7.42	29.5
120	37.18	7.58	35.86
160	42.86	6.78	41.8
200	45.68	6.7	44.98

2D -Experiment

L	SGMR	SGMR &
	w/o	all others
40	18.92	16.68
80	26.76	22.88
120	40.32	33.32
160	52.72	42
200	61.8	43.16

6 Conclusion

In this paper, a curvature-based reduction algorithm for Gaussian mixtures was presented. The key idea is the formulation of the reduction problem as an optimization problem. The optimization balances the integral squared distance between true and reduced density with the reduction in approximate shape curvature. The arising problem is solved for the weights of the Gaussian mixture only, i.e., reduced components are assigned a weight of zero, allowing a formulation as a quadratic program. The main contributions are the alleviation of the model selection problem, as the number of components is chosen by the algorithm automatically and an easy as well as efficient implementation. The experiments show the high quality and low number of components of the approach's results.

As future work it remains to derive an analytic upper bound for the n-dimensional curvature and to improve the numerical calculations. For very large reduction problems, more efficient algorithms could be obtained from exploiting the locality of Gaussian mixtures.

References

[1] Daniel L. Alspach and Harold W. Sorenson. Nonlinear Bayesian Estimation using Gaussian Sum Approximation. *IEEE Transactions on Automatic Control*, 17(4):439–448, August 1972.

[2] Yaakov Bar-Shalom and Xiao-Rong Li. *Multitarget-multisensor Tracking: Principles and Techniques.* YBS Publishing, Storrs, CT, 1995.

[3] Samuel S. Blackman. *Multiple-Target Tracking with Radar Applications.* Norwood, MA: Artech House, 1986.

[4] Manfredo Do Carmo. *Differential Geometry of Curves and Surfaces.* Prentice-Hall, Englewood Cliffs, NJ, 1976.

[5] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active Learning with Statistical Models. *Journal of Artificial Intelligence Research*, 4:129–145, March 1996.

[6] Michael C. Grant and Stephen P. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors,

Recent Advances in Learning and Control, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

[7] Uwe D. Hanebeck, Kai Briechle, and Andreas Rauh. Progressive Bayes: A New Framework for Nonlinear State Estimation. In *Proceedings of SPIE, AeroSense Symposium*, volume 5099, pages 256–267, Orlando, Florida, May 2003.

[8] Marco Huber, Dietrich Brunn, and Uwe D. Hanebeck. Closed-Form Prediction of Nonlinear Dynamic Systems by Means of Gaussian Mixture Approximation of the Transition Density. In *Proceedings of the 2006 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 98–103, Heidelberg, Germany, September 2006.

[9] Marco F. Huber and Uwe D. Hanebeck. Progressive Gaussian Mixture Reduction. In *Proceedings of the 11th International Conference on Information Fusion (Fusion)*, pages 1–8, Cologne, Germany, July 2008.

[10] Vladimir Maz'ya and Gunther Schmidt. On approximate approximations using gaussian kernels. *IMA Journal of Numerical Analysis*, 16(1):13–29, 1996.

[11] Jim Ramsay and Bernard Silverman. *Functional Data Analysis*. Springer Series in Statistics. Springer, New York, Berlin, Heidelberg, 1997.

[12] Andrew R. Runnalls. Kullback-Leibler Approach to Gaussian Mixture Reduction. *IEEE Transactions on Aerospace and Electronic Systems*, 43(3):989–999, July 2007.

[13] David J. Salmond. Mixture reduction algorithms for target tracking in clutter. In *Proceedings of SPIE Signal and Data Processing of Small Targets*, volume 1305, pages 434–445, October 1990.

[14] Dennis Schieferdecker and Marco F. Huber. Gaussian Mixture Reduction via Clustering. In *Proceedings of the 12th International Conference on Information Fusion (Fusion)*, pages 1536–1543, Seattle, Washington, July 2009.

[15] Bernhard Schölkopf and Alexander Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive computation and machine learning series. MIT Press, Cambridge, Massachusetts, 2002.

[16] Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis.* Monographs on Statistics and Applied Probability ; 26. CRC Press, Boca Raton, 1998.

[17] Mike West. Approximating Posterior Distributions by Mixtures. *Journal of the Royal Statistical Society: Series B*, 55(2):409–422, 1993.

[18] Jason L. Williams and Peter S. Maybeck. Cost-Function-Based Gaussian Mixture Reduction for Target Tracking. In *Proceedings of the Sixth International Conference of Information Fusion (Fusion)*, volume 2, pages 1047–1054, Cairns, Australia, July 2003.

Paper H

Analytic Moment-based Gaussian Process Filtering

Authors: Marc P. Deisenroth, Marco F. Huber, and Uwe D. Hanebeck

Edited version of the paper:

M. P. Deisenroth, M. F. Huber, and U. D. Hanebeck. Analytic Moment-based Gaussian Process Filtering. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, Montreal, Canada, June 2009.

Analytic Moment-based Gaussian Process Filtering

Marc P. Deisenroth*, Marco F. Huber**, and Uwe D. Hanebeck**

*Department of Engineering
University of Cambridge
Cambridge, UK
`mpd37@cam.ac.uk`

** Intelligent Sensor-Actuator-Systems
Laboratory (ISAS)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
`{marco.huber|uwe.hanebeck}@ieee.org`

Abstract

We propose an analytic moment-based filter for nonlinear stochastic dynamic systems modeled by Gaussian processes. Exact expressions for the expected value and the covariance matrix are provided for both the prediction step and the filter step, where an additional Gaussian assumption is exploited in the latter case. Our filter does not require further approximations. In particular, it avoids finite-sample approximations. We compare the filter to a variety of Gaussian filters, that is, the EKF, the UKF, and the recent GP-UKF proposed by [7].

1 Introduction

Recursively estimating the internal state of a nonlinear dynamic system from noisy observations is a common problem in many technical applications, for instance, in sensor networks, robotics, or signal processing. Exact Bayesian solutions in closed form, however, can be found only in a few special cases. For example, for linear Gaussian systems, the Kalman filter [4] is exact.

For most nonlinear cases, approximate methods are required to obtain efficient analytic/closed-form solutions. A variety of approximate Gaussian filters has been proposed in the past. For example, the *Extended Kalman Filter (EKF)* linearizes the transition and measurement functions by means of a Taylor series expansion and applies the Kalman filter to propagate full densities through them [15]. Instead of approximating functions, the *Unscented Kalman Filter*

(UKF) by [3] uses a deterministic sampling approach to approximate distributions, while using the original nonlinear functions to propagate them. This approach is considered equivalent to stochastic linearization [9].

Both the EKF and the UKF employ a known parametric model of the transition dynamics and the measurement function. However, lack of modeling accuracy as well as difficulties in the identification of the noise and the model parameters are typically ignored. Instead of a parametric description, [7] and [6] derive the GP-EKF and the GP-UKF by incorporating probabilistic non-parametric Gaussian process (GP) models of the transition dynamics and the measurement function into the EKF and UKF. Model uncertainty can explicitly be incorporated into the prediction and the filtering processes, which is usually not the case for filtering approaches based on a parametric model. Moreover, they train the GP models offline using ground truth of the hidden states.

In this paper, we derive a Gaussian filter algorithm for nonlinear dynamic systems, where the transition dynamics and the observation map are described by GP models. In contrast to finite-sample approximations (UKF, GP-UKF) of the prior and the predictive distribution, we propagate full densities by exploiting specific properties of GP models. Furthermore, we approximate the predictive distribution by a Gaussian with the exact mean and the exact covariance matrix, which can be computed analytically using results from [12]. This approximation, on which our filter is based, is known as *moment matching*. Hence, the proposed filter, which we call GP-ADF, is an efficient form of an *Assumed Density Filter* (ADF) [10].

The paper is organized as follows: In Section 2, the models under consideration are reviewed and the prediction and filtering problems are stated. A survey of related work is given in Section 3. In Section 4, we provide background on prediction with GP models. The GP-ADF itself is derived in Section 5. Simulation results are presented in Section 6. In Section 7, we discuss properties of the filter algorithm. Section 8 summarizes the paper and gives a survey of future work.

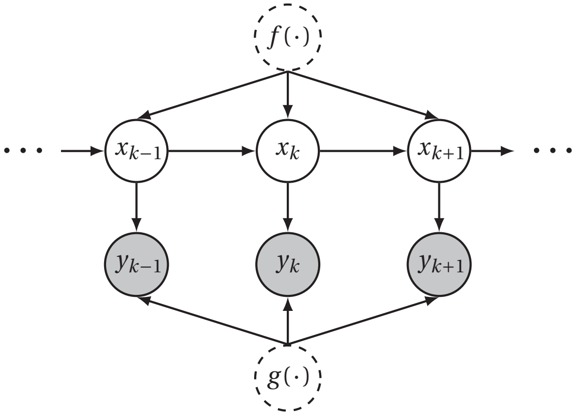


Figure 1: Graphical model of a nonlinear dynamic system. The shaded nodes $\underline{y}_i$ are observed variables, the other nodes are latent variables. The dependencies between variables are given by the arrows. The dashed nodes represent functions f and g, which can either be observed or latent depending on the model used.

2 Model and Problem Statement

We consider discrete-time dynamic systems with transition dynamics given by

$$\underline{x}_k = f\left(\underline{x}_{k-1}\right) + \underline{w}\,, \tag{1}$$

where f is a possibly nonlinear function and $\underline{w} \sim \mathcal{N}\left(\underline{0}, \mathbf{C}_w\right)$ is white, additive Gaussian system noise with uncorrelated dimensions. The D-dimensional continuous-valued state is denoted by $\underline{x}$, and k is a discrete time index. Furthermore, we consider observations/measurements

$$\underline{y}_k = g\left(\underline{x}_k\right) + \underline{v}\,, \tag{2}$$

where g is a (non)linear function, $\underline{y}_k$ is the E-dimensional observation, and $\underline{v} \sim \mathcal{N}\left(\underline{0}, \mathbf{C}_v\right)$ is white, additive Gaussian measurement noise with uncorrelated dimensions.

Figure 1 is a graphical model of the considered nonlinear dynamic system. We included dashed "function nodes" for f and g. The function node is shaded if and only if the function is explicitly known.

We assume a prior on $\underline{x}_0$ and aim to determine probability distributions of the hidden state $\underline{x}_k$ based on all observations $\underline{y}_{1:k}$. We distinguish between prediction (moving from $\underline{x}_{k-1}$ to $\underline{x}_k$) and filtering (going from $\underline{y}_k$ to $\underline{x}_k$). Typically, prediction and filtering alternate.

Prediction Step

When we predict, we determine the distribution $p\left(\underline{x}_k | \underline{y}_{1:k-1}\right)$ of the hidden state $\underline{x}_k$, where the result $p\left(\underline{x}_{k-1} | \underline{y}_{1:k-1}\right)$ of the previous filter update serves as the prior. Bayes' law yields

$$p\left(\underline{x}_k | \underline{y}_{1:k-1}\right) = \int p(\underline{x}_k | \underline{x}_{k-1}) p\left(\underline{x}_{k-1} | \underline{y}_{1:k-1}\right) \mathrm{d}\underline{x}_{k-1} \tag{3}$$

by averaging over $\underline{x}_{k-1}$. Often, the involved integral and the multiplication cannot be solved analytically and require approximate methods.

Filter Update

The filter update determines the distribution $p\left(\underline{x}_k | \underline{y}_{1:k}\right)$ of the hidden state $\underline{x}_k$ based on collected observations from all previous and the current time steps. Bayes' law yields the *filter update*

$$p\left(\underline{x}_k | \underline{y}_{1:k}\right) = \frac{p\left(\underline{y}_k | \underline{x}_k\right) p\left(\underline{x}_k | \underline{y}_{1:k-1}\right)}{p\left(\underline{y}_k | \underline{y}_{1:k-1}\right)} \,. \tag{4}$$

The likelihood $p\left(\underline{y}_k | \underline{x}_k\right)$ is defined through the measurement equation (2), the prior $p\left(\underline{x}_k | \underline{y}_{1:k-1}\right)$ is the result of the preceding prediction step (3). Often, the filter update (4) does not admit a closed-form solution since the integral in the normalization constant

$$p\left(\underline{y}_k | \underline{y}_{1:k-1}\right) = \int p\left(\underline{y}_k | \underline{x}_k\right) p\left(\underline{x}_k | \underline{y}_{1:k-1}\right) \mathrm{d}\underline{x}_k$$

and the density multiplication in the numerator in equation (4) cannot be computed exactly.

3 Related Work

Table 1 classifies the Gaussian filter methods discussed in this paper. We present density representation against knowledge of the parameterization of the transition dynamics f and the observation function g.

The UKF by [3] deterministically chooses *sigma points* that capture the moments of the state distribution and maps them using a known parameterization of the original nonlinear functions f and g, respectively. The transformed sigma points provide a *finite-sample approximation* of the true predictive distribution. The UKF is not moment preserving.

[7] and [6] propose GPs to model the transition and observation functions f and g. GPs are incorporated into standard filters, such as the UKF. The resulting GP-UKF maps the UKF sigma points through the GP models instead of the parametric functions f and g. Like in the UKF, all considered distributions are described by a finite number of samples and the GP-UKF is not moment preserving. In the limit of perfect GP models, that is, the posterior mean functions match the latent functions f and g and the posterior uncertainty is zero, both the UKF and the GP-UKF are equivalent.

Like [7], we utilize GPs to model f and g. In contrast to both the UKF and the GP-UKF, our proposed GP-ADF does not propagate samples from a Gaussian, but the full Gaussian *density*. Our GP-ADF heavily exploits the fact that the true moments of the GP predictive distribution can be computed in closed form. The predictive distribution is approximated by a Gaussian with the exact predictive mean and the exact predictive covariance (moment matching). Therefore, GP-ADF is a form of Assumed Density Filtering (ADF), which has previously been introduced by [10], [1], and [11]. Furthermore, to compute the first two predictive moments, GP-ADF takes the uncertainty about the latent functions f,g into account. GP-ADF is moment preserving.

Table 1: Classification of Gaussian filter methods.

	samples	full density
f,g : known	UKF	EKF
f,g : unknown	GP-UKF	GP-ADF

The UKF propagates samples through known or directly accessible functions, that is, the nodes for f and g in Figure 1 are shaded. A classical ADF and the EKF propagate entire densities, but they also require known functions f and g. GP-UKF and GP-ADF are based on probabilistic models of the latent functions. Hence, the nodes f,g in Figure 1 are unshaded. The filters differ in the propagation method: GP-UKF propagates a finite-sample approximation of a Gaussian, whereas GP-ADF propagates the full Gaussian.

[2] discuss the EKF for nonlinear dynamic systems, where the transition dynamics and the measurement function are modeled by a radial basis function network, a parametric approximation with limited expressiveness.

4 Gaussian Processes

Following the book by [14], we briefly introduce the notation and standard prediction models for Gaussian processes, which are used to infer a latent function h from (noisy) observations $y_i = h(\underline{x}_i)+\epsilon$, $\epsilon \sim \mathcal{N}(0,\sigma_\epsilon^2)$. A GP is completely specified by a mean function $m(\cdot)$ and a positive semidefinite covariance function $k(\cdot,\cdot)$, also called a *kernel*. We write $h \sim \mathcal{GP}$ if the latent function h is GP distributed. Throughout this paper, we consider the squared exponential (SE) kernel

$$k(\underline{x},\underline{x}') = \alpha^2 \exp\big(-\tfrac{1}{2}(\underline{x}-\underline{x}')^{\mathrm{T}}\Lambda^{-1}(\underline{x}-\underline{x}')\big)\,, \tag{5}$$

where Λ is a diagonal matrix of the characteristic length-scales of the SE kernel, and α^2 is the variance of the latent function h. The posterior predictive distribution of the function value $h_* = h(\underline{x}_*)$ for an arbitrary test input $\underline{x}_*$ is Gaussian with mean and variance

$$m_h(\underline{x}_*) = \mathrm{E}_h\{h_*\} = \underline{k}_*^{\mathrm{T}}\big(\mathbf{K}+\sigma_\epsilon^2\mathbf{I}\big)^{-1}\underline{y} = \underline{k}_*^{\mathrm{T}}\underline{\beta}\,, \tag{6}$$

$$\sigma_h^2(\underline{x}_*) = \mathrm{var}_h\{h_*\} = k_{**} - \underline{k}_*^{\mathrm{T}}\big(\mathbf{K}+\sigma_\epsilon^2\mathbf{I}\big)^{-1}\underline{k}_*\,, \tag{7}$$

respectively, with $\underline{k}_* \triangleq k\big(\mathbf{X},\underline{x}_*\big)$, $k_{**} \triangleq k\big(\underline{x}_*,\underline{x}_*\big)$, $\underline{\beta} \triangleq \big(\mathbf{K}+\sigma_\epsilon^2\mathbf{I}\big)^{-1}\underline{y}$, and where $\mathbf{K}$ is the kernel matrix with $K_{ij} = k\big(\underline{x}_i,\underline{x}_j\big)$. Moreover, $\mathbf{X} = \big[\underline{x}_1,\ldots,\underline{x}_n\big]$ are the training inputs, and $\underline{y} = \big[y_1,\ldots,y_n\big]^{\mathrm{T}}$ are the corresponding training targets (observations).

4.1 Predictions for Uncertain Inputs

We review results by [13], [12], and [8] of how to predict with GPs when the test input $\underline{x}_*$ is uncertain, which means that it has a probability distribution.

Consider the problem of predicting a function value $h(\underline{x}_*)$ for an *uncertain* test input $\underline{x}_* \sim \mathcal{N}(\underline{\mu}, \mathbf{C})$, where $h \sim \mathcal{GP}$ with an SE kernel k_h. The prediction problem corresponds to seeking the distribution

$$p\big(\mathrm{E}\{h(\underline{x}_*)\}\big|\underline{\mu},\mathbf{C}\big) = \int p(h(\underline{x}_*)|\underline{x}_*)\, p\big(\underline{x}_*|\underline{\mu},\mathbf{C}\big)\,\mathrm{d}\underline{x}_* \,. \tag{8}$$

The mean and variance of the GP predictive distribution for $p\big(h(\underline{x}_*)|\underline{x}_*\big)$ are given in equations (6) and (7), respectively. For the SE kernel, we can compute the mean μ_* and the variance σ_*^2 of equation (8) in closed form. The mean μ_* is

$$\begin{aligned}\mu_* &= \mathrm{E}_{\underline{x}_*}\{\mathrm{E}_h\{h(\underline{x}_*)\}|\underline{\mu},\mathbf{C}\} \overset{(6)}{=} \mathrm{E}_{\underline{x}_*}\{m_h(\underline{x}_*)|\underline{\mu},\mathbf{C}\}\\ &= \int m_h(\underline{x}_*)\cdot\mathcal{N}\big(\underline{x}_*;\underline{\mu},\mathbf{C}\big)\,\mathrm{d}\underline{x}_* = \underline{\beta}^{\mathrm{T}}\underline{l}\end{aligned} \tag{9}$$

with $\underline{l} = [l_1,\ldots,l_n]^{\mathrm{T}}$, where

$$\begin{aligned}l_i = \int k_h\big(\underline{x}_i,\underline{x}_*\big)\,p\big(\underline{x}_*\big)\,\mathrm{d}\underline{x}_* &= \alpha^2\big|\mathbf{C}\Lambda^{-1}+\mathbf{I}\big|^{-\frac{1}{2}}\\ &\quad\times\exp\Big(-\tfrac{1}{2}\big(\underline{x}_i-\underline{\mu}\big)^{\mathrm{T}}(\mathbf{C}+\Lambda)^{-1}\big(\underline{x}_i-\underline{\mu}\big)\Big)\end{aligned}$$

is an expectation of $k_h\big(\underline{x}_i,\underline{x}_*\big)$ with respect to $\underline{x}_*$. Note that the predictive mean explicitly depends on the mean and covariance of the distribution of the input $\underline{x}_*$. The variance σ_*^2 of $p\big(\mathrm{E}\{h(\underline{x}_*)\}|\underline{\mu},\mathbf{C}\big)$ is

$$\begin{aligned}\sigma_*^2 &= \mathrm{E}_{\underline{x}_*}\{m_h(\underline{x}_*)^2\big|\underline{\mu},\mathbf{C}\} + \mathrm{E}_{\underline{x}_*}\{\sigma_h^2(\underline{x}_*)\big|\underline{\mu},\mathbf{C}\} - \mathrm{E}_{\underline{x}_*}\{m_h(\underline{x}_*)\big|\underline{\mu},\mathbf{C}\}^2\\ &= \underline{\beta}^{\mathrm{T}}\tilde{\mathbf{L}}\underline{\beta} + \alpha^2 - \mathrm{Tr}\Big(\big(\mathbf{K}+\sigma_\epsilon^2\mathbf{I}\big)^{-1}\tilde{\mathbf{L}}\Big) - \mu_*^2\,,\end{aligned} \tag{10}$$

where $\mathrm{Tr}(\cdot)$ is the trace and

$$\tilde{L}_{ij} = \frac{k_h\big(\underline{x}_i,\underline{\mu}\big)\,k_h\big(\underline{x}_j,\underline{\mu}\big)}{\big|2\mathbf{C}\Lambda^{-1}+\mathbf{I}\big|^{\frac{1}{2}}} \times \exp\Big(\big(\underline{\tilde{z}}_{ij}-\underline{\mu}\big)^{\mathrm{T}}\big(\mathbf{C}+\tfrac{1}{2}\Lambda\big)^{-1}\mathbf{C}\Lambda^{-1}\big(\underline{\tilde{z}}_{ij}-\underline{\mu}\big)\Big) \tag{11}$$

with $\underline{\tilde{z}}_{ij} \triangleq \frac{1}{2}(\underline{x}_i + \underline{x}_j)$. Like the predicted mean in equation (9), the predictive variance explicitly depends on the mean and the covariance matrix of the input distribution. We approximate the predictive distribution $p\big(\mathrm{E}\{h(\underline{x}_*)|\underline{\mu},\mathbf{C}\}\big)$ by a Gaussian $\mathcal{N}(\mu_*,\sigma_*^2)$ that exactly matches the predictive mean and variance.

4.2 Multivariate Predictions

We extend the previous results to the case of a latent function $h : \mathbf{R}^D \to \mathbf{R}^E$, $h \sim \mathcal{GP}$ with an SE kernel k_h. We train E GP models independently using the same training inputs $\mathbf{X}$, but different training targets $\underline{y}_a = [y_1^a,\ldots,y_n^a]^\mathrm{T}$, $a = 1,\ldots,E$. This model implies that any two target dimensions are conditionally independent given the input. Intuitively, different target dimensions can only "communicate" via the input.

For a *deterministically* given input $\underline{x}_*$, the mean and the variance of a predicted function value for each target dimension are given by equations (6) and (7), respectively. The predicted covariance matrix is diagonal since we assume that the predicted target dimensions are conditionally independent given the input.

For an *uncertain* input $\underline{x}_* \sim \mathcal{N}(\underline{\mu},\mathbf{C})$, the predictive mean $\underline{\mu}_*$ of $p\big(\mathrm{E}\{h(\underline{x}_*)\}|\underline{\mu},\mathbf{C}\big)$ is the collection of all E individual predicted means μ_*^a given by equation (9). The target dimensions, however, co-vary and the corresponding predictive covariance matrix

$$\mathbf{C}_*|\underline{\mu},\mathbf{C} = \begin{bmatrix} \mathrm{var}\{h_1^*|\underline{\mu},\mathbf{C}\} & \ldots & \mathrm{cov}\{h_1^*,h_E^*|\underline{\mu},\mathbf{C}\} \\ \vdots & \ddots & \vdots \\ \mathrm{cov}\{h_E^*,h_1^*|\underline{\mu},\mathbf{C}\} & \ldots & \mathrm{var}\{h_E^*|\underline{\mu},\mathbf{C}\} \end{bmatrix}$$

is no longer diagonal. The variances on the diagonal are the predictive variances of the individual target dimensions given by equation (10). The cross-covariances are given by

$$\mathrm{cov}\{h_a^*, h_b^*|\underline{\mu},\mathbf{C}\} = \mathrm{E}_{h,\underline{x}_*}\{h_a^* h_b^*|\underline{\mu},\mathbf{C}\} - \mu_*^a \mu_*^b\,,$$

where $a,b \in \{1,\ldots,E\}$ and $h_a^* \triangleq h_a(\underline{x}_*)$. We rewrite

$$\begin{aligned}\mathrm{E}_{h,\underline{x}_*}\{h_a^* h_b^*|\underline{\mu},\mathbf{C}\} &= \iint h_a^* h_b^* p(h_a,h_b|\underline{x}_*)p(\underline{x}_*)\,\mathrm{d}h\,\mathrm{d}\underline{x}_* \\ &\overset{(9)}{=} \int m_h^a(\underline{x}_*)m_h^b(\underline{x}_*)p(\underline{x}_*)\,\mathrm{d}\underline{x}_*\,.\end{aligned}$$

With $\underline{\beta}_a \triangleq \left(\mathbf{K}_a + \sigma_{\epsilon_a}^2 \mathbf{I}\right)^{-1} \underline{y}_a$ in equation (6), we obtain

$$\begin{aligned}
\mathrm{E}_{h,\underline{x}_*}\{h_a^* h_b^* | \underline{\mu}, \mathbf{C}\} &= \int m_h^a(\underline{x}_*) m_h^b(\underline{x}_*) p(\underline{x}_*) \,\mathrm{d}\underline{x}_* \\
&\overset{(6)}{=} \int k_h^a(\underline{x}_*, \mathbf{X}) \underline{\beta}_a \cdot k_h^b(\underline{x}_*, \mathbf{X}) \underline{\beta}_b \cdot p(\underline{x}_*) \,\mathrm{d}\underline{x}_* \\
&= \underline{\beta}_a^{\mathrm{T}} \underbrace{\int k_h^a(\mathbf{X}, \underline{x}_*)\, k_h^b(\underline{x}_*, \mathbf{X}) p(\underline{x}_*) \,\mathrm{d}\underline{x}_*}_{=:\mathbf{L}} \underline{\beta}_b \,.
\end{aligned}$$

Furthermore, with $\mathbf{R} \triangleq \left(\Lambda_a + \Lambda_b\right)^{-1} + \mathbf{C}$,

$$\begin{aligned}
L_{ij} &= \alpha_a^2 \alpha_b^2 \left|(\Lambda_a^{-1} + \Lambda_b^{-1})\mathbf{C} + \mathbf{I}\right|^{-\frac{1}{2}} \\
&\quad \times \exp\left(-\tfrac{1}{2}(\underline{x}_i - \underline{x}_j)^{\mathrm{T}} (\Lambda_a + \Lambda_b)^{-1} (\underline{x}_i - \underline{x}_j)\right) \\
&\quad \times \exp\left(-\tfrac{1}{2}(\underline{z}_{ij} - \underline{\mu})^{\mathrm{T}} \mathbf{R}^{-1} (\underline{z}_{ij} - \underline{\mu})\right) , \\
\underline{z}_{ij} &\triangleq \Lambda_b (\Lambda_a + \Lambda_b)^{-1} \underline{x}_i + \Lambda_a (\Lambda_a + \Lambda_b)^{-1} \underline{x}_j \,.
\end{aligned} \tag{12}$$

Note that $\mathbf{L}$ equals $\tilde{\mathbf{L}}$ in equation (11) if $a = b$.

With these results, the first two moments $\underline{\mu}_*, \mathbf{C}_*$ of $p\left(\mathrm{E}\{h(\underline{x}_*)\} | \underline{\mu}, \mathbf{C}\right)$ can be exactly determined.

5 GP-ADF: Assumed Density Filtering with Gaussian Processes

We assume that the transition dynamics f and the measurement function g in equations (1) and (2) are either not known or no longer accessible. Thus, we use models of the latent functions. We will model both functions by the GPs $\mathcal{GP}_f$ and $\mathcal{GP}_g$ with SE kernels k_f and k_g, respectively. We assume that we have access to ground truth observations of the hidden state during training.[1] In the following, we show how to exploit these GP models for assumed density filtering and derive the GP-ADF. We closely follow the steps in Section 2.

1 This can be described by the graphical model in Figure 1, where the states $\underline{x}_\tau$ are observed (shaded), and the index τ runs from $-n$ to -1.

5.1 Prediction Step ($\underline{x}_{k-1} \to \underline{x}_k$)

We compute the predictive distribution $p(\underline{x}_k|\underline{y}_{1:k-1})$ in equation (3). Using $p(\underline{x}_{k-1}|\underline{y}_{1:k-1})$, the result of the preceding filter step, as a Gaussian prior on $\underline{x}_{k-1}$, we predict the outcome of f for uncertain inputs according to Section 4.2 by treating $\underline{x}_{k-1}$ as $\underline{x}_*$ and f as h. Note that the transition density $p(\underline{x}_k|\underline{x}_{k-1})$ is exactly Gaussian due to $\mathcal{GP}_f$. By integrating out $\underline{x}_{k-1}$ using equation (3), we determine the first two moments $\underline{\mu}_k^p$ and $\mathbf{C}_k^p$ of the predictive distribution exactly and approximate $p(\underline{x}_k|\underline{y}_{1:k-1})$ by $\mathcal{N}(\underline{\mu}_k^p, \mathbf{C}_k^p)$.[2]

5.2 Filter Update ($\underline{y}_k \to \underline{x}_k$)

Now, let us consider the actual filter update at time step k. The goal is to determine $p(\underline{x}_k|\underline{y}_{1:k})$. The preceding prediction result $p(\underline{x}_k|\underline{y}_{1:k-1}) \approx \mathcal{N}(\underline{\mu}_k^p, \mathbf{C}_k^p)$ serves as the prior on $\underline{x}_k$ and will be combined with the recent observation $\underline{y}_k$ to determine the filter update (4) of the hidden state $\underline{x}_k$.

First, we determine the joint distribution

$$p(\underline{x}_k, \underline{y}_k|\underline{y}_{1:k-1}) = p(\underline{y}_k|\underline{x}_k) \cdot p(\underline{x}_k|\underline{y}_{1:k-1}) \,. \tag{13}$$

The GP measurement model $\mathcal{GP}_g$ yields an exact Gaussian likelihood $p(\underline{y}_k|\underline{x}_k)$, which is combined with the Gaussian prior $p(\underline{x}_k|\underline{y}_{1:k-1})$, to obtain an approximate Gaussian predictive distribution $p(\underline{y}_k|\underline{y}_{1:k-1}) \approx \mathcal{N}(\underline{\mu}_k^y, \mathbf{C}_k^y)$. Note that $\underline{\mu}_k^y$ and $\mathbf{C}_k^y$ are the exact moments of the predictive distribution, which can be computed analytically using the results from Section 4.2 by treating $\underline{x}_k$ as $\underline{x}_*$ and g as h. [3]

2 We write $\underline{\mu}_k^p$ and $\mathbf{C}_k^p$ to indicate a one-step ahead prediction from time step $k-1$ to k given $\underline{y}_{1:k-1}$.

3 In the following paragraph, we will implicitly assume that all variables are conditioned on the previous observations $\underline{y}_{1:k-1}$. Moreover, we will omit the time index k for brevity and clarity reasons. For example, $p(\underline{x}_k, \underline{y}_k|\underline{y}_{1:k-1})$ will be denoted by $p(\underline{x}, \underline{y})$.

To approximate the joint distribution $p(\underline{x},\underline{y})$ by a Gaussian distribution[4], we compute the cross terms $\mathbf{C}_{xy} = \mathrm{E}_{\underline{x},g}\{\underline{x}\,\underline{y}^{\mathrm{T}}\} - \underline{\mu}_k^p(\underline{\mu}_k^y)^{\mathrm{T}}$ of the joint covariance

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_k^p & \mathbf{C}_{xy} \\ \mathbf{C}_{xy}^{\mathrm{T}} & \mathbf{C}_k^y \end{bmatrix} .$$

For the unknown values $\mathrm{E}_{\underline{x},g}\{\underline{x}\,y^a\}$, we obtain

$$\begin{aligned}
\mathrm{E}_{\underline{x},g_a}\{\underline{x}\,y^a\} &= \mathrm{E}_{\underline{x},g_a}\{\underline{x}\,(g_a(\underline{x}) + \underline{v})\} = \mathrm{E}_{\underline{x},g_a}\{\underline{x}\,g_a(\underline{x})\} \\
&= \int \underline{x} \underbrace{\left(\int g_a(\underline{x})\,p\big(g_a|\underline{x}\big)\,\mathrm{d}g_a\right)}_{=\mathrm{E}_{g_a}\left\{g_a(\underline{x})\middle|\underline{x}\right\}=m_g^a(\underline{x})} p(\underline{x})\,\mathrm{d}\underline{x} \\
&\overset{(6)}{=} \int \underline{x} \left(\sum_{i=1}^{n} \beta_i^a\, k_g^a\big(\underline{x},\underline{x}_i\big)\right) p(\underline{x})\,\mathrm{d}\underline{x} \\
&= \sum_{i=1}^{n} \beta_i^a \int \underline{x}\, c_1\, \mathcal{N}\big(\underline{x}|\underline{x}_i,\Lambda_a\big)\mathcal{N}\Big(\underline{x}|\underline{\mu}_k^p,\mathbf{C}_k^p\Big)\,\mathrm{d}\underline{x}
\end{aligned}$$

for each target dimension $a = 1,\ldots,E$. Here, c_1^{-1} is the normalization constant of the unnormalized SE kernel k_g^a. Note that $\underline{x}_i$, $i = 1,\ldots,n$, are the training inputs of $\mathcal{GP}_g$. The product of the two Gaussians results in a new (unnormalized) Gaussian, the normalization constant of which is denoted by c_2^{-1}. The mean of this new Gaussian is a function of $\underline{x}_i$ and $\underline{\mu}_k^p$ and denoted by $\underline{\psi}\big(\underline{x}_i,\underline{\mu}_k^p\big)$. Hence, we finally obtain

$$\mathrm{E}_{\underline{x},g}\{\underline{x}\,y^a\} = c_1 c_2^{-1} \sum_{i=1}^{n} \beta_i^a \cdot \underline{\psi}\Big(\underline{x}_i,\underline{\mu}_k^p\Big) , \quad a = 1,\ldots,E ,$$

and the covariance matrix $\mathbf{C}$ is completely determined.

Second and finally, the joint Gaussian distribution

$$p\Big(\underline{x}_k,\underline{y}_k\big|\underline{y}_{1:k-1}\Big) = \mathcal{N}\left(\left[(\underline{\mu}_k^p)^{\mathrm{T}},(\underline{\mu}_k^y)^{\mathrm{T}}\right]^{\mathrm{T}},\mathbf{C}\right)$$

4 This approximation also appears in standard Gaussian filters, such as the UKF by [3].

leads to the actual filter update

$$p\left(\underline{x}_k \middle| \underline{y}_{1:k}\right) = \mathcal{N}\left(\underline{x}_k; \underline{\mu}_k^e, \mathbf{C}_k^e\right) , \tag{14}$$

with mean and covariance

$$\begin{aligned}\underline{\mu}_k^e &= \underline{\mu}_k^p + \mathbf{C}_{xy}\left(\mathbf{C}_k^y\right)^{-1}\left(\underline{y}_k - \underline{\mu}_k^y\right) ,\\ \mathbf{C}_k^e &= \mathbf{C}_k^p - \mathbf{C}_{xy}\left(\mathbf{C}_k^y\right)^{-1}\mathbf{C}_{xy}^{\mathrm{T}} ,\end{aligned}$$

respectively.

5.3 Assumptions and Computational Complexity

For performing prediction and filtering in closed form, we employ two approximations: First, if the input $\underline{x}_*$ is Gaussian distributed, we approximate the true predictive distributions $f(\underline{x}_*)$ and $g(\underline{x}_*)$ by a Gaussian with the exact mean and covariance. Second, the assumption that the joint distribution (13) is Gaussian, is only true if there is a linear relationship between $\underline{x}$ and $\underline{y}$. Otherwise, it is an approximation.

No sampling or finite-sample approximations are required in GP-ADF. In contrast to the UKF or the GP-UKF, the GP-ADF propagates *densities* instead of samples from them, which will allow for gradient-based parameter learning in nonlinear dynamic systems.

The computational complexity of predicting and filtering (after training the GPs) is $\mathcal{O}(E^3) + \mathcal{O}(DE^2n^2)$ due to the inversion of the predicted covariance matrices in equation (14), and the computation of the **L**-matrix (12) for the predictive covariance matrix. Here, D and E are the dimensionalities of the training inputs and the training targets, respectively, and n is the size of the GP training set. Classical filters, such as the EKF or the UKF, scale in $\mathcal{O}(E^3)$ computations.

6 Results

We assess filter performances for a 1D example with a single filter step and a time-series in a 2D example. The GP-UKF and the GP-ADF use the same models for the transition and observation functions. The UKF and EKF always have access to the true underlying functions and noise models. We implemented the UKF and the GP-UKF as described by [7].[5]

6.1 1D Example

We consider the one-dimensional nonlinear problem

$$x_{k+1} = \tfrac{1}{2}x_k + 25\tfrac{x_k}{1+x_k^2} + w\,, \quad w \sim \mathcal{N}\big(0, 0.2^2\big)\,,$$
$$y_k = 5\sin(2\,x_k) + v\,, \quad v \sim \mathcal{N}\big(0, 0.01^2\big)\,,$$

which is similar to the growth model by [5]. We randomly distributed 100 points in $[-10,10]$ to train $\mathcal{GP}_f$ and $\mathcal{GP}_g$. The prior on x_0 is Gaussian with mean $\mu_0 \in [-10,10]$ and variance $\sigma_0^2 = 0.5^2$. For 200 independent pairs $\big(x_0^{(i)}, y_1^{(i)}\big)$ of states and observations of the successor states, we assess the performance of a single filter step of four filters, the EKF, the UKF, the GP-UKF, and the GP-ADF. Figure 2 shows a typical realization of the filtered state distributions.

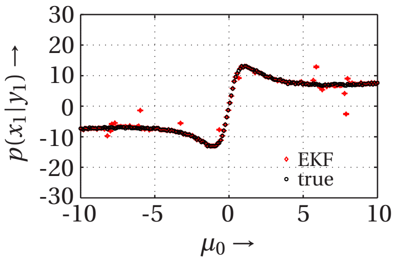
We evaluate the performance of the filters using two performance measures, the Mahalanobis distance

$$M_x = \sqrt{\big(\underline{x}_{\text{true}} - \underline{\mu}_k^e\big)^{\mathrm{T}} \big(\mathbf{C}_k^e\big)^{-1} \big(\underline{x}_{\text{true}} - \underline{\mu}_k^e\big)} \tag{15}$$

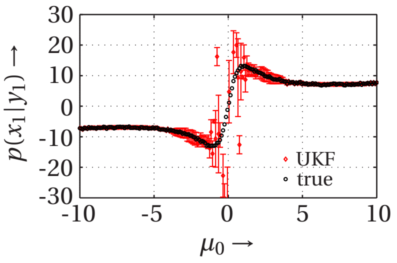
between the ground truth and the filtered mean and the negative log-likelihood NL_x of the hidden states. The filtered state distribution is an approximate Gaussian $\mathcal{N}\big(\underline{\mu}_k^e, \mathbf{C}_k^e\big)$. The units of M_x are standard deviations of $\underline{x}_{\text{true}}$ from the mean of the filter distribution. For both NL_x and M_x, lower values indicate better performance. NL_x penalizes both uncertainty and inconsistency, while M_x solely penalizes inconsistency. A distribution is *inconsistent* if the true underlying value is an outlier under the distribution.

Figure 3 shows that finite-sample approximations of densities can lead to overconfident predictions. The predictive distribution $p(y) = \mathcal{N}(-4.9, 0.0003)$ based upon finite samples claims full confidence. The actual measurement $y = -2.6$ cannot be explained.

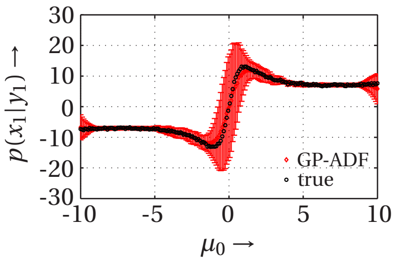
5 The UKF and GP-UKF implementations are based on Nando de Freitas' UPF software available at `http://www.cs.ubc.ca/~nando/software.html`. The GP-ADF code will be publicly available at `http://mlg.eng.cam.ac.uk/marc/download_icml2009.php`.



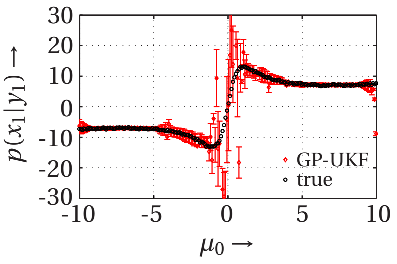
(a) EKF



(b) UKF



(c) GP-ADF



(d) GP-UKF

Figure 2: True hidden states (black) and filter distributions (red) for EKF, UKF, GP-ADF, and GP-UKF. The x-axis shows μ_0, the mean value of $p\big(x_0^{(i)}\big)$, the y-axis is the filtered distribution $p(x_1|y_1) \triangleq p\big(x_1^{(i)}|y_1^{(i)},\mu_0^{(i)},\sigma_0^2\big)$ of the hidden state. The error bars show twice the standard deviations of the filtered state distributions. The filtered state distributions of the EKF, UKF, and the GP-UKF suffer from occasional inconsistencies that do not explain the true state at all. In contrast, GP-ADF is always consistent.

Table 2: Average filter performances (1D example).

	$NL_x^{0.25}$	$NL_x^{0.5}$	$NL_x^{0.75}$	M_x
EKF	2.4×10^5	2.9×10^5	3.5×10^5	30.2 ± 3.2
UKF	4.7×10^4	6.5×10^4	1.1×10^5	3.9 ± 0.9
GP-UKF	319	1.1×10^3	1.3×10^4	1.5 ± 1.0
GP-ADF	**90**	**98**	**106**	$\mathbf{0.46\pm0.04}$

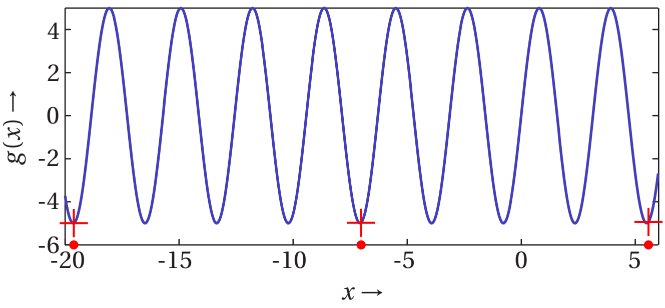


Figure 3: Typical failing of unscented filters. Although the function highly varies, the sigma points (red dots) are mapped to almost the same function value (red crosses). The sample predictive distribution is overconfident.

Table 2 shows the average performance of the filters after 100 independent runs of the filter experiment. We report the upper and lower quantiles $NL_x^{0.75}, NL_x^{0.25}$ and the median of NL_x as well as the mean and the standard deviation of M_x.

According to NL_x, EKF is outperformed by all other filters. The EKF and the UKF heavily suffer from inconsistencies. The GP-UKF performs better than the UKF since particularly $\mathcal{GP}_g$ does not have training data in all relevant regions, which alleviates the overconfidence problem in Figure 3. According to the error measure M_x, GP-ADF yields substantially better results than all other filters. Moreover, the performance of GP-ADF is stable, which is expressed by the quantiles.

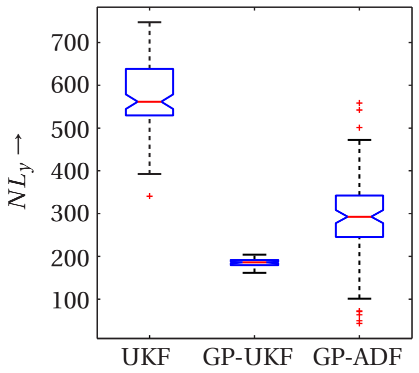
6.2 Recursive Filtering: Time-Series

We consider the problem of recursively filtering a time-series of a two-dimensional pendulum, where

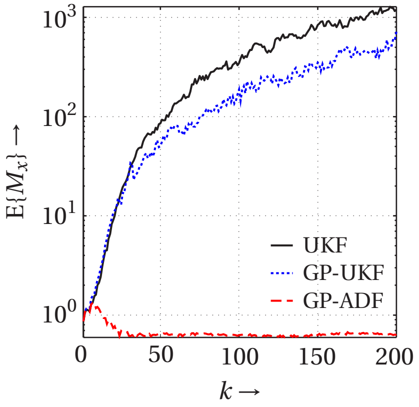
$$\underline{x}_k = \begin{bmatrix} \varphi_{k-1} + \Delta_t \dot{\varphi}_{k-1} + \frac{\Delta_t^2}{2} \frac{mgl\sin(\varphi_{k-1}) + u_{k-1}}{ml^2} \\ \dot{\varphi}_{k-1} + \Delta_t \frac{mgl\sin(\varphi_{k-1}) + u_{k-1}}{ml^2} \end{bmatrix} + \underline{w} ,$$

is the time-discretized dynamics model and

$$\underline{y}_k = \begin{bmatrix} \arctan\left(\frac{p_1 - l\sin(\varphi_k)}{p_1 - l\cos(\varphi_k)}\right) \\ \arctan\left(\frac{p_2 - l\sin(\varphi_k)}{p_2 - l\cos(\varphi_k)}\right) \end{bmatrix} + \underline{v} , \quad \text{with } \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \end{bmatrix} , \qquad (16)$$



(a) The median (notch), the lower and upper quantile (blue box), and the spread of the negative log predictive likelihood. The crosses are outliers.



(b) The x-axis shows the time steps, the y-axis displays the averaged Mahalanobis distance M_x on a logarithmic scale.

Figure 4: Recursive filter performances of UKF, GP-ADF, and GP-UKF for the 2D pendulum. (a) shows the negative log predictive likelihood NL_y. While the performances of the UKF and the GP-UKF vary strongly and depend on the particular noise realizations, the GP-ADF reliably provides a good solution. (b) shows the averaged Mahalanobis distances of the filters. In contrast to the GP-ADF, the UKF and the GP-UKF quickly become inconsistent.

is the observation model. We choose $\mathbf{C}_w = \text{diag}(0.1^2, 0.3^2)$, $\mathbf{C}_v = \text{diag}(0.2^2, 0.2^2)$. Here, $\underline{x}^\mathrm{T} = [\varphi, \dot{\varphi}]$ with $\varphi, \dot{\varphi}$ are the angle and the angular velocity, respectively. The applied torque is denoted by $u \in [-5,5]$ Nm, the acceleration of gravity is $g = 9.81\,\text{m/s}^2$, the length of the pendulum is $l = 1$ m, the mass of the pendulum is $m = 1$ kg. The discretization constant is $\Delta_t = 400$ ms. The measurement equation (16) describes *bearings-only measurements* of the Cartesian coordinates of the pendulum tip and solely depends on the angle. Thus, the filter distribution of the angular velocity has to be reconstructed by using the cross-correlation information between angle and angular velocity in the transition dynamics model. We used 200 data points to train $\mathcal{GP}_f$ and $\mathcal{GP}_g$.

We start 100 independent trajectories from the initial state distribution $\underline{x}_0 \sim \mathcal{N}(\underline{\mu}_0, \mathbf{C}_0)$ with $\underline{\mu}_0^\mathrm{T} = [-\pi, 0]$ and $\mathbf{C}_0 = \text{diag}(0.1^2, 0.2^2)$. This corresponds to the still pendulum hanging downward. We fuse information of a state prediction and a corresponding observation at each time step k. This filtered state distribution serves as prior for the subsequent state prediction. We iterate this procedure for 200 time steps.

In Figure 4, we compare the performances of the UKF, the GP-ADF, and the GP-UKF by considering NL_y, the negative log predictive likelihood of a full trajectory. NL_y assesses whether the observations $\underline{y}_k$ can be explained by the predicted measurement distributions $p(\underline{y}_k|\underline{y}_{1:k-1}) = \mathcal{N}(\underline{\mu}_k^y, \mathbf{C}_k^y)$. Note that in contrast to NL_x, NL_y solely depends on observations $\underline{y}$, and no longer on the hidden variables $\underline{x}$. Additionally, we consider the M_x-measure.

A major observation is that the UKF and the GP-UKF are unaware of losing track of the state since the final covariances are tiny. Therefore, they often yield inconsistent solutions after 200 time steps, whereas the GP-ADF determines tight, but consistent distributions.

In general, we observed that the performance of GP-ADF is particularly good for non-negligible noise levels and fairly nonlinear mappings f and g. If the state uncertainty is small or the functions f and g are nearly linear, the UKF and the GP-UKF perform well.

7 Discussion

Non-parametric probabilistic GP models describe distributions over all functions that plausibly explain the data. In the context of our work, this property matters if a parametric model cannot easily be determined or the real system does not closely follow idealized models.

We observe that the uncertainty in the GP-ADF is often larger than the uncertainty in the UKF and the GP-UKF, which depends on two factors. First, in contrast to the GP-UKF, the GP-ADF explicitly incorporates the uncertainty about the underlying function. Second, the predictive uncertainty is computed using the entire prior. Due to the appropriate treatment of uncertainties, we observe that the predictions of the GP-ADF are rarely inconsistent.

Both UKF-based algorithms can easily fail when the functions, which are used for mapping the sigma points, are highly nonlinear and the input distribution is wide (see Figure 3). The UKF and EKF are solely applicable when the functions are known or directly accessible. If only samples of the underlying function are available, models have to be employed. [6] replace transition and measurement functions by GP models in standard filters, such as the EKF and the UKF. However, they do not exploit the GP structure that allows for an exact computation of the first two predictive moments given a Gaussian prior. Since [7] and [6] do not exploit these properties, the GP-UKF is not moment preserving.

The GP-ADF can be considered the limit of the GP-UKF propagating infinitely many samples from a Gaussian input distribution if additionally the corresponding function values are sampled from the GP predictive distribution.

Like [7] and [6], we assume that the transition function and the measurement function can be learned by having access to ground truth observations of the hidden states. The measurement function could be learned independent of the transition function, but (measurement) noise-free observations of the hidden states in Figure 1 can be difficult to obtain.

For highly uncertain models for the latent functions f,g GP-ADF is still consistent and shows the same stable performance as described in Figure 4a.

8 Summary and Future Work

In this paper, we propose the GP-ADF, a fully Bayesian approach to assumed density filtering for nonlinear dynamics and observation models. Similar to the papers by [7] and [6], we model the transition dynamics and the measurement function by GPs. However, we propagate full densities and approximate the predictive distribution by a Gaussian with the exact moments. In contrast to the EKF, the UKF, and the recent GP-UKF, our filter is consistent and moment preserving.

We will complete the forward-backward algorithm and learn the GP models for the transition dynamics and the measurements without the need of direct access to the hidden states. We will utilize Expectation Maximization for this purpose since GP-ADF allows for gradient-based parameter optimization.

Acknowledgements

We thank Ryan Turner, Carl Edward Rasmussen, and the reviewers for very helpful comments and suggestions. MPD acknowledges support by the German Research Foundation (DFG) through grant RA 1030/1-3.

References

[1] Xavier Boyen and Daphne Koller. Tractable Inference for Complex Stochastic Processes. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 33–42, San Francisco, CA, USA, 1998.

[2] Zoubin Ghahramani and Sam T. Roweis. Learning Nonlinear Dynamical Systems using an EM Algorithm. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 599–605. The MIT Press, 1999.

[3] Simon J. Julier and Jeffrey K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *IEEE Review*, 92(3):401–422, March 2004.

[4] Rudolf E. Kalman. A new Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME, Journal of Basic Engineering*, 82 (Series D)(1):35–45, 1960.

[5] Genshiro Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[6] Jonathan Ko and Dieter Fox. GP-BayesFilters: Bayesian Filtering Using Gaussian Process Prediction and Observation Models. In *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3471–3476, Nice, France, September 2008.

[7] Jonathan Ko, Daniel J. Klein, Dieter Fox, and Dirk Haehnel. Gaussian Processes and Reinforcement Learning for Identification and Control of an Autonomous Blimp. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 742–747, Rome, Italy, April 2007.

[8] Malte Kuss. *Gaussian Process Models for Robust Regression, Classification, and Reinforcement Learning*. PhD thesis, Technische Universität Darmstadt, Germany, February 2006.

[9] Tine Lefebvre, Herman Bruyninckx, and Joris De Schutter. *Nonlinear Kalman Filtering for Force-Controlled Robot Tasks.* Springer Berlin, 2005.

[10] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 141 of *Mathematics in Science and Engineering.* Academic Press, Inc., 1979.

[11] Manfred Opper. A Bayesian Approach to Online Learning. In *Online Learning in Neural Networks*, pages 363–378. Cambridge University Press, 1998.

[12] Joaquin Quiñonero-Candela, Agathe Girard, Jan Larsen, and Carl E. Rasmussen. Propagation of Uncertainty in Bayesian Kernel Models—Application to Multiple-Step Ahead Forecasting. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 701–704, April 2003.

[13] Carl E. Rasmussen and Zoubin Ghahramani. Bayesian Monte Carlo. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 489–496. The MIT Press, Cambridge, MA, USA, 2003.

[14] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning.* The MIT Press, Cambridge, MA, USA, 2006.

[15] Dan Simon. *Optimal State Estimation: Kalman, H-Infinity, and Nonlinear Approaches.* John Wiley & Sons, Inc., 1st edition, 2006.

Paper I

Robust Filtering and Smoothing with Gaussian Processes

Authors: Marc P. Deisenroth, Ryan D. Turner, Marco F. Huber, Uwe D. Hanebeck, and Carl E. Rasmussen

Edited version of the paper:

M. P. Deisenroth, R. D. Turner, M. F. Huber, U. D. Hanebeck, and C. E. Rasmussen. Robust Filtering and Smoothing with Gaussian Processes. In *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pages 1865–1871, July 2012.

Robust Filtering and Smoothing with Gaussian Processes

Marc P. Deisenroth[*], Ryan D. Turner[**], Marco F. Huber[***], Uwe D. Hanebeck[****], and Carl E. Rasmussen[**]

[*]Intelligent Autonomous Systems
TU Darmstadt
Darmstadt, Germany
marc@ias.tu-darmstadt.de

[**]Department of Engineering
University of Cambridge
Cambridge, UK
{rt324|cer54}@cam.ac.uk

[***]AGT International
Darmstadt, Germany
marco.huber@ieee.org

[****] Intelligent Sensor-Actuator-Systems Laboratory (ISAS)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
uwe.hanebeck@ieee.org

Abstract

We propose a principled algorithm for robust Bayesian filtering and smoothing in nonlinear stochastic dynamic systems when both the transition function and the measurement function are described by non-parametric Gaussian process (GP) models. GPs are gaining increasing importance in signal processing, machine learning, robotics, and control for representing unknown system functions by posterior probability distributions. This modern way of "system identification" is more robust than finding point estimates of a parametric function representation. In this article, we present a principled algorithm for robust analytic smoothing in GP dynamic systems, which are increasingly used in robotics and control. Our numerical evaluations demonstrate the robustness of the proposed approach in situations where other state-of-the-art Gaussian filters and smoothers can fail.

1 Introduction

Filtering and smoothing in context of dynamic systems refers to a Bayesian methodology for computing posterior distributions of the latent state based on a history of noisy measurements. This kind of methodology can be found, e.g., in navigation, control engineering, robotics, and machine learning [1, 3, 28, 30]. Solutions to filtering [1, 3, 14, 28, 30] and smoothing [18, 23, 26, 27] in linear dynamic systems are well known, and numerous approximations for nonlinear systems have been proposed, for both filtering [2, 6, 12, 13, 16, 20] and smoothing [7, 11, 15, 29].

In this article, we focus on Gaussian filtering and smoothing in Gaussian process (GP) dynamic systems. GPs are a robust non-parametric method for approximating unknown functions by a posterior distribution over them [19, 25]. Although GPs have been around for decades, they only recently became computationally interesting for applications in robotics, control, and machine learning [8, 9, 17, 21, 22].

The contribution of this article is the derivation of a novel, principled, and robust Rauch-Tung-Striebel (RTS) smoother for GP dynamic systems, which we call the *GP-RTSS*. The GP-RTSS computes a Gaussian approximation to the smoothing distribution in closed form. The posterior filtering and smoothing distributions can be computed *without* linearization [20] or (small) sampling approximations of densities [13].

We provide numerical evidence that the GP-RTSS is more robust than state-of-the-art nonlinear Gaussian filtering and smoothing algorithms including the extended Kalman filter (EKF) [20], the unscented Kalman filter (UKF) [13], the cubature Kalman filter (CKF) [2], the GP-UKF [16], and their corresponding RTS smoothers. *Robustness* refers to the ability of an inferred distribution to explain the "true" state/measurement.

The paper is structured as follows: In Sections 1.1–1.2, we introduce the problem setup and necessary background on Gaussian smoothing and GP dynamic systems. In Section 2, we briefly introduce Gaussian process regression, discuss the expressiveness of a GP, and explain how to train GPs. Section 3 details our proposed method (GP-RTSS) for smoothing in GP dynamic systems. In Section 4, we provide experimental evidence of the robustness of the GP-RTSS. Section 5 concludes the paper with a discussion.

1.1 Problem Formulation and Notation

In this article, we consider discrete-time stochastic systems

$$\underline{x}_t = f\big(\underline{x}_{t-1}\big) + \underline{w}_t\,, \tag{1}$$

$$\underline{z}_t = g\big(\underline{x}_t\big) + \underline{v}_t\,, \tag{2}$$

where $\underline{x}_t \in \mathbb{R}^D$ is the state, $\underline{z}_t \in \mathbb{R}^E$ is the measurement at time step t, $\underline{w}_t \sim \mathcal{N}(\underline{0}, \mathbf{C}_w)$ is Gaussian system noise, $\underline{v}_t \sim \mathcal{N}(\underline{0}, \mathbf{C}_v)$ is Gaussian measurement noise, f is the transition function (or system function) and g is the measurement function. The discrete time steps t run from 0 to T. The initial state $\underline{x}_0$ of the time series is distributed according to a Gaussian prior distribution $p\big(\underline{x}_0\big) = \mathcal{N}\big(\underline{\mu}_0^x, \mathbf{C}_0^x\big)$. The purpose of filtering and smoothing is to find approximations to the posterior distributions $p\big(\underline{x}_t|\underline{z}_{1:\tau}\big)$, where $1{:}\tau$ in a subindex abbreviates $1,\ldots,\tau$ with $\tau = t$ during filtering and $\tau = T$ during smoothing. In this article, we consider Gaussian approximations $p\big(\underline{x}_t|\underline{z}_{1:\tau}\big) \approx \mathcal{N}\big(\underline{x}_t; \underline{\mu}_{t|\tau}^x, \mathbf{C}_{t|\tau}^x\big)$ of the latent state posterior distributions $p\big(\underline{x}_t|\underline{z}_{1:\tau}\big)$. We use the short-hand notation $\underline{a}_{b|c}^d$ where $\underline{a} = \underline{\mu}$ denotes the mean $\underline{\mu}$ and $\underline{a} = \mathbf{C}$ denotes the covariance, b denotes the time step under consideration, c denotes the time step up to which we consider measurements, and $d \in \{x, z\}$ denotes either the latent space (x) or the observed space (z).

1.2 Gaussian RTS Smoothing

Given the filtering distributions $p\big(\underline{x}_t\big|\underline{z}_{1:t}\big) = \mathcal{N}\big(\underline{x}_t; \underline{\mu}_{t|t}^x, \mathbf{C}_{t|t}^x\big)$, $t = 1,\ldots,T$, a sufficient condition for Gaussian smoothing is the computation of Gaussian approximations of the joint distributions $p\big(\underline{x}_{t-1}, \underline{x}_t\big|\underline{z}_{1:t-1}\big)$, $t = 1,\ldots,T$ [7].

In Gaussian smoothers, the standard smoothing distribution for the dynamic system in Equations (1)–(2) is always

$$p\big(\underline{x}_{t-1}\big|\underline{z}_{1:T}\big) = \mathcal{N}\big(\underline{x}_{t-1}; \underline{\mu}_{t-1|T}^x, \mathbf{C}_{t-1|T}^x\big)\,, \tag{3}$$

where

$$\underline{\mu}_{t-1|T}^x = \underline{\mu}_{t-1|t-1}^x + \mathbf{J}_{t-1}\left(\underline{\mu}_{t|T}^x - \underline{\mu}_{t|t-1}^x\right)\,, \tag{4}$$

$$\mathbf{C}_{t-1|T}^x = \mathbf{C}_{t-1|t-1}^x + \mathbf{J}_{t-1}\big(\mathbf{C}_{t|T}^x - \mathbf{C}_{t|t-1}^x\big)\mathbf{J}_{t-1}^{\mathrm{T}}\,, \tag{5}$$

$$\mathbf{J}_{t-1} \triangleq \mathbf{C}_{t-1,t|t-1}^x\big(\mathbf{C}_{t|t-1}^x\big)^{-1}\,, \tag{6}$$

for $t = T, \ldots, 1$. Depending on the methodology of computing this joint distribution, we can directly derive arbitrary RTS smoothing algorithms, including the URTSS [29], the EKS [1, 20], the CKS [7], a smoothing extension to the CKF [2], or the GP-URTSS, a smoothing extension to the GP-UKF [16]. The individual smoothers (URTSS, EKS, CKS, GP-based smoothers etc.) simply differ in the way of computing/estimating the means and covariances required in Equations (4)–(6) (see [7]).

To derive the GP-URTSS, we closely follow the derivation of the URTSS [29]. The GP-URTSS is a novel smoother, but its derivation is relatively straightforward and therefore not detailed in this article. Instead, we detail the derivation of the GP-RTSS, a robust Rauch-Tung-Striebel smoother for GP dynamic systems, which is based on analytic computation of the means and (cross-)covariances in Equations (4)–(6).

In *GP dynamics systems*, the transition function f and the measurement function g in Equations (1)–(2) are modeled by Gaussian processes. This setup is getting more relevant in practical applications such as robotics and control, where it can be difficult to find an accurate parametric form of f and g, respectively [4, 9]. Given the increasing use of GP models in robotics and control, the robustness of Bayesian state estimation is important.

2 Gaussian Processes

In the standard GP regression model, we assume that the data $\mathcal{D} \triangleq \{\mathbf{X}, \underline{y}\}$ with $\mathbf{X} \triangleq [\underline{x}_1, \ldots, \underline{x}_n]$ and $\underline{y} \triangleq [y_1, \ldots, y_n]^\mathrm{T}$ have been generated according to $y_i = h(\underline{x}_i) + \epsilon_i$, where $h : \mathbb{R}^D \to \mathbb{R}$ and $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$ is independent (measurement) noise. GPs consider h a random function and infer a posterior distribution over h from data. The posterior is used to make predictions about function values $h(\underline{x}_*)$ for arbitrary inputs $\underline{x}_* \in \mathbb{R}^D$.

Similar to a Gaussian distribution, which is fully specified by a mean vector and a covariance matrix, a GP is fully specified by a mean *function* $m_h(\cdot)$ and a covariance *function*

$$k_h(\underline{x}, \underline{x}') \triangleq \mathrm{E}_h\left\{\big(h(\underline{x}) - m_h(\underline{x})\big)\big(h(\underline{x}') - m_h(\underline{x}')\big)\right\} \tag{7}$$

$$= \mathrm{cov}_h\left\{h(\underline{x}), h(\underline{x}')\right\} \in \mathbb{R}\,, \tag{8}$$

with $\underline{x}$, $\underline{x}' \in \mathbb{R}^D$. The latter specifies the covariance between any two function values. Here, E_h denotes the expectation with respect to the function h. The covariance function $k_h(\cdot,\cdot)$ is also called a *kernel*.

Unless stated otherwise, we consider a prior mean function $m_h \equiv 0$ and use the squared exponential (SE) covariance function with automatic relevance determination

$$k_{\mathrm{SE}}\left(\underline{x}_p,\underline{x}_q\right) \triangleq \alpha^2 \exp\left(-\tfrac{1}{2}\left(\underline{x}_p - \underline{x}_q\right)^{\mathrm{T}} \Lambda^{-1}\left(\underline{x}_p - \underline{x}_q\right)\right) \tag{9}$$

for $\underline{x}_p, \underline{x}_q \in \mathbb{R}^D$, plus a noise covariance function $k_{\mathrm{noise}} \triangleq \delta_{pq}\sigma_\epsilon^2$, such that $k_h = k_{\mathrm{SE}} + k_{\mathrm{noise}}$. The δ denotes the Kronecker symbol that is unity when $p = q$ and zero otherwise, resulting in i.i.d. measurement noise. In (9), $\Lambda = \operatorname{diag}\left(\ell_1^2,\ldots,\ell_D^2\right)$ is a diagonal matrix of squared characteristic length-scales ℓ_i, $i = 1,\ldots,D$, and α^2 is the signal variance of the latent function h. By using the SE covariance function from (9) we assume that the latent function h is smooth and stationary. Smoothness and stationarity is easier to justify than fixed parametric form of the underlying function.

2.1 Expressiveness of the Model

Although the SE covariance function and the zero-prior mean function are common defaults, they retain a great deal of expressiveness. Inspired by [19**?**], we demonstrate this expressiveness and show the correspondence of our GP model to a universal function approximator: Consider a function

$$h(x) = \sum_{i\in\mathbb{Z}} \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \gamma_n \cdot \exp\left(-\frac{(x-(i+\frac{n}{N}))^2}{\lambda^2}\right) \tag{10}$$

where $\gamma_n \sim \mathcal{N}(0,1)$, $n = 1,\ldots,N$. Note that in the limit $h(x)$ is represented by infinitely many Gaussian-shaped basis functions along the real axis with variance $\lambda\sqrt{2}$ and prior (Gaussian) random weights γ_n, for $x \in \mathbb{R}$, and for all $i \in \mathbb{Z}$. The model in (10) is considered a universal function approximator. Writing the sums in (10) as an integral over the real axis $\mathbb{R}$, we obtain

$$\begin{aligned} h(x) &= \sum_{i\in\mathbb{Z}} \int_i^{i+1} \gamma(s)\cdot \exp\left(-\frac{(x-s)^2}{\lambda^2}\right)\mathrm{d}s \\ &= \int_{-\infty}^{\infty} \gamma(s)\cdot \exp\left(-\frac{(x-s)^2}{\lambda^2}\right)\mathrm{d}s = \left(\gamma * \mathcal{K}\right)(x)\,, \end{aligned} \tag{11}$$

where $\gamma(s) \sim \mathcal{N}(0,1)$ is a white-noise process and $\mathcal{K}$ is a Gaussian convolution kernel. The function values of h are jointly normal, which follows from the convolution $\gamma * \mathcal{K}$. We now analyze the mean function and the covariance function of h, which fully specify the distribution of h. The only random variables are the weights $\gamma(s)$. Computing the expected function of this model (prior mean function) requires averaging over $\gamma(s)$ and yields

$$\mathrm{E}_\gamma\{h(x)\} = \int h(x) p(\gamma(s))\,\mathrm{d}\gamma(s) \tag{12}$$

$$\overset{(11)}{=} \int \exp\left(-\frac{(x-s)^2}{\lambda^2}\right) \int \gamma(s) p\big(\gamma(s)\big)\,\mathrm{d}\gamma(s)\,\mathrm{d}s = 0 \tag{13}$$

since $\mathrm{E}_\gamma\{\gamma(s)\} = 0$. Hence, the mean function of h equals zero everywhere. Let us now find the covariance function. Since the mean function equals zero, for any $x, x' \in \mathbb{R}$ we obtain

$$\begin{aligned}\mathrm{cov}_\gamma\left\{h(x), h(x')\right\} &= \int h(x) h(x') p\big(\gamma(s)\big)\,\mathrm{d}\gamma(s) \\ &= \int \exp\left(-\frac{(x-s)^2}{\lambda^2}\right) \exp\left(-\frac{(x'-s)^2}{\lambda^2}\right) \\ &\qquad \times \int \gamma(s)^2 p\big(\gamma(s)\big)\,\mathrm{d}\gamma(s)\,\mathrm{d}s\,,\end{aligned} \tag{14}$$

where we used the definition of h in (11). Using $\mathrm{var}_\gamma\left\{\gamma(s)\right\} = 1$ and completing the squares yields

$$\begin{aligned}\mathrm{cov}_\gamma\left\{h(x), h(x')\right\} &= \int \exp\left(-\frac{2\big(s - \frac{x+x'}{2}\big)^2 + \frac{(x-x')^2}{2}}{\lambda^2}\right)\mathrm{d}s \\ &= \alpha^2 \exp\left(-\frac{(x-x')^2}{2\lambda^2}\right)\end{aligned} \tag{15}$$

for suitable α^2.

From (13) and (15), we see that the mean function and the covariance function of the universal function approximator in (10) correspond to the GP model assumptions we made earlier: a prior mean function $m_h \equiv 0$ and the SE covariance function in (9) for a one-dimensional input space. Hence, the considered GP prior implicitly assumes latent functions h that can be described by the universal function approximator in (11). Examples of covariance functions that encode different model assumptions are given in [25].

2.2 Training via Evidence Maximization

For E target dimensions, we train E GPs assuming that the target dimensions are independent at a deterministically given test input (if the test input is uncertain, the target dimensions covary): After observing a data set $\mathcal{D}$, for each (training) target dimension, we learn the $D+1$ hyper-parameters of the covariance function and the noise variance of the data using *evidence maximization* [19, 25]: Collecting all $(D+2)E$ hyper-parameters in the vector $\underline{\theta}$, evidence maximization yields a point estimate $\hat{\underline{\theta}} \in \arg\max_{\underline{\theta}} \log p\big(\underline{y}\big|\mathbf{X},\underline{\theta}\big)$. Evidence maximization automatically trades off data fit with function complexity and avoids overfitting [25].

From here onward, we consider the GP dynamics system setup, where two GP models have been trained using evidence maximization: $\mathcal{GP}_f$, which models the mapping $\underline{x}_{t-1} \mapsto \underline{x}_t$, $\mathbb{R}^D \to \mathbb{R}^D$, see (1), and $\mathcal{GP}_g$, which models the mapping $\underline{x}_t \mapsto \underline{z}_t$, $\mathbb{R}^D \to \mathbb{R}^E$, see (2). To keep the notation uncluttered, we do not explicitly condition on the hyper-parameters $\hat{\underline{\theta}}$ and the training data $\mathcal{D}$ in the following.

3 Robust Smoothing in Gaussian Process Dynamic Systems

Analytic moment-based filtering in GP dynamic systems has been proposed in [6], where the filter distribution is given by

$$p\big(\underline{x}_t\big|\underline{z}_{1:t}\big) = \mathcal{N}\big(\underline{x}_t; \underline{\mu}^x_{t|t}, \mathbf{C}^x_{t|t}\big)\,, \tag{16}$$

$$\underline{\mu}^x_{t|t} = \underline{\mu}^x_{t|t-1} + \mathbf{C}^{xz}_{t|t-1}\big(\mathbf{C}^z_{t|t-1}\big)^{-1}\Big(\underline{z}_t - \underline{\mu}^z_{t|t-1}\Big)\,, \tag{17}$$

$$\mathbf{C}^x_{t|t} = \mathbf{C}^x_{t|t-1} - \mathbf{C}^{xz}_{t|t-1}\big(\mathbf{C}^z_{t|t-1}\big)^{-1}\mathbf{C}^{zx}_{t|t-1}\,, \tag{18}$$

for $t = 1,\ldots,T$. Here, we extend these filtering results to analytic moment-based smoothing, where we explicitly take nonlinearities into account (no linearization required) while propagating full Gaussian densities (no sigma/cubature-point representation required) through nonlinear GP models.

In the following, we detail our novel RTS smoothing approach for GP dynamic systems. We fit our smoother in the standard frame of (4)–(6). For this, we compute the means and covariances of the Gaussian approximation

$$\mathcal{N}\left(\begin{bmatrix}\underline{x}_{t-1}\\ \underline{x}_t\end{bmatrix};\begin{bmatrix}\underline{\mu}^x_{t-1|t-1}\\ \underline{\mu}^x_{t|t-1}\end{bmatrix},\begin{bmatrix}\mathbf{C}^x_{t-1|t-1} & \mathbf{C}^x_{t-1,t|t-1}\\ \mathbf{C}^x_{t,t-1|t-1} & \mathbf{C}^x_{t|t-1}\end{bmatrix}\right) \tag{19}$$

to the joint $p(\underline{x}_{t-1},\underline{x}_t|\underline{z}_{1:t-1})$, after which the smoother is fully determined [7]. Our approximation does not involve sampling, linearization, or numerical integration. Instead, we present closed-form expressions of a deterministic Gaussian approximation of the joint distribution in (19).

In our case, the mapping $\underline{x}_{t-1}\mapsto\underline{x}_t$ is not known, but instead it is distributed according to $\mathcal{GP}_f$, a distribution over system functions. For robust filtering and smoothing, we therefore need to take the GP (model) uncertainty into account by Bayesian averaging according to the GP distribution [6, 24]. The marginal $p(\underline{x}_{t-1}|\underline{z}_{1:t-1})=\mathcal{N}(\underline{\mu}^x_{t-1|t-1},\mathbf{C}^x_{t-1|t-1})$ is known from filtering [6]. In Section 3.1, we compute the mean and covariance of second marginal $p(\underline{x}_t|\underline{z}_{1:t-1})$ and then in Section 3.2 the cross-covariance terms $\mathbf{C}^x_{t-1,t|t-1}=\operatorname{cov}\{\underline{x}_{t-1},\underline{x}_t|\underline{z}_{1:t-1}\}$.

3.1 Marginal Distribution

Marginal Mean

Using the system (1) and integrating over all three sources of uncertainties (the system noise, the state $\underline{x}_{t-1}$, and the system function itself), we apply the law of total expectation and obtain the marginal mean

$$\underline{\mu}^x_{t|t-1}=\mathrm{E}_{\underline{x}_{t-1}}\left\{\mathrm{E}_f\left\{f(\underline{x}_{t-1})|\underline{x}_{t-1}\right\}|\underline{z}_{1:t-1}\right\}. \tag{20}$$

The expectations in (20) are taken with respect to the posterior GP distribution $p(f)$ and the filter distribution $p(\underline{x}_{t-1}|\underline{z}_{1:t-1})=\mathcal{N}(\underline{\mu}^x_{t-1|t-1},\mathbf{C}^x_{t-1|t-1})$ at time step $t-1$. Equation (20) can be rewritten as $\underline{\mu}^x_{t|t-1}=\mathrm{E}_{\underline{x}_{t-1}}\left\{m_f(\underline{x}_{t-1})\big|\underline{z}_{1:t-1}\right\}$ with $m_f(\underline{x}_{t-1})\triangleq\mathrm{E}_f\left\{f(\underline{x}_{t-1})\big|\underline{x}_{t-1}\right\}$ is the posterior mean function of $\mathcal{GP}_f$. Writing

m_f as a finite sum over the SE kernels centered at all n training inputs [25], the predicted mean for each target dimension $a = 1,\dots,D$ is

$$\begin{aligned}(\underline{\mu}^x_{t|t-1})_a &= \int m_{f_a}(\underline{x}_{t-1}) p(\underline{x}_{t-1}|\underline{z}_{1:t-1}) \,\mathrm{d}\underline{x}_{t-1} \\ &= \sum_{i=1}^{n} \beta^x_{a_i} \int k_{f_a}(\underline{x}_{t-1}, \underline{x}_i) p(\underline{x}_{t-1}|\underline{z}_{1:t-1}) \,\mathrm{d}\underline{x}_{t-1} \,,\end{aligned} \tag{21}$$

where $p(\underline{x}_{t-1}|\underline{z}_{1:t-1}) = \mathcal{N}(\underline{x}_{t-1}; \underline{\mu}^x_{t-1|t-1}, \mathbf{C}^x_{t-1|t-1})$ is the filter distribution at time $t-1$. Moreover, $\underline{x}_i$, $i = 1,\dots,n$, are the training set of $\mathcal{GP}_f$, k_{f_a} is the covariance function of $\mathcal{GP}_f$ for the ath target dimension (GP hyper-parameters are not shared across dimensions), and $\underline{\beta}^x_a \triangleq (\mathbf{K}_{f_a} + \sigma^2_{w_a}\mathbf{I})^{-1} \underline{y}_a \in \mathbb{R}^n$. For dimension a, $\mathbf{K}_{f_a}$ denotes the kernel matrix (Gram matrix), where $\mathbf{K}_{f_{a_{ij}}} = k_{f_a}(\underline{x}_i, \underline{x}_j)$, $i, j = 1,\dots,n$. Moreover, $\underline{y}_a$ are the training targets, and $\sigma^2_{w_a}$ is the learned system noise variance. The vector $\underline{\beta}^x_a$ has been pulled out of the integration since it is independent of $\underline{x}_{t-1}$. Note that $\underline{x}_{t-1}$ serves as a test input from the perspective of the GP regression model.

For the SE covariance function in (9), the integral in (21) can be computed analytically (other tractable choices are covariance functions containing combinations of squared exponentials, trigonometric functions, and polynomials). The marginal mean is given as

$$(\underline{\mu}^x_{t|t-1})_a = (\underline{\beta}^x_a)^{\mathrm{T}} \underline{q}^x_a \tag{22}$$

where we defined

$$\begin{aligned} q^x_{a_i} &\triangleq \alpha^2_{f_a} \left|\mathbf{C}^x_{t-1|t-1}\Lambda_a^{-1} + \mathbf{I}\right|^{-\frac{1}{2}} \\ &\quad \times \exp\left(-\tfrac{1}{2}\left(\underline{x}_i - \underline{\mu}^x_{t-1|t-1}\right)^{\mathrm{T}} \mathbf{S}^{-1} \left(\underline{x}_i - \underline{\mu}^x_{t-1|t-1}\right)\right) \,, \end{aligned} \tag{23}$$

$$\mathbf{S} \triangleq \mathbf{C}^x_{t-1|t-1} + \Lambda_a \,, \tag{24}$$

for $i = 1,\dots,n$, being the solution to the integral in (21). Here, $\alpha^2_{f_a}$ is the signal variance of the ath target dimension of $\mathcal{GP}_f$, a learned hyper-parameter of the SE covariance function, see (9).

Marginal Covariance Matrix

We now explicitly compute the entries of the corresponding covariance $\mathbf{C}^x_{t|t-1}$. Using the law of total covariance, we obtain for $a,b = 1,\ldots,D$

$$\begin{aligned}
\left(\Sigma^x_{t|t-1}\right)_{(ab)} &= \operatorname{cov}_{\underline{x}_{t-1},f,\underline{w}}\left\{x_t^{(a)}, x_t^{(b)} | \underline{z}_{1:t-1}\right\} \\
&= \mathrm{E}_{\underline{x}_{t-1}}\left\{\operatorname{cov}_{f,\underline{w}}\left\{f_a\left(\underline{x}_{t-1}\right) + w_a, f_b\left(\underline{x}_{t-1}\right) + w_b \middle| \underline{x}_{t-1}\right\} \middle| \underline{z}_{1:t-1}\right\} \\
&\quad + \operatorname{cov}_{\underline{x}_{t-1}}\left\{\mathrm{E}_{f_a}\left\{f_a\left(\underline{x}_{t-1}\right) \middle| \underline{x}_{t-1}\right\}, \mathrm{E}_{f_b}\left\{f_b\left(\underline{x}_{t-1}\right) \middle| \underline{x}_{t-1}\right\} \middle| \underline{z}_{1:t-1}\right\}, \qquad (25)
\end{aligned}$$

where we exploited in the last term that the system noise $\underline{w}$ has zero mean. Note that (25) is the sum of the covariance of (conditional) expected values and the expectation of a (conditional) covariance. We analyze these terms in the following.

The *covariance of the expectations* in (25) is

$$\int m_{f_a}\left(\underline{x}_{t-1}\right) m_{f_b}\left(\underline{x}_{t-1}\right) p\left(\underline{x}_{t-1}\right) \mathrm{d}\underline{x}_{t-1} - \left(\mu^x_{t|t-1}\right)_a \cdot \left(\mu^x_{t|t-1}\right)_b, \qquad (26)$$

where we used that $\mathrm{E}_f\left\{f\left(\underline{x}_{t-1}\right) \middle| \underline{x}_{t-1}\right\} = m_f\left(\underline{x}_{t-1}\right)$. With $\underline{\beta}^x_a = \left(\mathbf{K}_a + \sigma^2_{w_a}\mathbf{I}\right)^{-1} \underline{y}_a$ and $m_{f_a}\left(\underline{x}_{t-1}\right) = k_{f_a}\left(\mathbf{X}, \underline{x}_{t-1}\right)^{\mathrm{T}} \underline{\beta}^x_a$, we obtain

$$\operatorname{cov}_{\underline{x}_{t-1}}\left\{m_{f_a}\left(\underline{x}_{t-1}\right), m_{f_b}\left(\underline{x}_{t-1}\right) \middle| \underline{z}_{1:t-1}\right\} = \left(\underline{\beta}^x_a\right)^{\mathrm{T}} \cdot \mathbf{Q} \cdot \underline{\beta}^x_b - \left(\mu^x_{t|t-1}\right)_a \cdot \left(\mu^x_{t|t-1}\right)_b. \qquad (27)$$

Following [5], the entries of $\mathbf{Q} \in \mathbb{R}^{n \times n}$ are given as

$$\begin{aligned}
Q_{ij} &= \frac{k_{f_a}\left(\underline{x}_i, \underline{\mu}^x_{t-1|t-1}\right) k_{f_b}\left(\underline{x}_j, \underline{\mu}^x_{t-1|t-1}\right)}{\sqrt{|\mathbf{R}|}} \times \exp\left(\tfrac{1}{2} \underline{z}_{ij}^{\mathrm{T}} \mathbf{R}^{-1} \mathbf{C}^x_{t-1|t-1} \underline{z}_{ij}\right) \\
&= \frac{\exp\left(n^2_{ij}\right)}{\sqrt{|\mathbf{R}|}}, \qquad (28) \\
n^2_{ij} &= \log\left(\alpha^2_{f_a}\right) + \log\left(\alpha^2_{f_b}\right) - \tfrac{1}{2}\left(\underline{\zeta}_i^{\mathrm{T}} \Lambda_a^{-1} \underline{\zeta}_i + \underline{\zeta}_j^{\mathrm{T}} \Lambda_b^{-1} \underline{\zeta}_j - \underline{z}_{ij}^{\mathrm{T}} \mathbf{R}^{-1} \mathbf{C}^x_{t-1|t-1} \underline{z}_{ij}\right),
\end{aligned}$$

where we defined $\mathbf{R} \triangleq \mathbf{C}^x_{t-1|t-1}\left(\Lambda_a^{-1} + \Lambda_b^{-1}\right) + \mathbf{I}$, $\underline{\zeta}_i \triangleq \underline{x}_i - \underline{\mu}^x_{t-1|t-1}$, and $\underline{z}_{ij} \triangleq \Lambda_a^{-1}\underline{\zeta}_i + \Lambda_b^{-1}\underline{\zeta}_j$.

The *expected covariance* in (25) is given as

$$E_{\underline{x}_{t-1}}\{\mathrm{cov}_f\{f_a(\underline{x}_{t-1}), f_b(\underline{x}_{t-1})|\underline{x}_{t-1}\}|\underline{z}_{1:t-1}\} + \delta_{ab}\cdot\sigma^2_{w_a} \tag{29}$$

since the noise covariance matrix $\mathbf{C}_w$ is diagonal. Following our GP training assumption that different target dimensions do not covary if the input is deterministically given, (29) is only non-zero if $a = b$, i.e., (29) plays a role only for diagonal entries of $\mathbf{C}^x_{t|t-1}$. For these diagonal entries ($a = b$), the expected covariance in (29) is

$$\alpha^2_{f_a} - \mathrm{Tr}\left(\left(\mathbf{K}_{f_a} + \sigma^2_{w_a}\mathbf{I}\right)^{-1}\mathbf{Q}\right) + \sigma^2_{w_a}\,. \tag{30}$$

Hence, the desired marginal covariance matrix in (25) is

$$\left(\Sigma^x_{t|t-1}\right)_{ab} = \begin{cases}\text{Eq. (27)} + \text{Eq. (30)} & \text{, if } a = b\\ \text{Eq. (27)} & \text{, otherwise}\end{cases}\,. \tag{31}$$

We have now solved for the marginal distribution $p(\underline{x}_t|\underline{z}_{1:t-1})$ in (19). Since the approximate Gaussian filter distribution $p(\underline{x}_{t-1}|\underline{z}_{1:t-1}) = \mathcal{N}(\underline{\mu}^x_{t-1|t-1}, \mathbf{C}^x_{t-1|t-1})$ is also known, it remains to compute the cross-covariance $\mathbf{C}^x_{t-1,t|t-1}$ to fully determine the Gaussian approximation in (19).

3.2 Cross-Covariance

By the definition of a covariance and the system (1), the missing cross-covariance matrix $\mathbf{C}^x_{t-1,t|t-1}$ in (19) is

$$\mathbf{C}^x_{t-1,t|t-1} = \mathrm{E}_{\underline{x}_{t-1},f,\underline{w}_t}\left\{\underline{x}_{t-1}\cdot\left(f(\underline{x}_{t-1}) + \underline{w}_t\right)^{\mathrm{T}}|\underline{z}_{1:t-1}\right\} - \underline{\mu}^x_{t-1|t-1}\left(\underline{\mu}^x_{t|t-1}\right)^{\mathrm{T}}, \tag{32}$$

where $\underline{\mu}^x_{t-1|t-1}$ is the mean of the filter update at time step $t-1$ and $\underline{\mu}^x_{t|t-1}$ is the mean of the time update, see (20). Note that we explicitly average out the model uncertainty about f. Using the law of total expectations, we obtain

$$\begin{aligned}\mathbf{C}^x_{t-1,t|t-1} &= \mathrm{E}_{\underline{x}_{t-1}}\left\{\underline{x}_{t-1}\cdot\mathrm{E}_{f,\underline{w}_t}\{f(\underline{x}_{t-1}) + \underline{w}_t|\underline{x}_{t-1}\}^{\mathrm{T}}|\underline{z}_{1:t-1}\right\} - \underline{\mu}^x_{t-1|t-1}\left(\underline{\mu}^x_{t|t-1}\right)^{\mathrm{T}}\\ &= \mathrm{E}_{\underline{x}_{t-1}}\left\{\underline{x}_{t-1}\cdot m_f(\underline{x}_{t-1})^{\mathrm{T}}|\underline{z}_{1:t-1}\right\} - \underline{\mu}^x_{t-1|t-1}\left(\underline{\mu}^x_{t|t-1}\right)^{\mathrm{T}},\end{aligned} \tag{33}$$

where we used the fact that $\mathrm{E}_{f,\underline{w}_t}\{f(\underline{x}_{t-1}) + \underline{w}_t|\underline{x}_{t-1}\} = m_f(\underline{x}_{t-1})$ is the mean function of $\mathcal{GP}_f$, which models the mapping $\underline{x}_{t-1} \mapsto \underline{x}_t$, evaluated at $\underline{x}_{t-1}$. We thus obtain

$$\mathbf{C}^x_{t-1,t|t-1} = \int \underline{x}_{t-1} \cdot m_f(\underline{x}_{t-1})^\mathrm{T} p(\underline{x}_{t-1}|\underline{z}_{1:t-1})\,\mathrm{d}\underline{x}_{t-1} - \underline{\mu}^x_{t-1|t-1}\left(\underline{\mu}^x_{t|t-1}\right)^\mathrm{T}. \quad (34)$$

Writing $m_f(\underline{x}_{t-1})$ as a finite sum over kernels [25] and moving the integration into this sum, the integration in Eq. (34) turns into

$$\int \underline{x}_{t-1} m_{f_a}(\underline{x}_{t-1}) p(\underline{x}_{t-1}|\underline{z}_{1:t-1})\,\mathrm{d}\underline{x}_{t-1} \\ = \sum_{i=1}^n \beta^x_{a_i} \int \underline{x}_{t-1} k_{f_a}(\underline{x}_{t-1}, \underline{x}_i) p(\underline{x}_{t-1}|\underline{z}_{1:t-1})\,\mathrm{d}\underline{x}_{t-1}$$

for each state dimension $a = 1, \ldots, D$. With the SE covariance function k_SE defined in (9), we compute the integral analytically and obtain

$$\int \underline{x}_{t-1} m_{f_a}(\underline{x}_{t-1}) p(\underline{x}_{t-1}|\underline{z}_{1:t-1})\,\mathrm{d}\underline{x}_{t-1} \\ = \sum_{i=1}^n \beta^x_{a_i} \cdot c_3 \int \underline{x}_{t-1} \mathcal{N}(\underline{x}_i, \Lambda_a) \mathcal{N}\left(\underline{\mu}^x_{t-1|t-1}, \mathbf{C}^x_{t-1|t-1}\right) \mathrm{d}\underline{x}_{t-1}, \quad (35)$$

with $c_3^{-1} \triangleq \left(\alpha^2_{f_a}(2\pi)^{\frac{D}{2}}\sqrt{|\Lambda_a|}\right)^{-1}$ such that $k_{f_a}(\underline{x}_{t-1}, \underline{x}_i) = c_3\mathcal{N}(\underline{x}_{t-1}; \underline{x}_i, \Lambda_a)$. In the definition of c_3, $\alpha^2_{f_a}$ is a hyper-parameter of $\mathcal{GP}_f$ responsible for the variance of the latent function in dimension a. Using the definition of $\mathbf{S}$ in (24), the product of the two Gaussians in (35) results in a new (unnormalized) Gaussian $c_4^{-1}\mathcal{N}(\underline{x}_{t-1}; \underline{\psi}_i, \Psi)$ with

$$c_4^{-1} = (2\pi)^{-\frac{D}{2}} \left|\Lambda_a + \Sigma^x_{t-1|t-1}\right|^{-\frac{1}{2}} \times \exp\left(-\tfrac{1}{2}\left(\underline{x}_i - \underline{\mu}^x_{t-1|t-1}\right)^\mathrm{T} \mathbf{S}^{-1} \left(\underline{x}_i - \underline{\mu}^x_{t-1|t-1}\right)\right),$$

$$\Psi = \left(\Lambda_a^{-1} + \left(\Sigma^x_{t-1|t-1}\right)^{-1}\right)^{-1},$$

$$\underline{\psi}_i = \Psi \cdot \left(\Lambda_a^{-1}\underline{x}_i + \left(\Sigma^x_{t-1|t-1}\right)^{-1}\underline{\mu}^x_{t-1|t-1}\right).$$

Pulling all constants outside the integral in (35), the integral determines the expected value of the product of the two Gaussians, $\underline{\psi}_i$. For $a = 1, \ldots, D$, we obtain

$$\mathrm{E}\{\underline{x}_{t-1}\, x_{t_a} \big| \underline{z}_{1:t-1}\} = \sum_{i=1}^{n} c_3 c_4^{-1} \beta_{a_i}^x \underline{\psi}_i \ .$$

Using $c_3 c_4^{-1} = q_{a_i}^x$, see (23), and some matrix identities, we finally obtain

$$\mathrm{cov}_{\underline{x}_{t-1}, f, \underline{w}_t} \{\underline{x}_{t-1}, x_{t_a} \big| \underline{z}_{1:t-1}\} = \sum_{i=1}^{n} \beta_{a_i}^x q_{a_i}^x \mathbf{C}_{t-1|t-1}^x \left(\mathbf{C}_{t-1|t-1}^x + \Lambda_a\right)^{-1} \left(\underline{x}_i - \underline{\mu}_{t-1|t-1}^x\right) \qquad (36)$$

Thus, the joint covariance matrix of $p(\underline{x}_{t-1}, \underline{x}_t | \underline{z}_{1:t-1})$ and hence, the full Gaussian approximation in (19) is completely determined.

With the mean and the covariance of the joint distribution $p(\underline{x}_{t-1}, \underline{x}_t | \underline{z}_{1:t-1})$ given by (22), (31), (36), and the filter step, all necessary components are provided to compute the smoothing distribution $p(\underline{x}_t | \underline{z}_{1:T})$ analytically [7].

4 Simulations

In the following, we present results analyzing the robustness of state-of-the-art nonlinear filters (Section 4.1) and the performances of the corresponding smoothers (Section 4.2).

4.1 Filter Robustness

We consider the nonlinear stochastic dynamic system

$$x_t = \tfrac{x_{t-1}}{2} + \tfrac{25\, x_{t-1}}{1+x_{t-1}^2} + w_t \ , \qquad \text{with } w_t \sim \mathcal{N}\left(0, \sigma_w^2 = 0.2^2\right) , \qquad (37)$$

$$z_t = 5 \cdot \sin(x_t) + v_t \ , \qquad \text{with } v_t \sim \mathcal{N}\left(0, \sigma_v^2 = 0.2^2\right) , \qquad (38)$$

which is a modified version of the model used in [10, 15]. The system is modified in two ways: First, (37) does not contain a purely time-dependent term in the system, which would not allow for learning stationary transition dynamics. Second, we substituted a sinusoidal measurement function for the originally quadratic measurement function used by [15] and [10]. The sinusoidal measurement function increases the difficulty in computing the marginal distribution $p(\underline{z}_t | \underline{z}_{1:t-1})$ if the time update distribution $p(\underline{x}_t | \underline{z}_{1:t-1})$ is fairly uncertain: While the quadratic measurement function can only lead to bimodal distributions (assuming a Gaussian input distribution), the sinusoidal measurement function in (38) can lead to an arbitrary number of modes—for a broad input distribution.

The prior variance was set to $\sigma_0^2 = 0.5^2$, i.e., the initial uncertainty was fairly high. The system and measurement noises (see (37)–(38)) were relatively small considering the amplitudes of the system function and the measurement function. For the numerical analysis, a linear grid in the interval $[-3,3]$ of mean values $\left(\mu_0^x\right)_i$, $i = 1,\ldots,100$, was defined. Then, a single latent (initial) state $x_0^{(i)}$ was sampled from $p\left(x_0^{(i)}\right) = \mathcal{N}\left(\left(\mu_0^x\right)_i, \sigma_0^2\right)$, $i = 1,\ldots,100$.

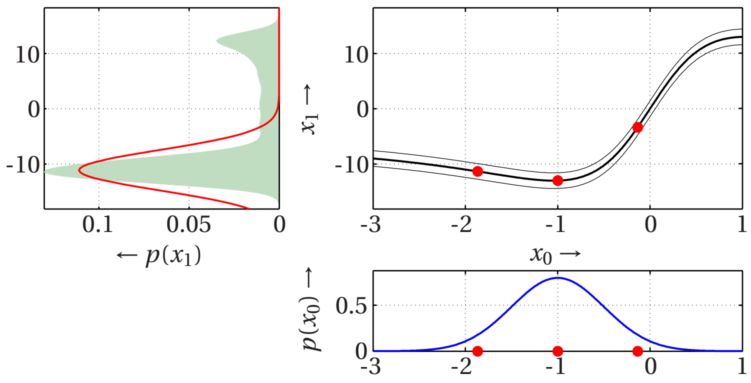
For the dynamic system in (37)–(38), we analyzed the robustness in a single filter step of the EKF, the UKF, the CKF, an SIR PF (sequential importance resampling particle filter) with 200 particles, the GP-UKF, and the GP-ADF against the ground truth, closely approximated by the Gibbs-filter [7]. Compared to the evaluation of longer trajectories, evaluating a single filter step makes it easier to analyze the robustness of individual filtering algorithms.

Table 1 summarizes the expected performances (root-mean-square error (rmse), mean-absolute error (mae), negative log-likelihood (nll)) of the EKF, the UKF, the CKF, the GP-UKF, the GP-ADF, the Gibbs-filter, and the SIR PF for estimating the latent state x. The results in the table are based on averages over 1,000 test runs and 100 randomly sampled start states per test run (see experimental setup). The table also reports the 95% standard error of the expected performances. Table 1 indicates that the GP-ADF is the most robust filter and statistically significantly outperforms all filters but the sampling-based Gibbs-filter and the SIR PF. The green color highlights a near-optimal Gaussian filter (Gibbs-filter) and the near-optimal particle filter. Amongst all other filters the GP-ADF is the closest Gaussian filter to the computationally expensive Gibbs-filter [7]. Note that the SIR PF is not a Gaussian filter and is able to express multi-modality in distributions. Therefore, its performance is typically better than the one of Gaussian filters. The difference between the SIR PF and a near-optimal Gaussian filter, the Gibbs-filter, is expressed in Table 1. The performance difference essentially depicts how much we lose by using a Gaussian filter instead of a particle filter. The nll values for the SIR PF are obtained by moment-matching the particles.

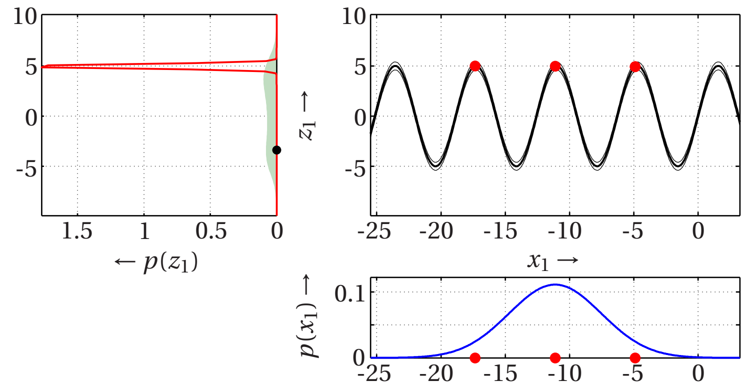
The poor performance of the EKF is due to linearization errors. The filters based on small sample approximations of densities (UKF, GP-UKF, CKF) suffer from the degeneracy of these approximations, which is illustrated in Figure 1. Note that the CKF uses a smaller set of cubature points than the UKF to determine

Table 1: Average filter performances (rmse, mae, nll) with standard errors (95% confidence interval) and p-values testing the hypothesis that the other filters are better than the GP-ADF using a one-sided t-test.

	rmse_x		mae_x		nll_x	
	average	p-value	average	p-value	average	p-value
EKF [20]	3.62 ± 0.212	4.1×10^{-2}	2.36 ± 0.176	0.38	$3.05 \times 10^{3} \pm 3.02 \times 10^{2}$	$< 10^{-4}$
UKF [13]	10.5 ± 1.08	$< 10^{-4}$	8.58 ± 0.915	$< 10^{-4}$	25.6 ± 3.39	$< 10^{-4}$
CKF [2]	9.24 ± 1.13	2.8×10^{-4}	7.31 ± 0.941	4.2×10^{-4}	$2.22 \times 10^{2} \pm 17.5$	$< 10^{-4}$
GP-UKF [16]	5.36 ± 0.461	7.9×10^{-4}	3.84 ± 0.352	3.3×10^{-3}	6.02 ± 0.497	$< 10^{-4}$
GP-ADF [6]	$\mathbf{2.85 \pm 0.174}$	—	$\mathbf{2.17 \pm 0.151}$	—	$\mathbf{1.97 \pm 6.55 \times 10^{-2}}$	—
Gibbs-filter [7]	$\mathbf{2.82 \pm 0.171}$	0.54	$\mathbf{2.12 \pm 0.148}$	0.56	$\mathbf{1.96 \pm 6.62 \times 10^{-2}}$	0.55
SIR PF	$\mathbf{1.57 \pm 7.66 \times 10^{-2}}$	1.0	$\mathbf{0.36 \pm 2.28 \times 10^{-2}}$	1.0	$\mathbf{1.03 \pm 7.30 \times 10^{-2}}$	1.0



(a) UKF time update $p(x_1|\varnothing)$, which misses out substantial probability mass of the true predictive distribution.



(b) UKF determines $p(z_1|\varnothing)$, which is too sensitive and cannot explain the actual measurement z_1 (black dot, left sub-figure).

Figure 1: Degeneracy of the unscented transformation (UT) underlying the UKF. Input distributions to the UT are the Gaussians in the sub-figures at the bottom in each panel. The functions the UT is applied to are shown in the top right sub-figures, i.e, the transition mapping (37) in (a) and the measurement mapping (38) in (b). Sigma points are marked by red dots. The predictive distributions are shown in the left sub-figures of each panel. The true predictive distributions are the shaded areas; the UT predictive distributions are the solid Gaussians. The predictive distribution of the time update in (a) equals the input distribution at the bottom of (b).

predictive distributions, which makes the CKF statistically even less robust than the UKF.

4.2 Smoother Robustness

We consider a pendulum tracking example taken from [6]. We evaluate the performances of four filters and smoothers, the EKF/EKS, the UKF/URTSS, the GP-UKF/GP-URTSS, the CKF/CKS, the Gibbs-filter/smoother, and the GP-ADF/GP-RTSS. The pendulum has mass $m = 1\,\mathrm{kg}$ and length $l = 1\,\mathrm{m}$. The state $\underline{x} = \left[\dot{\varphi}, \varphi\right]^{\mathrm{T}}$ of the pendulum is given by the angle φ (measured anti-clockwise from hanging down) and the angular velocity $\dot{\varphi}$. The pendulum can exert a constrained torque $u \in [-5,5]\,\mathrm{Nm}$. We assumed a frictionless system such that the transition function f is

$$f\left(\underline{x}_t, u_t\right) = \int_t^{t+\Delta_t} \begin{bmatrix} \frac{u(\tau) - 0.5\, mlg \sin\left(\varphi(\tau)\right)}{0.25\, ml^2 + I} \\ \dot{\varphi}(\tau) \end{bmatrix} \mathrm{d}\tau \,, \tag{39}$$

where I is the moment of inertia and g the acceleration of gravity. Then, the successor state

$$\underline{x}_{t+1} = \underline{x}_{t+\Delta_t} = f\left(\underline{x}_t, u_t\right) + \underline{w}_t \,, \tag{40}$$

was computed using an ODE solver for (39) with a zero-order hold control signal $u(\tau)$. In (40), we set $\mathbf{C}_w = \mathrm{diag}\left(0.5^2, 0.1^2\right)$. In our experiment, the torque was sampled randomly according to $u \sim \mathcal{U}[-5,5]\,\mathrm{Nm}$ and implemented using a zero-order-hold controller. Every time increment $\Delta_t = 0.2\,\mathrm{s}$, the state was measured according to

$$z_t = \arctan\left(\tfrac{-1 - l \sin(\varphi_t)}{0.5 - l \cos(\varphi_t)}\right) + v_t \,, \quad \text{with } \sigma_v^2 = 0.05^2 \,. \tag{41}$$

Note that the scalar measurement (41) solely depends on the angle. Thus, the full distribution of the latent state $\underline{x}$ had to be reconstructed using the cross-correlation information between the angle and the angular velocity.

Trajectories of length $T = 6\,\mathrm{s} = 30$ time steps were started from a state sampled from the prior $p\left(\underline{x}_0\right) = \mathcal{N}\left(\underline{\mu}_0, \mathbf{C}_0\right)$ with $\underline{\mu}_0 = \left[0, 0\right]^{\mathrm{T}}$ and $\mathbf{C}_0 = \mathrm{diag}\left(0.01^2, (\pi/16)^2\right)$. For each trajectory, GP models $\mathcal{GP}_f$ and $\mathcal{GP}_g$ were learned based on randomly generated data using either 250 or 20 data points.

Table 2 reports the expected values of the nll_x-measure for the EKF/EKS, the UKF/URTSS, the GP-UKF/GP-URTSS, the GP-ADF/GP-RTSS, and the CKF/CKS

Table 2: Expected filtering and smoothing performances (pendulum tracking) with 95% confidence intervals.

Filters	nll_x	Smoothers	nll_x
EKF [20]	$1.6\times10^2\pm29.1$	EKS [20]	$\mathbf{3.3\times10^2\pm60.5}$
UKF [13]	6.0 ± 3.02	URTSS [29]	$\mathbf{17.2\pm10.0}$
CKF [2]	28.5 ± 9.83	CKS [7]	$\mathbf{72.0\pm25.1}$
GP-UKF$_{250}$ [16]	4.4 ± 1.32	GP-URTSS$^\star_{250}$	$\mathbf{10.3\pm3.85}$
GP-ADF$_{250}$ [6]	$\mathbf{1.44\pm0.117}$	GP-RTSS$^\star_{250}$	$\mathbf{1.04\pm0.204}$
GP-ADF$_{20}$ [6]	6.63 ± 0.149	GP-RTSS$^\star_{20}$	6.57 ± 0.148

when tracking the pendulum over a horizon of 6 s, averaged over 1,000 runs. The $\star$ indicates a method developed in this paper. As in the example in Section 4.1, the nll_x-measure emphasizes the robustness of our proposed method: The GP-RTSS is the only method that consistently reduced the negative log-likelihood value compared to the corresponding filtering algorithm. Increasing the nll_x-values (red color in Table 2) occured when the filter distribution cannot explain the latent state/measurement, an example of which is given in Figure 1b. Even with only 20 training points, the GP-ADF/GP-RTSS outperformed the commonly used EKF/EKS, UKF/URTSS, CKF/CKS.

We experimented with even smaller signal-to-noise ratios. The GP-RTSS remains robust, while the other smoothers remain unstable.

5 Discussion and Conclusion

In this paper, we presented GP-RTSS, an analytic Rauch-Tung-Striebel smoother for GP dynamic systems, where the GPs with SE covariance functions are practical implementations of universal function approximators. We showed that the GP-RTSS is more robust to nonlinearities than state-of-the-art smoothers. There are two main reasons for this: First, the GP-RTSS relies neither on linearization (EKS) nor on density approximations (URTSS/CKS) to compute an optimal Gaussian approximation of the predictive distribution when mapping a Gaussian distribution through a nonlinear function. This property avoids incoherent estimates of the filtering and smoothing distributions as discussed in Sec 4.1. Second, GPs allow for more robust “system identification” than standard methods since they coherently represent uncertainties about the system and measurement functions at locations that have not been encountered in the data collection phase. The GP-RTSS is a robust smoother since it accounts for model uncertainties in a principled Bayesian way.

After training the GPs, which can be performed off-line, the computational complexity of the GP-RTSS (including filtering) is $\mathcal{O}\big(T\big(E^3 + n^2\big(D^3 + E^3\big)\big)\big)$ for a time series of length T. Here, n is the size of the GP training sets, and D and E are the dimensions of the state and the measurements, respectively. The computational complexity is due to the inversion of the D and E-dimensional covariance matrices, and the computation of the matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$ in (28), required for each entry of a D and E-dimensional covariance matrix. The computational complexity scales linearly with the number of time steps. The computational demand of classical Gaussian smoothers, such as the URTSS and the EKS is $\mathcal{O}\big(T\big(D^3 + E^3\big)\big)$. Although not reported here, we verified the computational complexity experimentally. Approximating the on-line computations of the GP-RTSS by numerical integration or grids scales poorly with increasing dimension. These problems already appear in the histogram filter [30]. By explicitly providing equations for the solution of the involved integrals, we show that numerical integration is not necessary and the GP-RTSS is a practical approach to filtering in GP dynamic systems.

Although the GP-RTSS is computationally more involved than the URTSS, the EKS, and the CKS, this does not necessarily imply that smoothing with the GP-RTSS is slower: function evaluations, which are heavily used by the EKS/CKS/URTSS are not necessary in the GP-RTSS (after training). In the pendulum example, repeatedly calling the ODE solver caused the EKS/CKS/URTSS to be slower than the GP-RTSS (with 250 training points) by a factor of two.

The increasing use of GPs for model learning in robotics and control will eventually require principled smoothing methods for GP models. To our best knowledge, the proposed GP-RTSS is the most principled GP-smoother since all computations can be performed analytically exactly, i.e., without function linearization or sigma/cubature point representation of densities, while exactly integrating out the model uncertainty induced by the GP distribution.

Code will be made publicly available at `http://mloss.org`.

Acknowledgements

This work was partially supported by ONR MURI grant N00014-09-1-1052, by Intel Labs, and by DataPath, Inc.

References

[1] Brian D. O. Anderson and John B. Moore. *Optimal Filtering.* Dover Publications, 2005.

[2] Ienkaran Arasaratnam and Simon Haykin. Cubature Kalman Filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, 2009.

[3] Karl J. **r**Aström. *Introduction to Stochastic Control Theory.* Dover Publications, Inc., 2006.

[4] Christopher G. Atkeson and Juan C. Santamaría. A Comparison of Direct and Model-Based Reinforcement Learning. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pages 3557–3564, 1997.

[5] Marc P. Deisenroth. *Efficient Reinforcement Learning using Gaussian Processes.* PhD thesis, Karlsruhe Institute of Technology (KIT), 2010.

[6] Marc P. Deisenroth, Marco F. Huber, and Uwe D. Hanebeck. Analytic Moment-based Gaussian Process Filtering. In *International Conference on Machine Learning*, pages 225–232, 2009.

[7] Marc P. Deisenroth and Henrik Ohlsson. A General Perspective on Gaussian Filtering and Smoothing: Explaining Current and Deriving New Algorithms. In *American Control Conference*, pages 1807–1812, 2011.

[8] Marc P. Deisenroth and Carl E. Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *International Conference on Machine Learning*, 2011.

[9] Marc P. Deisenroth, Carl E. Rasmussen, and Dieter Fox. Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning. In *Robotics: Science & Systems*, 2011.

[10] Arnaud Doucet, Simon J. Godsill, and Christophe Andrieu. On Sequential Monte Carlo Sampling Methods for Bayesian Filtering. *Statistics and Computing*, 10:197–208, 2000.

[11] Simon J. Godsill, Arnaud Doucet, and Mike West. Monte Carlo Smoothing for Nonlinear Time Series. *Journal of the American Statistical Association*, 99(465):438–449, 2004.

[12] Uwe D. Hanebeck. Optimal Filtering of Nonlinear Systems Based on Pseudo Gaussian Densities. In *Symposium on System Identification*, pages 331–336, 2003.

[13] Simon J. Julier and Jeffrey K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[14] Rudolf E. Kalman. A new Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME, Journal of Basic Engineering*, 82 (Series D)(1):35–45, 1960.

[15] Genshiro Kitagawa. Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[16] Jonathan Ko and Dieter Fox. GP-BayesFilters: Bayesian Filtering using Gaussian Process Prediction and Observation Models. *Autonomous Robots*, 27(1):75–90, 2009.

[17] Jus Kocijan, Roderick Murray-Smith, Carl E. Rasmussen, and Bojan Likar. Predictive Control with Gaussian Process Models. In *IEEE Region 8 Eurocon 2003: Computer as a Tool*, pages 352–356, 2003.

[18] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, February 2001.

[19] David J. C. MacKay. Introduction to Gaussian Processes. In *Neural Networks and Machine Learning*, volume 168, pages 133–165. Springer, 1998.

[20] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 141. Academic Press, Inc., 1979.

[21] Roderick Murray-Smith, Daniel Sbarbaro, Carl E. Rasmussen, and Agathe Girard. Adaptive, Cautious, Predictive Control with Gaussian Process Priors. In *The 13th IFAC Symposium on System Identification*, pages 1155–1160, 2003.

[22] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. Local Gaussian Process Regression for Real Time Online Model Learning. In *Advances in Neural Information Processing Systems*, pages 1193–1200. 2009.

[23] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[24] Joaquin Quiñonero-Candela, Agathe Girard, Jan Larsen, and Carl E. Rasmussen. Propagation of Uncertainty in Bayesian Kernel Models—Application to Multiple-Step Ahead Forecasting. In *International Conference on Acoustics, Speech and Signal Processing*, pages 701–704, 2003.

[25] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[26] H. E. Rauch, F. Tung, and C. T. Striebel. Maximum Likelihood Estimates of Linear Dynamic Systems. *AIAA Journal*, 3(8):1445–1450, August 1965.

[27] Sam T. Roweis and Zoubin Ghahramani. A Unifying Review of Linear Gaussian Models. *Neural Computation*, 11(2):305–345, 1999.

[28] Sam T. Roweis and Zoubin Ghahramani. *Kalman Filtering and Neural Networks*, chapter Learning Nonlinear Dynamical Systems using the EM Algorithm, pages 175–220. Wiley, 2001.

[29] Simo Särkkä. Unscented Rauch-Tung-Striebel Smoother. *IEEE Transactions on Automatic Control*, 53(3):845–849, 2008.

[30] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005.

Paper J

Recursive Gaussian Process Regression

Authors: Marco F. Huber

Edited version of the paper:

M. F. Huber. Recursive Gaussian Process Regression. In *Proceedings of the 38th International Conference on Acoustics, Sound, and Signal Processing (ICASSP)*, pages 3362–3366, Vancouver, BC, Canada, May 2013.

Recursive Gaussian Process Regression

Marco F. Huber

AGT International
Darmstadt, Germany
marco.huber@ieee.org

Abstract

For large data sets, performing Gaussian process regression is computationally demanding or even intractable. If data can be processed sequentially, the recursive regression method proposed in this paper allows incorporating new data with constant computation time. For this purpose two operations are performed alternating on a fixed set of so-called basis vectors used for estimating the latent function: First, inference of the latent function at the new inputs. Second, utilization of the new data for updating the estimate. Numerical simulations show that the proposed approach significantly reduces the computation time and at the same time provides more accurate estimates compared to existing on-line and/or sparse Gaussian process regression approaches.

1 Introduction

Gaussian processes (GPs) allow non-parametric learning of a regression function from noisy data. They can be considered Gaussian distributions over functions conditioned on the data [11]. In contrast to classical regression, GPs provide not only a regression function but also provide uncertainty estimates (error bars) depending on the noise and variability of the data.

Unfortunately, due to their non-parametric nature, GPs require computations that scale with $\mathcal{O}(n^3)$ for training, where n is the number of data points. In order to reduce the computational load, sparse approximations have been proposed in the recent years (see for example [7, 10, 14, 15, 17–19]). Typically, these approximations operate on a subset of size s of the training data, which reduces the computation load to $\mathcal{O}(s^2 \cdot n)$ for training. However, most of these approximations assume that the whole data set is available prior to the training and thus, training is performed *off-line* in a batch mode.

Only a few approaches have been proposed that allow sequential training of GPs for data that arrives *on-line*, e.g., from a time series. In [8] for instance, clusters on the incoming data points are identified and created sequentially. The assignment of data points to clusters and the number of clusters, however, depend a threshold value that is heavily application specific and requires careful tuning. For specific kernel functions, the approach proposed in [4] allows transforming GP regression into a Kalman filtering and smoothing problem that merely scales with $\mathcal{O}(n)$. Unfortunately, this approach so far is only applicable for one-dimensional inputs.

Similar to [4], the approach proposed in this paper considers GP training a Bayesian filtering problem. To allow for a large number of inputs of *arbitrary* dimension, the regression function is represented by means of a finite set of *basis vectors*. Training with incoming data, i.e., updating the mean and covariance estimate featured by the basis vectors, is performed *on-line* in a recursive fashion. Thus, after updating, the newly arrived data points can be discarded, while the estimate provided by the basis vectors is sufficient for prediction.

2 Problem Statement

For GP regression, it is assumed that a set of data $\mathcal{D} = \{(\underline{x}_1, y_1), \ldots, (\underline{x}_n, y_n)\}$ is drawn from the noisy process

$$y_i = g(\underline{x}_i) + \varepsilon \;,$$

where $\underline{x}_i \in \mathbb{R}^d$ are the inputs, $y_i \in \mathbb{R}$ are the observations or outputs, and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is zero-mean Gaussian noise with variance σ^2. For brevity reasons, $\mathbf{X}_\mathcal{D} \triangleq [\underline{x}_1, \ldots, \underline{x}_n]$ are all inputs and $\underline{y} \triangleq [y_1, \ldots, y_n]^\mathrm{T}$ are the corresponding observations in the following.

A GP is used to infer the latent function $g(.)$ from the data $\mathcal{D}$. The GP is completely defined by a mean function $m(\underline{x}) \triangleq \mathrm{E}\{g(\underline{x})\}$, which specifies the expected output value, and a positive semi-definite covariance function $k(\underline{x}, \underline{x}') \triangleq \mathrm{cov}\{g(\underline{x}), g(\underline{x}')\}$, which specifies the covariance between pairs of inputs and is often called a *kernel*. Typical examples are the zero mean function $m(\underline{x}) = 0$ and the squared exponential (SE) kernel

$$k(\underline{x}, \underline{x}') = \alpha^2 \cdot \exp\left(-\tfrac{1}{2}(\underline{x} - \underline{x}')^{\mathrm{T}} \Lambda^{-1} (\underline{x} - \underline{x}')\right) . \tag{1}$$

It is worth mentioning that the approach proposed in this paper holds for arbitrary mean and covariance functions. In (1), Λ is a diagonal matrix of the characteristic length-scales for each input dimension and α^2 is the variance of the latent function g. Such parameters of the mean and covariance functions together with the noise variance σ^2 are called the *hyperparameters* of the GP. In this paper, it is assumed that the hyperparameters are given and thus, are not learned from data.

As a GP forms a Gaussian distribution over functions, we can write $g(\underline{x}) \sim \mathcal{GP}\left(m(\underline{x}), k(\underline{x},\underline{x}')\right)$. For any finite set of inputs, the resulting distribution of the outputs is a multivariate Gaussian. For example, the distribution of the function value $g_* = g(\underline{x}_*)$ for an arbitrary test input $\underline{x}_*$ is a univariate Gaussian with mean and variance

$$\mu_g(\underline{x}_*) = \mathrm{E}\{g_*\} = m_* + \underline{k}_*^{\mathrm{T}} \mathbf{K}_x^{-1} (\underline{y} - \underline{m}) , \tag{2}$$

$$\sigma_g^2(\underline{x}_*) = \mathrm{var}\{g_*\} = k_{**} - \underline{k}_*^{\mathrm{T}} \mathbf{K}_x^{-1} \underline{k}_* , \tag{3}$$

respectively. Here, var{.} is the variance, $\mathbf{K}_x \triangleq \mathbf{K} + \sigma^2 \mathbf{I}$, $m_* \triangleq m(\underline{x}_*)$, $\underline{m} \triangleq m(\mathbf{X}_\mathcal{D})$, $\underline{k}_* \triangleq k(\mathbf{X}_\mathcal{D}, \underline{x}_*)$, $k_{**} \triangleq k(\underline{x}_*, \underline{x}_*)$, and $\mathbf{K} \triangleq k(\mathbf{X}_\mathcal{D},\mathbf{X}_\mathcal{D})$ is the kernel matrix.

For GP prediction, i.e., for calculating the distribution for a given set of test inputs according to (2) and (3), it is necessary to calculate the kernel matrix $\mathbf{K}$, to invert the matrix $\mathbf{K}_x$, and to multiply $\mathbf{K}_x$ with $\underline{k}_*$. Both the kernel matrix calculation and the multiplication scale with $\mathcal{O}(n^2)$, while the inversion even scales with $\mathcal{O}(n^3)$. Thus, for large data sets $\mathcal{D}$, storing the kernel matrix and solving all calculations is prohibitive. The following recursive GP regression approach aims at performing all calculations computationally very efficient on a set of $s \ll n$ so-called *basis vectors*.

Let $\mathbf{X} \triangleq [\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_s]$ be the matrix of locations of the basis vectors and $\underline{g} \triangleq g(\mathbf{X})$ the corresponding (unkown) values of the latent function. It is assumed that the basis vectors remain fixed for all processing steps $t = 0, 1, \ldots$ Since $g(\underline{x})$ is assumed to be a GP, the distribution $p_0(\underline{g}) = \mathcal{N}(\underline{g}; \underline{\mu}_0^g, \mathbf{C}_0^g)$ of $\underline{g}$ at the initial step $t = 0$ of the recursive processing is Gaussian with mean $\underline{\mu}_0^g \triangleq m(\mathbf{X})$

and covariance $\mathbf{C}_0^g \triangleq k(\mathbf{X},\mathbf{X})$. At an arbitrary step $t > 0$, new observations $\underline{y}_t \triangleq [y_{t,1}, y_{t,2}, \dots, y_{t,n_t}]^\mathrm{T}$ at inputs $\mathbf{X}_t \triangleq [\underline{x}_{t,1}, \underline{x}_{t,2}, \dots, \underline{x}_{t,n_t}]$ become available. The goal is now to calculate the posterior distribution $p_t\left(\underline{g}|\underline{y}_{1:t}\right)$, with $\underline{y}_{1:t} = \left(\underline{y}_1, \dots, \underline{y}_t\right)$, by updating the prior distribution of $\underline{g}$ at step $t-1$

$$p_{t-1} \triangleq p_{t-1}\left(\underline{g}|\underline{y}_{1:t-1}\right) = \mathcal{N}\left(\underline{g}; \underline{\mu}_{t-1}^g, \mathbf{C}_{t-1}^g\right) \tag{4}$$

with the new observations $\underline{y}_t$.

3 Recursive Processing

One might think of exploiting (2) and (3) for incorporating the new observations. This however, is not suitable for recursive processing for mainly two reasons. Firstly, the latest estimate of the latent function in terms of the distribution p_{t-1} or the mean $\underline{\mu}_{t-1}^g$ and covariance $\mathbf{C}_{t-1}^g$ is not utilized. Secondly, no correlation or cross-covariance between $\mathbf{X}$ and $\mathbf{X}_t$ is provided, which however is of paramount importance for updating p_{t-1}. Instead, for deriving a recursive algorithm, the desired posterior distribution is expanded according to

$$p_t = c_t \int \underbrace{p_t\left(\underline{y}_t|\underline{g}, \underline{g}_t\right) \cdot p_{t-1}\left(\underline{g}, \underline{g}_t|\underline{y}_{1:t-1}\right)}_{= p_t\left(\underline{g}, \underline{g}_t|\underline{y}_{1:t}\right)} \mathrm{d}\underline{g}_t \tag{5}$$

by applying Bayes' law and by integrating out $\underline{g}_t \triangleq g(\mathbf{X}_t)$ from the joint distribution $p_t\left(\underline{g}, \underline{g}_t|\underline{y}_{1:t}\right)$. Here, c_t is a normalization constant. Based on (5), calculating the posterior distribution can be performed in two steps: I. *Inference*, i.e., calculating the joint prior $p_{t-1}\left(\underline{g}, \underline{g}_t|\underline{y}_{1:t-1}\right)$ given the prior p_{t-1} in (4). II. *Update*, i.e., updating the joint prior with the observations $\underline{y}_t$ and integrating out $\underline{g}_t$.

3.1 Inference

In order to determine the joint prior $p_{t-1}\left(\underline{g},\underline{g}_t|\underline{y}_{1:t-1}\right)$, it is important to emphasize that the joint distribution $p\left(\underline{g},\underline{g}_t\right)$ is Gaussian with mean and covariance

$$\underline{\mu} = \begin{bmatrix} m(\mathbf{X}) \\ m(\mathbf{X}_t) \end{bmatrix} \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} k(\mathbf{X},\mathbf{X}) & k(\mathbf{X},\mathbf{X}_t) \\ k(\mathbf{X}_t,\mathbf{X}) & k(\mathbf{X}_t,\mathbf{X}_t) \end{bmatrix} , \tag{6}$$

respectively. This follows from the fact that $g(.)$ is a GP and any finite representation of this GP yields a Gaussian distribution. Thus, the joint prior can be written as

$$\begin{aligned} p_{t-1}\left(\underline{g},\underline{g}_t|\underline{y}_{1:t-1}\right) &\approx p\left(\underline{g}_t|\underline{g}\right)\cdot p_{t-1} \\ &= \mathcal{N}\left(\underline{g}_t;\underline{\mu}_t^p,\mathbf{B}\right)\cdot\mathcal{N}\left(\underline{g};\underline{\mu}_{t-1}^g,\mathbf{C}_{t-1}^g\right) , \end{aligned} \tag{7}$$

with

$$\begin{aligned} \underline{\mu}_t^p &= m(\mathbf{X}_t) + \mathbf{J}_t\cdot\left(\underline{\mu}_{t-1}^g - m(\mathbf{X})\right) , \\ \mathbf{B} &= k(\mathbf{X}_t,\mathbf{X}_t) - \mathbf{J}_t\cdot k(\mathbf{X},\mathbf{X}_t) , \\ \mathbf{J}_t &= k(\mathbf{X}_t,\mathbf{X})\cdot k(\mathbf{X},\mathbf{X})^{-1} . \end{aligned}$$

The first equality in (7) follows from assuming that $\underline{g}_t$ is conditionally independent of the past observations $\underline{y}_{1:t-1}$ given $\underline{g}$. Hence, the conditional distribution $p\left(\underline{g}_t|\underline{g}\right)$ is Gaussian and results from the joint distribution $p\left(\underline{g},\underline{g}_t\right)$ in (6) by conditioning on $\underline{g}$ (see for example Chapter 2.6 in [5]).

For solving the product in (7), at first the exponential functions of both Gaussians are considered. By defining $\underline{f} \triangleq \underline{g}_t - m(\mathbf{X}_t) + \mathbf{J}_t\cdot m(\mathbf{X})$ and utilizing the result of Lemma 1 in Appendix A, the sum of both exponents can be transformed into

$$\begin{aligned} -\tfrac{1}{2}\left(\left(\underline{f}-\mathbf{J}_t\cdot\underline{g}\right)^{\mathrm{T}}\mathbf{B}^{-1}\left(\underline{f}-\mathbf{J}_t\cdot\underline{g}\right)+\left(\underline{g}-\underline{\mu}_{t-1}^g\right)\left(\mathbf{C}_{t-1}^g\right)^{-1}\left(\underline{g}-\underline{\mu}_{t-1}^g\right)\right) = \\ -\tfrac{1}{2}\left(\left(\underline{g}-\underline{d}\right)^{\mathrm{T}}\mathbf{D}^{-1}\left(\underline{g}-\underline{d}\right)+\underbrace{\left(\underline{f}-\mathbf{J}_t\cdot\underline{\mu}_{t-1}^g\right)}_{=\underline{g}_t-\underline{\mu}_t^p}\left(\mathbf{C}_t^p\right)^{-1}\left(\underline{f}-\mathbf{J}_t\cdot\underline{\mu}_{t-1}^g\right)\right) , \end{aligned} \tag{8}$$

with

$$\begin{aligned}\underline{d} &\triangleq \underline{\mu}^g_{t-1} + \mathbf{D}\mathbf{J}_t^{\mathrm{T}}\mathbf{B}^{-1}\left(\underline{f} - \mathbf{J}_t \cdot \underline{\mu}^g_{t-1}\right) = \underline{\mu}^g_{t-1} + \mathbf{D}\mathbf{J}_t^{\mathrm{T}}\mathbf{B}^{-1}\left(\underline{g}_t - \underline{\mu}^p_t\right) , \\ \mathbf{D}^{-1} &\triangleq \mathbf{J}_t^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{J}_t + \left(\mathbf{C}^g_{t-1}\right)^{-1} , \\ \mathbf{C}^p_t &\triangleq \mathbf{B} + \mathbf{J}_t\mathbf{C}^g_{t-1}\mathbf{J}_t^{\mathrm{T}} , \end{aligned} \tag{9}$$

where $\underline{\mu}^p_t$ and $\mathbf{C}^p_t$ are the respective mean and covariance of $\underline{g}_t$. The inverse of (9) given by

$$\left(\mathbf{C}^p_t\right)^{-1} = \mathbf{B}^{-1} - \mathbf{B}^{-1}\mathbf{J}_t\mathbf{D}\mathbf{J}_t^{\mathrm{T}}\mathbf{B}^{-1} \tag{10}$$

results from applying the Woodbury formula [9]. By employing (10) together with some basic algebraic transformations, (8) can be rearranged to

$$\begin{aligned} -\tfrac{1}{2}\Bigg(&\left(\underline{g} - \underline{d}\right)^{\mathrm{T}}\mathbf{D}^{-1}\left(\underline{g} - \underline{d}\right) + \left(\underline{g}_t - \underline{\mu}^p_t\right)^{\mathrm{T}}\mathbf{B}^{-1}\left(\underline{g}_t - \underline{\mu}^p_t\right) - \\ &\left(\underline{g}_t - \underline{\mu}^p_t\right)^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{J}_t\mathbf{D}\mathbf{J}_t^{\mathrm{T}}\mathbf{B}^{-1}\left(\underline{g}_t - \underline{\mu}^p_t\right)\Bigg) = -\tfrac{1}{2}\Bigg(\left(\underline{g} - \underline{\mu}^g_{t-1}\right)\mathbf{D}^{-1}\left(\underline{g} - \underline{\mu}^g_{t-1}\right) - \\ &2\left(\underline{g} - \underline{\mu}^g_{t-1}\right)^{\mathrm{T}}\mathbf{J}_t^{\mathrm{T}}\mathbf{B}^{-1}\left(\underline{g}_t - \underline{\mu}^p_t\right) + \left(\underline{g}_t - \underline{\mu}^p_t\right)^{\mathrm{T}}\mathbf{B}^{-1}\left(\underline{g}_t - \underline{\mu}^p_t\right)\Bigg) . \end{aligned}$$

The right-hand term corresponds to the exponent of a joint Gaussian density $p_{t-1}\left(\underline{g}, \underline{g}_t | \underline{y}_{1:t-1}\right)$ of $\underline{g}$ and $\underline{g}_t$ with mean and (inverse) covariance

$$\underline{q} = \begin{bmatrix} \underline{\mu}^g_{t-1} \\ \underline{\mu}^p_t \end{bmatrix} \quad \text{and} \quad \mathbf{Q}^{-1} = \begin{bmatrix} \mathbf{D}^{-1} & -\mathbf{J}_t^{\mathrm{T}}\mathbf{B}^{-1} \\ -\mathbf{B}^{-1}\mathbf{J}_t & \mathbf{B}^{-1} \end{bmatrix} = \begin{bmatrix} \mathbf{C}^g_{t-1} & \mathbf{C}^g_{t-1}\mathbf{J}_t^{\mathrm{T}} \\ \mathbf{J}_t\mathbf{C}^g_{t-1} & \mathbf{C}^p_t \end{bmatrix}^{-1} , \tag{11}$$

respectively, where the covariance $\mathbf{Q}$ on the right-hand side of (11) is obtained from applying Lemma 2 in Appendix A on $\mathbf{Q}^{-1}$.

Besides the exponential functions, the multiplication in (7) also comprises the product of the normalization factors of both Gaussians, which can be simplified according to

$$\begin{aligned} \frac{1}{\sqrt{|2\pi\mathbf{B}|}} \cdot \frac{1}{\sqrt{\left|2\pi\mathbf{C}^g_{t-1}\right|}} &= \frac{1}{\sqrt{(2\pi)^{(n_t+m)} \cdot |\mathbf{B}| \cdot \left|\mathbf{C}^g_{t-1}\right|}} \\ &\overset{(a)}{=} \frac{1}{\sqrt{(2\pi)^{(n_t+m)} \cdot \left|\mathbf{C}^g_{t-1}\right| \cdot \Big|\underbrace{\mathbf{B} + \mathbf{J}_t\mathbf{C}^g_{t-1}\mathbf{J}_t^{\mathrm{T}}}_{=\mathbf{C}^p_t} - \mathbf{J}_t\mathbf{C}^g_{t-1}\left(\mathbf{C}^g_{t-1}\right)^{-1}\mathbf{C}^g_{t-1}\mathbf{J}_t^{\mathrm{T}}\Big|}} \overset{(b)}{=} \frac{1}{\sqrt{|2\pi\mathbf{Q}|}} , \end{aligned}$$

where (a) results from adding the zero $\mathbf{J}_t\mathbf{C}_{t-1}^g\mathbf{J}_t^{\mathrm{T}} - \mathbf{J}_t\mathbf{C}_{t-1}^g\mathbf{J}_t^{\mathrm{T}} = \mathbf{0}$ and (b) results from applying (24).

3.2 Update

Given the result of the previous section that the joint prior in (7) is a Gaussian $\mathcal{N}(\underline{q}, \mathbf{Q})$, the next step is to perform the update and marginalization in (5). For this purpose, (5) is rearranged to

$$p_t = \int \underbrace{c_t \cdot p_t\left(\underline{y}_t | \underline{g}_t\right) \cdot \overbrace{p_{t-1}\left(\underline{g}_t | \underline{y}_{1:t-1}\right) \cdot p_{t-1}\left(\underline{g} | \underline{g}_t, \underline{y}_{1:t-1}\right)}^{= p_{t-1}\left(\underline{g}, \underline{g}_t | \underline{y}_{1:t-1}\right)}}_{= p_t\left(\underline{g}_t | \underline{y}_{1:t}\right) \text{ (Kalman filter)}} \mathrm{d}\underline{g}_t \tag{12}$$

under consideration that $\underline{g}$ is not observed and thus, $p_t\left(\underline{y}_t | \underline{g}_t\right)$ is independent of $\underline{g}$. Since $p_t\left(\underline{y}_t | \underline{g}_t\right) = \mathcal{N}\left(\underline{y}_t; \underline{g}_t, \sigma^2\mathbf{I}\right)$ and $p_{t-1}\left(\underline{g}_t | \underline{y}_{1:t-1}\right) = \mathcal{N}\left(\underline{g}_t; \underline{\mu}_t^p, \mathbf{C}_t^p\right)$ are both Gaussian, $\underline{g}_t$ can be updated easily via a *Kalman filter* update step. Updating $\underline{g}$ and integrating out $\underline{g}_t$ is then performed simultaneously.

Applying the well-known Kalman filter update equations yields $p_t\left(\underline{g}_t | \underline{y}_{1:t}\right) = \mathcal{N}\left(\underline{g}_t; \underline{\mu}_t^e, \mathbf{C}_t^e\right)$ with mean and covariance

$$\underline{\mu}_t^e = \underline{\mu}_t^p + \mathbf{G}_t \cdot \left(\underline{y}_t - \underline{\mu}_t^p\right), \tag{13}$$

$$\mathbf{C}_t^e = \mathbf{C}_t^p - \mathbf{G}_t\mathbf{C}_t^p, \tag{14}$$

respectively, where $\mathbf{G}_t = \mathbf{C}_t^p \cdot \left(\mathbf{C}_t^p + \sigma^2\mathbf{I}\right)^{-1}$ is the Kalman gain. The multiplication of the two Gaussians $p_t\left(\underline{g}_t | \underline{y}_{1:t}\right)$ and $p_{t-1}\left(\underline{g} | \underline{g}_t, \underline{y}_{1:t-1}\right)$ in (12) again results in a joint Gaussian distribution of $\underline{g}$ and $\underline{g}_t$ with mean and covariance

$$\underline{\mu}_t = \begin{bmatrix} \underline{\mu}_t^g \\ \underline{\mu}_t^e \end{bmatrix} \quad \text{and} \quad \mathbf{C}_t = \begin{bmatrix} \mathbf{C}_t^g & \mathbf{L}_t\mathbf{C}_t^e \\ \mathbf{C}_t^e\mathbf{L}_t^{\mathrm{T}} & \mathbf{C}_t^e \end{bmatrix}$$

respectively, where $\mathbf{L}_t \triangleq \mathbf{C}^g_{t-1}\mathbf{J}^\mathrm{T}_t \left(\mathbf{C}^p_t\right)^{-1}$ and

$$\underline{\mu}^g_t = \underline{\mu}^g_{t-1} + \mathbf{L}_t \cdot \left(\underline{\mu}^e_t - \underline{\mu}^p_t\right) , \tag{15}$$

$$\mathbf{C}^g_t = \mathbf{C}^g_{t-1} + \mathbf{L}_t \cdot \left(\mathbf{C}^e_t - \mathbf{C}^p_t\right) \cdot \mathbf{L}^\mathrm{T}_t . \tag{16}$$

However, since we are merely interested in obtaining the distribution $p_t = \mathcal{N}\left(\underline{g}; \underline{\mu}^g_t, \mathbf{C}^g_t\right)$, i.e., updating the latent function at the basis vectors $\mathbf{X}$ in order to keep the memory and computational complexity bounded over time, $\underline{g}_t$ is integrated out. This corresponds to neglecting the mean $\underline{\mu}^e_t$ and covariance $\mathbf{C}^e_t$ of $\underline{g}_t$ as well as the cross-covariance $\mathbf{L}_t\mathbf{C}^e_t$.

3.3 Summary

Putting all together, at steps $t = 1, 2, \ldots$ the proposed approach recursively processes observations $\underline{y}_t$ at the inputs $\mathbf{X}_t$ by means of the following set of equations:

Inference

$$\mathbf{J}_t = k\left(\mathbf{X}_t, \mathbf{X}\right) \cdot k\left(\mathbf{X}, \mathbf{X}\right)^{-1} , \tag{17}$$

$$\underline{\mu}^p_t = m\left(\mathbf{X}_t\right) + \mathbf{J}_t \cdot \left(\underline{\mu}^g_{t-1} - m\left(\mathbf{X}\right)\right) , \tag{18}$$

$$\mathbf{C}^p_t = k\left(\mathbf{X}_t, \mathbf{X}_t\right) + \mathbf{J}_t \cdot \left(\mathbf{C}^g_{t-1} - k(\mathbf{X}, \mathbf{X})\right) \cdot \mathbf{J}^\mathrm{T}_t , \tag{19}$$

Update

$$\tilde{\mathbf{G}}_t = \mathbf{C}^g_{t-1}\mathbf{J}^\mathrm{T}_t \cdot \left(\mathbf{C}^p_t + \sigma^2\mathbf{I}\right)^{-1} , \tag{20}$$

$$\underline{\mu}^g_t = \underline{\mu}^g_{t-1} + \tilde{\mathbf{G}}_t \cdot \left(\underline{y}_t - \underline{\mu}^p_t\right) , \tag{21}$$

$$\mathbf{C}^g_t = \mathbf{C}^g_{t-1} - \tilde{\mathbf{G}}_t\mathbf{J}_t\mathbf{C}^g_{t-1} . \tag{22}$$

This recursion commences from the initial mean $\underline{\mu}^g_0 \triangleq m\left(\mathbf{X}\right)$ and covariance $\mathbf{C}^g_0 \triangleq k\left(\mathbf{X}, \mathbf{X}\right)$ of $\underline{g}$. The updated mean (21) and covariance (22) result from substituting $\underline{\mu}^e_t$ in (15) with (13) and $\mathbf{C}^e_t$ in (16) with (14), respectively, where $\tilde{\mathbf{G}}_t = \mathbf{L}_t \cdot \mathbf{G}_t$. The effect of both operations, namely inference and update, is illustrated exemplary in Figure 1.

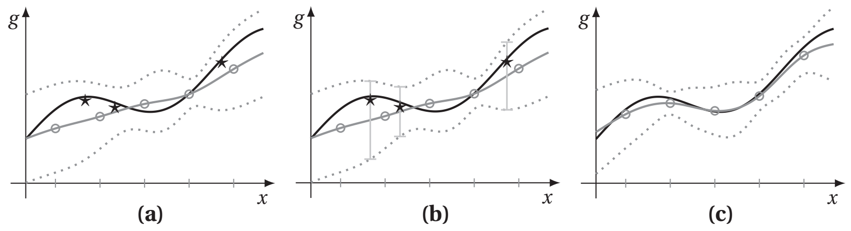


Figure 1: (a) The black line indicates the true function g, while the gray solid and dotted lines represent the mean and variance of recursive GP. The circles indicate the location of the basis vectors $\mathbf{X}$ (x-axis) and their mean values $\underline{\mu}^g$ (y-axis). The stars indicate new observations. (b) Inferring the mean and covariance of g at the locations of the new observations from the current recursive GP estimate. (c) Updating the GP with the new observations gives an improved estimate of the true function.

4 Discussion

A close inspection of the inference step shows that it has the same structure as the backward pass of the *Rauch-Tung-Striebel* (RTS) smoother [12]. In contrast to classical RTS smoothing, the inference step operates in the input domain and not in the time domain. It predicts the function value $g(\underline{x}_*)$ at any (test) input $\underline{x}_*$ given all information acquired so far and thus, is dual to the prediction (2), (3) of a classical GP.

So far, it was assumed that the set of basis vectors is fixed. The inference step, however, can also be utilized for introducing new basis vectors. This might be of interest in locations where the current estimate of the latent function is inaccurate. By replacing $\mathbf{X}_t$ with $[\mathbf{X}, \mathbf{X}']$, the inference step provides the initial mean and covariance as well as the cross-covariance between the new basis vectors and the old ones.

The computations of the inference step scale with $\mathcal{O}(s^2 \cdot n_t)$ due to calculating $\mathbf{J}_t$ in (17), where n_t is the number of observations at step t. Here, the inversion of the kernel matrix $k(\mathbf{X}, \mathbf{X})$ is computationally unproblematic, as it has to be calculated only once at step $t = 0$. Once the gain matrix $\mathbf{J}_t$ is calculated, predictions for a single test input are in $\mathcal{O}(s)$ (mean) and $\mathcal{O}(s^2)$ (covariance). Assuming that all observations are processed at once, predictions of the recursive GP are as complex as predictions of sparse GP approaches relying on a representation comparable to the basis vectors, e.g., pseudo-inputs in [15] or subset of regressors in [18]. In this case, the complexity is in $\mathcal{O}\big(s^2 \cdot n\big)$ for initialization as well as $\mathcal{O}(s)$ (mean) and $\mathcal{O}\big(s^2\big)$ (covariance) for predictions, respectively. In contrast to most sparse GP approaches, the proposed method can process new observations on-line.

The update step scales with $\mathcal{O}\big(n_t \cdot s^2\big)$, where the complexity results from matrix multiplications for which more efficient algorithms exist, e.g., Strassen's algorithm [16]. The inversion in (20) again is not critical as the affected matrix is of size $n_t \times n_t$, where typically $n_t \ll s$.

5 Simulation Examples

The proposed approach is compared to existing on-line and/or sparse GP approaches: a full GP (named FGP in the following), the on-line approaches local GP ([8], LGP) and sparse on-line GP ([2], SOGP), as well as the sparse (but off-line) approaches Bayesian committee machine ([17], BCM), subset of regressors ([18], SRM), and sparse GP using pseudo-inputs ([15], SGP). Further, our approach is applied in three different modes: updating/training the basis vectors with every observation (RGP_1), with a batch of 10 successive observations (RGP_{10}), or with a batch of all observations (RGP_{all}). In the latter case, RGP becomes an off-line algorithm. For training the on-line algorithms (LGP, SOGP, RGP_1, and RGP_{10}), the training data is presented sequentially, while for the remaining off-line algorithms, the training data is processed in a batch. All GP methods are implemented in MATLAB, where the GPML toolbox[1] is utilized for FGP and SGP.

At first, the one-dimensional nonlinear function

$$y = \tfrac{x}{2} + \tfrac{25 \cdot x}{1+x^2} \cdot \cos(x) + \varepsilon \quad , \quad \varepsilon \sim \mathcal{N}(0, 0.1) \tag{23}$$

is considered as an example. It is similar to the growth model proposed in [6]. To train the GPs, the inputs x_i, $i = 1, \ldots, n$ with $n \in \{50, 100, 150, 200\}$, are sampled uniform at random from the interval $[-10, 10]$. For testing, 200 input-observation pairs are considered. All GPs except of FGP use some sparse representation consisting of s elements (basis vectors, pseudo-inputs, clusters, etc.), where

1 `http://www.gaussianprocess.org/gpml/code`

$s = 20$ and $s = 40$ are considered. In case of the RGPs, the basis vectors are placed equidistant on the interval $[-10, 10]$.

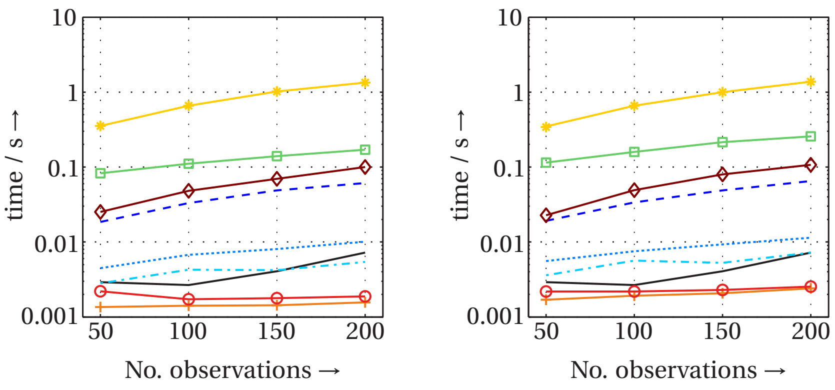
In the second example, the satellite observations of the Global Monitoring for Environment and Security program (GMES)[2] are considered. The inputs are the two-dimensional measurement locations and the observations are the particulate matter (PM_{10}) measurements at these locations. The data set comprises 10,000 elements and was recorded at 10th of October 2011. For training, $n \in \{500,$ 1,000, 1,500, 2,000$\}$ elements and for testing 1,000 elements are selected randomly from the data set. For the sparse representations both $s = 25$ and $s = 100$ elements are considered. The basis vectors of the recursive GPs are placed on a regular grid.

For both examples, a zero mean function and the SE kernel (1) are used. All GP methods use the same hyperparameters. For comparison, two performance criteria are considered: the total runtime in seconds comprising training and testing. Further, the negative log-likelihood (nll) of the predicted observations at the test inputs. For both a lower value indicates a better performance, where the nll penalizes uncertainty and inconsistency (prediction error). Regarding nll, FGP acts as the lower bound for all other methods. The averaged results of 100 simulation runs are depicted in Figure 2 and Figure 3 for the first and second example, respectively.

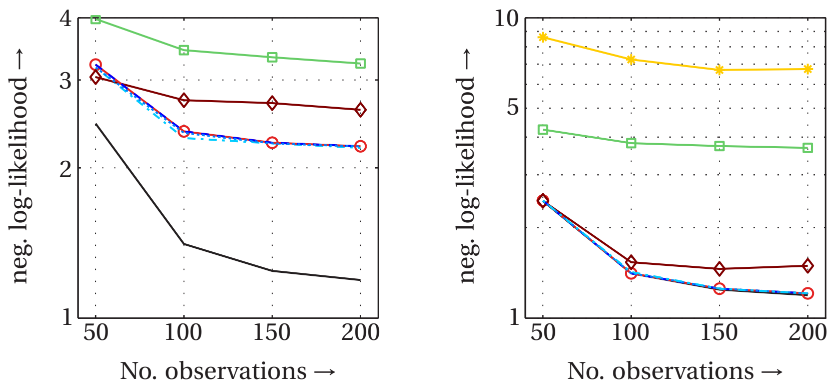
Among all on-line GP methods RGP_1 and RGP_{10} are the fastest and the most accurate. However, RGP_1 has a significantly higher runtime as RGP_{10} due to some overhead when performing inference and update for every single observation.

While BCM cannot compete with RGP regarding runtime and nll, SRM is faster but has a by far higher prediction error and provides too low variance values, i.e., SRM is inaccurate and at the time too confident about its predictions. SGP is as accurate as RGP for the first example and slightly worse for the second example. Regarding runtime, SGP is much faster. The proposed RGP however, is an on-line method that processes all observations only once. Operating SGP in an on-line fashion would require to revisit all observations acquired so far for each new input in order to provide updated pseudo-inputs. This of course would lead to a much higher computation time compared to RGP. Here, RGP clearly benefits from its recursive structure, which is not present for SGP.

2 `http://www.gmes.info/` - Data accessible via `ftp://data-portal.ecmwf.int/`



(a) Average runtime for $s = 20$ (left) and $s = 40$ (right).



(b) Average nll for $s = 20$ (left) and $s = 40$ (right).

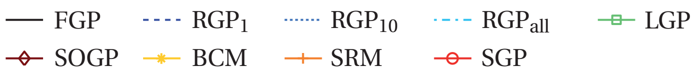
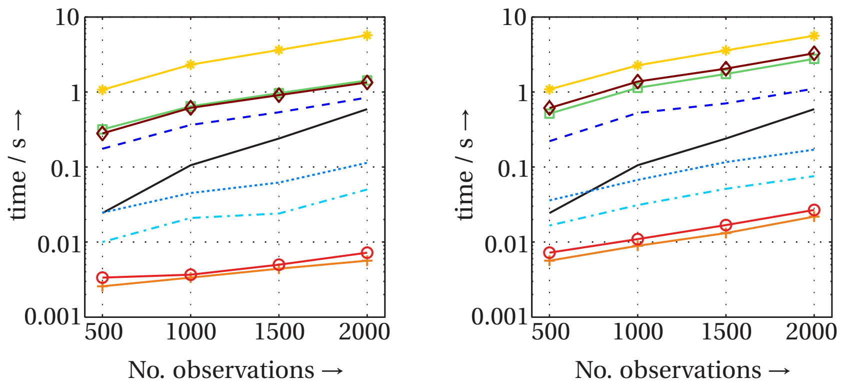
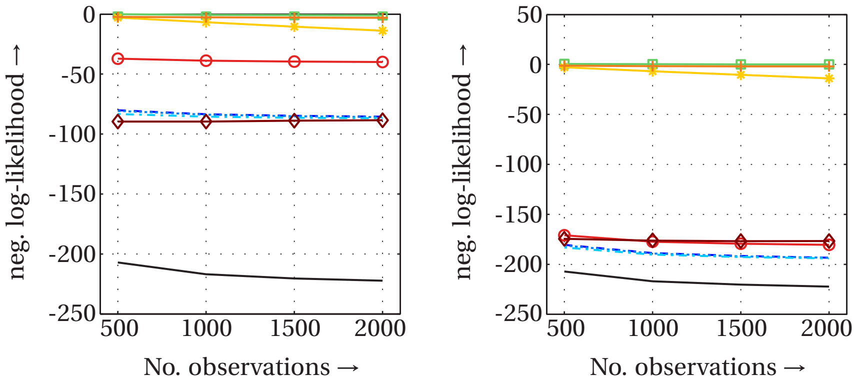


Figure 2: Average runtime and nll of first example according to (23) for different numbers of observations n and different numbers of sparse elements s. The average nll values are increased by one in order to have positive values and allow for a log-scale plot. In the left plot of (b), SRM and BCM have average values higher than four and thus, are not shown. In the right plot of (b), SRM is not depicted as its average values are significantly larger than 10.



(a) Average runtime for $s = 25$ (left) and $s = 100$ (right).



(b) Average nll for $s = 25$ (left) and $s = 100$ (right).

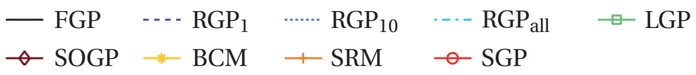


Figure 3: Average runtime and nll of second example (GMES) for different numbers of observations n and different numbers of sparse elements s.

For few observations, RGP is slower than FGP due to the overhead of managing and updating the basis vectors. However, RGP benefits from a large data set as considered in the second example. Here, RGP_{all} has a lower runtime compared to FGP since the prediction is less costly. Furthermore, RGP_{10} also becomes faster from 1,000 observations on. The same behavior is expected for RGP_1 for training data sets with more than 8,000 observations. If the number of basis vectors is sufficiently high, RGP can even have nll values similar to FGP.

6 Conclusions

The novel on-line Gaussian process regression approach proposed in this paper relies on a set of basis vectors that is updated recursively with new observations. The number of basis vectors and thus, the computation time for updating or prediction remains constant. Compared to existing on-line GP approaches, the proposed one is computationally more efficient and provides a higher prediction accuracy.

Future work is devoted to on-line adjustment of the hyperparameters. Furthermore, an extension towards multi-output GPs (see e.g. [1]) is intended.

A Useful Lemmas

The derivation of the solution for both inference step and update step are based on the following results.

Lemma 1 (Combination of Quadratic Forms) *If* **B** *and* **C** *are symmetric and positive definite matrices, then*

$$\left(\underline{a}-\mathbf{A}\underline{x}\right)^{\mathrm{T}}\mathbf{B}^{-1}\left(\underline{a}-\mathbf{A}\underline{x}\right)+\left(\underline{b}-\underline{x}\right)^{\mathrm{T}}\mathbf{C}^{-1}\left(\underline{b}-\underline{x}\right) = \left(\underline{x}-\underline{y}\right)^{\mathrm{T}}\mathbf{D}^{-1}\left(\underline{x}-\underline{y}\right)+r ,$$

where

$$\begin{aligned}\underline{y} &= \underline{b}+\mathbf{D}\mathbf{A}^{\mathrm{T}}\mathbf{B}^{-1}\left(\underline{a}-\mathbf{A}\underline{b}\right) , \\ \mathbf{D}^{-1} &= \mathbf{A}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{A}+\mathbf{C}^{-1} , \\ r &= \left(\underline{a}-\mathbf{A}\underline{b}\right)^{\mathrm{T}}\left(\mathbf{B}+\mathbf{A}\mathbf{C}\mathbf{A}^{\mathrm{T}}\right)^{-1}\left(\underline{a}-\mathbf{A}\underline{b}\right) . \end{aligned}$$

□

Lemma 2 (Inversion of Block Matrices) *If* $\mathbf{A}$ *and* $\mathbf{D}$ *are regular matrices, then*

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{A} & \mathbf{U} \\ \mathbf{V} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{U}\mathbf{Q}^{-1}\mathbf{V}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{U}\mathbf{Q}^{-1} \\ -\mathbf{Q}^{-1}\mathbf{V}\mathbf{A}^{-1} & \mathbf{Q}^{-1} \end{bmatrix},$$

where $\mathbf{Q} = \mathbf{D} - \mathbf{V}\mathbf{A}^{-1}\mathbf{U}$ *is the Schur complement of the block matrix* $\mathbf{M}$. *Furthermore, the determinant* $|.|$ *of the matrix* $\mathbf{M}$ *is*

$$|\mathbf{M}| = |\mathbf{A}| \cdot |\mathbf{Q}| \,. \tag{24}$$

□

For a proof of Lemma 1, see for example the proof of Lemma A.4, pp. 261–262 in [13]. Lemma 2 is proved in [3].

References

[1] Philipp Boyle and Marcus Frean. Dependent Gaussian Processes. In Lawrence K. Saul, Yair Weiss, and Leon Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 217–224. MIT Press, 2005.

[2] Lehel Csató and Manfred Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, March 2002.

[3] W. J. Duncan. Some devices for the solution of large sets of simultaneous linear equations. *Philosophical Magazine Series 7*, 35(249):660–670, 1944.

[4] Jouni Hartikainen and Simo Särkkä. Kalman Filtering and Smoothing Solutions to Temporal Gaussian Process Regression Models. In *Proceedings of IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 379–384, August 2010.

[5] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*. Dover Publications, Inc., 2007.

[6] Genshiro Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[7] Neil Lawrence, Matthias Seeger, and Ralf Herbich. Fast Sparse Gaussian Process Methods: The Informative Vector Machine. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 609–616. The MIT Press, 2003.

[8] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. Real-Time Local GP Model Learning. In Olivier Sigaud and Jan Peters, editors, *From Motor Learning to Interaction Learning in Robots*, volume 264 of *Studies in Computational Intelligence*, pages 193–207. Springer-Verlag, 2010.

[9] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007.

[10] Ananth Ranganathan, Ming-Hsuan Yang, and Jeffrey Ho. Online Sparse Gaussian Process Regression and Its Applications. *IEEE Transactions on Image Processing*, 20(2):391–404, February 2011.

[11] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[12] H. E. Rauch, F. Tung, and C. T. Striebel. Maximum Likelihood Estimates of Linear Dynamic Systems. *AIAA Journal*, 3(8):1445–1450, August 1965.

[13] David J. Salmond. *Tracking in Uncertain Environments*. PhD thesis, Royal Aerospace Establishment, Farnborough, Hants, UK, 1989.

[14] Alex J. Smola and Peter Bartlett. Sparse Greedy Gaussian Process Regression. In T. K. Leen, T. G. Diettrich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 619–625. The MIT Press, 2001.

[15] Ed Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1259–1266. The MIT Press, 2006.

[16] Volker Strassen. Gaussian Elimination is not Optimal. *Numerische Mathematik*, 13(4):354–356, August 1969.

[17] Volker Tresp. A Bayesian Committee Machine. *Neural Computation*, 12(11):2719–2741, 2000.

[18] Grace Wahba. *Spline Models for Observational Data*, chapter 7 - Finite-Dimensional Approximating Subspaces. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1990.

[19] Christopher K. I. Williams and Matthias Seeger. Using the Nyström Method to Speed Up Kernel Machines. In T. K. Leen, T. G. Diettrich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. The MIT Press, 2001.

Paper K

Recursive Gaussian Process: On-line Regression and Learning

Authors: Marco F. Huber

Edited version of the paper:

M. F. Huber. Recursive Gaussian Process: On-line Regression and Learning. *Pattern Recognition Letters,* vol. 45, pages 85–91, August 2014.

Recursive Gaussian Process: On-line Regression and Learning

Marco F. Huber

AGT International
Darmstadt, Germany
marco.huber@ieee.org

Abstract

Two approaches for on-line Gaussian process regression with low computational and memory demands are proposed. The first approach assumes known hyperparameters and performs regression on a set of basis vectors that stores mean and covariance estimates of the latent function. The second approach additionally learns the hyperparameters on-line. For this purpose, techniques from nonlinear Gaussian state estimation are exploited. The proposed approaches are compared to state-of-the-art sparse Gaussian process algorithms.

1 Introduction

Gaussian processes (GPs) allow non-parametric learning of a regression function from noisy data and can be considered Gaussian distributions over functions conditioned on the data [11]. Unfortunately, due to their non-parametric nature, GPs require computations that scale with $\mathcal{O}(n^3)$ for training, where n is the number of data points.

In order to reduce the computational load, sparse approximations have been proposed in the recent years. In [10] a unifying framework for so-called *active set* approaches has been derived. Here, instead of processing the entire training data set, only a subset of the data points—the active set with $s \ll n$ data points—is used. This framework comprises for instance the subset of regressors [14], sparse on-line GP (SOGP, [2]), or sparse pseudo-input GP (SPGP, [15]). Thanks to the sparse representation, the computational load is reduced to $\mathcal{O}(s^2 \cdot n)$ or even to $\mathcal{O}(s^3)$ (see the approach proposed in [4]).

GP regression can also be sped up by partitioning the training data into separate data sets, where for each data set a separate GP is learned (see e.g. [17, 131]). For calculating the GP prediction, the results of the separate GPs are combined. In contrast to active set methods, partitioning approaches make use of the entire training data.

Most of the above approximations assume that the whole data set is available a priori and thus, training can be performed *off-line* in a batch mode. Only a few sparse approaches have been proposed that allow sequential training of GPs for data that arrives *on-line*, i.e., streaming data. In [2] for instance, a score value is assigned to each element of the active set. If a new data point arrives, it is added to the active set, while an element with the lowest score is eliminated. For specific kernel functions, the approach proposed in [3] transforms GP regression into a Kalman state estimation problem that merely scales with $\mathcal{O}(n)$. Unfortunately, this approach so far is only applicable for one-dimensional inputs.

The approaches proposed in this paper allow for both a sparse representation *and* on-line processing. For this purpose, the regression function is represented by means of a finite set of *basis vectors*. Training with incoming data, i.e., updating mean and covariance estimates featured by the basis vectors (Section 3) as well as simultaneously learning hyperparameters (Section 4), is performed recursively via Bayesian state estimation techniques. After updating the newly arrived data points can be discarded, while the joint Gaussian state estimate of the regression function and the hyperparameters is sufficient for prediction.

2 Problem Formulation

For GP regression, it is assumed that a set of data $\mathcal{D} = \{(\underline{x}_1, y_1), \ldots, (\underline{x}_n, y_n)\}$ is drawn from the noisy process

$$y_i = g(\underline{x}_i) + \varepsilon\ , \tag{1}$$

where $\underline{x}_i \in \mathbb{R}^d$ are the inputs, $y_i \in \mathbb{R}$ are the observations or outputs, and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is zero-mean Gaussian noise with variance σ^2. For brevity reasons, $\mathbf{X}_\mathcal{D} = [\underline{x}_1, \ldots, \underline{x}_n]$ are all inputs and $\underline{y} = [y_1, \ldots, y_n]^\mathrm{T}$ are the corresponding observations in the following.

A GP is used to infer the latent function $g(.)$ from $\mathcal{D}$. The GP is completely defined by a mean function $m(\underline{x}) \triangleq \mathrm{E}\{g(\underline{x})\}$ specifying the expected output value, and a positive semi-definite covariance function $k(\underline{x},\underline{x}') \triangleq \mathrm{cov}\{g(\underline{x}), g(\underline{x}')\}$, which specifies the covariance between pairs of inputs and is often called a *kernel*. Typical examples are the zero mean function $m(\underline{x}) = 0$ and the squared exponential (SE) kernel

$$k(\underline{x}, \underline{x}') = \alpha^2 \cdot \exp\left(-\tfrac{1}{2}\left(\underline{x} - \underline{x}'\right)^{\mathrm{T}} \Lambda^{-1} \left(\underline{x} - \underline{x}'\right)\right) . \tag{2}$$

In (2) $\Lambda = \mathrm{diag}(l_1, l_2, \dots, l_d)$ is a diagonal matrix of the characteristic length-scales l_i for each input dimension and α^2 is the variance of the latent function g. Such parameters of the mean and covariance functions together with the noise standard deviation σ are called the *hyperparameters* of the GP. In the following, all hyperparameters are collected in the vector $\underline{\theta} \in \mathbb{R}^r$, e.g., $\underline{\theta} = [\alpha, l_1, \dots, l_d, \sigma]^{\mathrm{T}}$ comprising the parameters of the SE kernel (2) and the noise standard deviation. It is worth mentioning that the approach proposed in this paper holds for *arbitrary* mean and covariance functions.

For any finite set of inputs a GP provides a multivariate Gaussian distribution of the outputs. For example, the distribution of the function value $g_* = g(\underline{x}_*)$ for an arbitrary test input $\underline{x}_*$ is a univariate Gaussian with mean and variance

$$\begin{aligned} \mu_g(\underline{x}_*) &= \mathrm{E}\{g_*\} = m_* + \underline{k}_*^{\mathrm{T}} \mathbf{K}_x^{-1} \left(\underline{y} - \underline{m}\right) , \\ \sigma_g^2(\underline{x}_*) &= \mathrm{var}\{g_*\} = k_{**} - \underline{k}_*^{\mathrm{T}} \mathbf{K}_x^{-1} \underline{k}_* , \end{aligned} \tag{3}$$

respectively. Here, $\mathrm{var}\{.\}$ is the variance, $\mathbf{K}_x \triangleq \mathbf{K} + \sigma^2 \mathbf{I}$, $m_* \triangleq m(\underline{x}_*)$, $\underline{m} \triangleq m(\mathbf{X}_{\mathcal{D}})$, $\underline{k}_* \triangleq k(\mathbf{X}_{\mathcal{D}}, \underline{x}_*)$, $k_{**} \triangleq k(\underline{x}_*, \underline{x}_*)$, and $\mathbf{K} \triangleq k(\mathbf{X}_{\mathcal{D}}, \mathbf{X}_{\mathcal{D}})$ is the kernel matrix.

For GP prediction, i.e., for calculating the distribution for a given set of test inputs according to (3), it is necessary to calculate the kernel matrix $\mathbf{K}$, to invert the matrix $\mathbf{K}_x$, and to multiply $\mathbf{K}_x$ with $\underline{k}_*$. Both the kernel matrix calculation and the multiplication scale with $\mathcal{O}(n^2)$, while the inversion even scales with $\mathcal{O}(n^3)$. Thus, for large data sets $\mathcal{D}$, storing the kernel matrix and solving all calculations is prohibitive. The following recursive GP approach aims at performing all calculations computationally very efficient on a set of $m \ll n$ so-called *basis vectors*.

3 On-line Regression

At first, let us assume that the hyperparameters are already known and thus, have not to be learned from data. This assumption will be avoided in Section 4.

In the following, our approach proposed in [5] is summarized, which focuses on performing on-line regression given a set of m basis vectors. These basis vectors are located at $\mathbf{X} \triangleq [\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_m]$ and store local estimates $\underline{g} \triangleq g(\mathbf{X})$ of the latent function $g(.)$. Thus, the basis vectors can be considered an active set allowing a sparse GP representation. In contrast to most other active set approaches, the basis vectors are updated *on-line* with new observations $\underline{y}_t$ at inputs $\mathbf{X}_t \triangleq [\underline{x}_{t,1}, \underline{x}_{t,2}, \ldots, \underline{x}_{t,n_t}]$ and time step $t = 0, 1, \ldots$, which makes this approach well suited for streaming data. Also off-line processing is possible by presenting the data in $\mathcal{D}$ in batches to the algorithm.

For all steps $t = 0, 1, \ldots$ it assumed that the basis vectors are fixed in number and location. Since $g(\underline{x})$ is assumed to be a GP, the initial distribution $p_0\left(\underline{g}\right) = \mathcal{N}\left(\underline{g}; \underline{\mu}_0^g, \mathbf{C}_0^g\right)$ of $\underline{g}$ for $t = 0$ is Gaussian with mean $\underline{\mu}_0^g \triangleq m(\mathbf{X})$ and covariance $\mathbf{C}_0^g \triangleq k(\mathbf{X}, \mathbf{X})$.

The goal is now to calculate the posterior distribution $p\left(\underline{g}|\underline{y}_{1:t}\right)$, with $\underline{y}_{1:t}$ comprising all observations up to step t, recursively by updating the prior distribution of $\underline{g}$ from the previous step $t-1$

$$p\left(\underline{g}|\underline{y}_{1:t-1}\right) = \mathcal{N}\left(\underline{g}; \underline{\mu}_{t-1}^g, \mathbf{C}_{t-1}^g\right)$$

with the new observations $\underline{y}_t$.

One might think of exploiting (3) for incorporating the new observations. This however, is not suitable for recursive processing for mainly three reasons. First, (3) merely allow a prediction for given inputs and no incorporation of new information. Second, (3) operates directly on the data $\mathcal{D}$. To allow recursive processing with constant time and memory, not the data $\mathcal{D}$ but a distribution $p\left(\underline{g}|\underline{y}_{1:t-1}\right)$ sparsely representing $\mathcal{D}$ needs to processed. Third, no correlation or cross-covariance between $\mathbf{X}$ and $\mathbf{X}_t$ is provided, which however is of paramount importance for updating $p\left(\underline{g}|\underline{y}_{1:t-1}\right)$. Instead, for deriving a recursive algorithm, the desired posterior distribution is expanded according to

$$p\left(\underline{g}|\underline{y}_{1:t}\right) = \int \underbrace{c_t \cdot p\left(\underline{y}_t|\underline{g},\underline{g}_t\right) \cdot \overbrace{p\left(\underline{g}_t|\underline{g}\right) \cdot p\left(\underline{g}|\underline{y}_{1:t-1}\right)}^{=p\left(\underline{g},\underline{g}_t|\underline{y}_{1:t-1}\right)\text{ (inference)}}}_{=p\left(\underline{g},\underline{g}_t|\underline{y}_{1:t}\right)\text{ (update)}} \mathrm{d}\underline{g}_t \tag{4}$$

in two processing steps: (*inference*) calculating the joint prior $p\left(\underline{g},\underline{g}_t|\underline{y}_{1:t-1}\right)$ given the prior $p\left(\underline{g}|\underline{y}_{1:t-1}\right)$, which provides the required correlation information between $\mathbf{X}$ and $\mathbf{X}_t$, and (*update*) updating the joint prior with the observations $\underline{y}_t$. The second step follows from applying Bayes' law and integrating out $\underline{g}_t \triangleq g(\mathbf{X}_t)$, where c_t is a normalization constant. The integration is required for maintaining a constant number of basis vectors.

3.1 Inference

In order to determine the joint prior $p\left(\underline{g},\underline{g}_t|\underline{y}_{1:t-1}\right)$, the chain rule for probability distribution is applied, which yields

$$\begin{aligned} p\left(\underline{g},\underline{g}_t|\underline{y}_{1:t-1}\right) &= p\left(\underline{g}_t|\underline{g}\right) \cdot p\left(\underline{g}|\underline{y}_{1:t-1}\right) \\ &= \mathcal{N}\left(\underline{g}_t;\underline{\mu}_t^p,\mathbf{B}\right) \cdot \mathcal{N}\left(\underline{g};\underline{\mu}_{t-1}^g,\mathbf{C}_{t-1}^g\right) , \end{aligned} \tag{5}$$

with

$$\underline{\mu}_t^p \triangleq m(\mathbf{X}_t) + \mathbf{J}_t \cdot \left(\underline{\mu}_{t-1}^g - m(\mathbf{X})\right) , \tag{6}$$

$$\mathbf{B} \triangleq k(\mathbf{X}_t,\mathbf{X}_t) - \mathbf{J}_t \cdot k(\mathbf{X},\mathbf{X}_t) , \tag{7}$$

$$\mathbf{J}_t \triangleq k(\mathbf{X}_t,\mathbf{X}) \cdot k(\mathbf{X},\mathbf{X})^{-1} . \tag{8}$$

The first equality in (5) follows from assuming that $\underline{g}_t$ is conditionally independent of the past observations $\underline{y}_{1:t-1}$ given $\underline{g}$. As any finite representation of a GP is Gaussian, this also holds for the joint prior. Hence, the conditional distribution $p\left(\underline{g}_t|\underline{g}\right)$ is Gaussian as well and results from the joint prior by conditioning on $\underline{g}$ (see for example Chapter 2.6 in [7]), which results in the second equality.

After some algebraic transformations, where some basic properties of Gaussian distributions and the Woodbury formula is utilized, the product in (5) yields the joint Gaussian $p\left(\underline{g},\underline{g}_t|\underline{y}_{1:t-1}\right)=\mathcal{N}\left(\underline{q};\mathbf{Q}\right)$ of $\underline{g}$ and $\underline{g}_t$ with mean and covariance

$$\underline{q}\triangleq\begin{bmatrix}\underline{\mu}_{t-1}^g\\ \underline{\mu}_t^p\end{bmatrix}\quad\text{and}\quad\mathbf{Q}\triangleq\begin{bmatrix}\mathbf{C}_{t-1}^g & \mathbf{C}_{t-1}^g\mathbf{J}_t^{\mathrm{T}}\\ \mathbf{J}_t\mathbf{C}_{t-1}^g & \mathbf{C}_t^p\end{bmatrix}, \tag{9}$$

respectively, and with covariance $\mathbf{C}_t^p\triangleq\mathbf{B}+\mathbf{J}_t\mathbf{C}_{t-1}^g\mathbf{J}_t^{\mathrm{T}}$. This inference step coincides with the augmented Kalman Smoother proposed in [13], but there no update step for basis vectors as introduced next is derived.

3.2 Update

The next step is to perform the update and marginalization in (4). For this purpose, the joint prior $p\left(\underline{g},\underline{g}_t|\underline{y}_{1:t-1}\right)=p\left(\underline{g}_t|\underline{y}_{1:t-1}\right)\cdot p\left(\underline{g}|\underline{g}_t,\underline{y}_{1:t-1}\right)$ is now factorized by conditioning on $\underline{g}_t$. Furthermore, the fact that $\underline{g}$ is not observed is utilized and thus, $p\left(\underline{y}_t|\underline{g},\underline{g}_t\right)=p\left(\underline{y}_t|\underline{g}_t\right)$ is independent of $\underline{g}$. Since $p\left(\underline{y}_t|\underline{g}_t\right)=\mathcal{N}\left(\underline{y}_t;\underline{g}_t,\sigma^2\mathbf{I}\right)$ according to (1) and $p\left(\underline{g}_t|\underline{y}_{1:t-1}\right)=\mathcal{N}\left(\underline{g}_t;\underline{\mu}_t^p,\mathbf{C}_t^p\right)$ according to (9) are both Gaussian, $\underline{g}_t$ can be updated easily via a *Kalman filter* update step. Updating $\underline{g}$ and integrating out $\underline{g}_t$ is then performed simultaneously.

Applying the well-known Kalman filter update yields $p\left(\underline{g}_t|\underline{y}_{1:t}\right)=\mathcal{N}\left(\underline{g}_t;\underline{\mu}_t^e,\mathbf{C}_t^e\right)$ with mean and covariance

$$\begin{aligned}\underline{\mu}_t^e&\triangleq\underline{\mu}_t^p+\mathbf{G}_t\cdot\left(\underline{y}_t-\underline{\mu}_t^p\right),\\ \mathbf{C}_t^e&\triangleq\mathbf{C}_t^p-\mathbf{G}_t\mathbf{C}_t^p,\end{aligned}$$

respectively, where $\mathbf{G}_t\triangleq\mathbf{C}_t^p\cdot\left(\mathbf{C}_t^p+\sigma^2\mathbf{I}\right)^{-1}$ is the Kalman gain. The multiplication of the two Gaussians $p\left(\underline{g}_t|\underline{y}_{1:t}\right)$ and $p\left(\underline{g}|\underline{g}_t,\underline{y}_{1:t-1}\right)$ again results in a joint Gaussian distribution of $\underline{g}$ and $\underline{g}_t$ with mean and covariance

$$\underline{\mu}_t=\begin{bmatrix}\underline{\mu}_t^g\\ \underline{\mu}_t^e\end{bmatrix}\quad\text{and}\quad\mathbf{C}_t=\begin{bmatrix}\mathbf{C}_t^g & \mathbf{L}_t\mathbf{C}_t^e\\ \mathbf{C}_t^e\mathbf{L}_t^{\mathrm{T}} & \mathbf{C}_t^e\end{bmatrix}$$

Algorithm 1 Recursive Gaussian Process (RGP)

```
   ▷ Inference
1: Calculate gain matrix $\mathbf{J}_t$ according to (8)
2: Calculate mean $\underline{\mu}_t^p$ via (6) and covariance matrix $\mathbf{C}_t^p$ via (9)
   ▷ Update
3: Calculate gain matrix $\tilde{\mathbf{G}}_t$ according to (12)
4: Calculate mean $\underline{\mu}_t^g$ via (10) and covariance matrix $\mathbf{C}_t^g$ via (11)
```

respectively, where $\mathbf{L}_t \triangleq \mathbf{C}_{t-1}^g \mathbf{J}_t^{\mathrm{T}} \left(\mathbf{C}_t^p\right)^{-1}$ and

$$\underline{\mu}_t^g = \underline{\mu}_{t-1}^g + \tilde{\mathbf{G}}_t \cdot \left(\underline{y}_t - \underline{\mu}_t^p\right) , \tag{10}$$

$$\mathbf{C}_t^g = \mathbf{C}_{t-1}^g - \tilde{\mathbf{G}}_t \mathbf{J}_t \mathbf{C}_{t-1}^g , \tag{11}$$

$$\tilde{\mathbf{G}}_t = \mathbf{L}_t \cdot \mathbf{G}_t = \mathbf{C}_{t-1}^g \mathbf{J}_t^{\mathrm{T}} \cdot \left(\mathbf{C}_t^p + \sigma^2 \mathbf{I}\right)^{-1} . \tag{12}$$

We are merely interested in obtaining the distribution $p\left(\underline{g}\middle|\underline{y}_{1:t}\right) = \mathcal{N}\left(\underline{g}; \underline{\mu}_t^g, \mathbf{C}_t^g\right)$, i.e., updating the latent function at the basis vectors $\mathbf{X}$ in order to keep the memory and computational complexity bounded over time, and thus $\underline{g}_t$ is integrated out. This corresponds to neglecting the mean $\underline{\mu}_t^e$ and covariance $\mathbf{C}_t^e$ of $\underline{g}_t$ as well as the cross-covariance $\mathbf{L}_t \mathbf{C}_t^e$.

Putting all together, at steps $t = 1, 2, \ldots$ the proposed approach named *recursive GP* (RGP) recursively processes observations $\underline{y}_t$ at the inputs $\mathbf{X}_t$ as listed in Algorithm 1. This recursion commences from the initial mean $\underline{\mu}_0^g = m(\mathbf{X})$ and covariance $\mathbf{C}_0^g = k(\mathbf{X}, \mathbf{X})$.

4 On-line Learning

In this section, the assumption of a-priori known hyperparameters is relaxed. Instead, the goal is now to learn the hyperparameters $\underline{\theta}$ simultaneously with estimating the values of the latent function $g(.)$ at the basis vectors. This is achieved by formulating the learning part as a recursive parameter estimation problem, which can be performed together with the function value estimation. Similar to Section 3, this boils down to calculating a joint posterior distribution

$p\left(\underline{z}_t|\underline{y}_{1:t}\right) = \mathcal{N}\left(\underline{z}_t; \underline{\mu}_t^z, \mathbf{C}_t^z\right)$, where $\underline{z}_t^{\mathrm{T}} \triangleq \left[\underline{g}^{\mathrm{T}}, \underline{\theta}_t^{\mathrm{T}}\right]$ is the joint hidden state with mean and covariance

$$\underline{\mu}_t^z \triangleq \begin{bmatrix} \underline{\mu}_t^g \\ \underline{\mu}_t^\theta \end{bmatrix}, \quad \mathbf{C}_t^z \triangleq \begin{bmatrix} \mathbf{C}_t^g & \mathbf{C}_t^{g\theta} \\ \mathbf{C}_t^{\theta g} & \mathbf{C}_t^\theta \end{bmatrix}.$$

Starting point for this calculation is a joint prior distribution $p\left(\underline{z}_{t-1}|\underline{y}_{1:t-1}\right)$ at step $t-1$, which is updated with the new observations $\underline{y}_t$. This requires the following two operations: (*inference*) calculating a joint distribution $p\left(\underline{z}_{t-1}, \underline{g}_t|\underline{y}_{1:t-1}\right)$ by exploiting the results of Section 3.1, and (*update*) incorporation of the new observations $\underline{y}_t$ and marginalization to obtain $p\left(\underline{z}_t|\underline{y}_{1:t}\right)$.

4.1 Inference

To incorporate the new inputs $\mathbf{X}_t$, it is necessary to infer the latent function $g(.)$ at $\mathbf{X}_t$. For this purpose, the intermediate result (9) derived in Section 3.1 is exploited. The part of the mean $\underline{q}$ and the covariance $\mathbf{Q}$ regarding $\underline{g}_t$ can alternatively be calculated by employing a Kalman predictor on the linear state-space model

$$\underline{g}_t = \mathbf{J}_t \cdot \underline{g} + \underline{w}_t \quad , \quad \underline{w}_t \sim \mathcal{N}(\underline{b}, \mathbf{B}) , \tag{13}$$

where $\underline{b} \triangleq m(\mathbf{X}_t) - \mathbf{J}_t \cdot m(\mathbf{X})$ and $\mathbf{B}$ is according to (7). In order to also correlate $\underline{g}_t$ with the hyperparameters, the model in (13) is extended to a state-space model given by

$$\begin{bmatrix} \underline{z}_{t-1} \\ \underline{g}_t \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \\ \mathbf{J}_t\left(\underline{\theta}_{t-1}\right) & \mathbf{0} \end{bmatrix}}_{\triangleq \mathbf{A}_t\left(\underline{\theta}_{t-1}\right)} \cdot \underbrace{\begin{bmatrix} \underline{g} \\ \underline{\theta}_{t-1} \end{bmatrix}}_{\underline{z}_{t-1}} + \underline{w}_t , \tag{14}$$

where the noise $\underline{w}_t \sim \mathcal{N}\left(\underline{\mu}_t^w, \mathbf{C}_t^w\right)$ is Gaussian with mean and covariance

$$\underline{\mu}_t^w \triangleq \begin{bmatrix} \underline{0} \\ \underline{0} \\ \underline{b}\left(\underline{\theta}_{t-1}\right) \end{bmatrix}, \mathbf{C}_t^w \triangleq \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}\left(\underline{\theta}_{t-1}\right) \end{bmatrix}, \tag{15}$$

respectively. Please note, that we now made the dependence on the hyperparameters explicit. The first two rows in (14) and (15) are merely an identity mapping of the given joint state $\underline{z}_{t-1}$, while the last row corresponds to (13).

Based on model (14), performing a prediction would yield the desired joint distribution $p\left(\underline{z}_{t-1}, \underline{g}_t \middle| \underline{y}_{1:t-1}\right)$. Unfortunately, the model is nonlinear with respect to the hyperparameters $\underline{\theta}_{t-1}$ and thus, the prediction cannot be performed exactly in closed form. An approximate prediction has to be employed instead. In order to keep the approximation error bounded, we exploit the fact that the model in (14) is *conditionally linear*, i.e., for a given hyperparameter the model is linear and prediction can be performed exactly via the Kalman predictor.

For conditionally linear models, efficient prediction techniques have been proposed in [1]. Here, for the nonlinear part—the hyperparameters in our case—a collection of so-called *sigma points* $\hat{\underline{\theta}}_i$ is selected and given weights ω_i, $i = 1 \ldots s$. The mostly employed selection scheme for sigma points, which is also applied here, is the *unscented transform* [8]. However, any sigma point algorithm can be employed for the proposed GP learning approach, see for instance [6, 88]. Compared to Monte Carlo sampling, the sigma points have the benefit of being deterministically selected and the sample mean as well as the sample covariance coincide with the mean $\underline{\mu}_{t-1}^{\theta}$ and covariance $\mathbf{C}_{t-1}^{\theta}$.

For each sigma point $\hat{\underline{\theta}}_i$ a Kalman predictor is applied on (14). Combining the individual predictions yields the mean and covariance

$$\underline{\mu}_t^p = \sum_{i=1}^{s} \omega_i \cdot \underline{\mu}_i^p \,, \tag{16}$$

$$\mathbf{C}_t^p = \sum_{i=1}^{s} \omega_i \cdot \left(\left(\underline{\mu}_i^p - \underline{\mu}_t^p\right) \left(\underline{\mu}_i^p - \underline{\mu}_t^p\right)^{\mathrm{T}} + \mathbf{C}_i^p \right) \tag{17}$$

of the joint distribution $p\left(\underline{z}_{t-1}, \underline{g}_t \middle| \underline{y}_{1:t-1}\right) \approx \mathcal{N}\left(\underline{\mu}_t^p, \mathbf{C}_t^p\right)$ with

$$\underline{\mu}_i^p \triangleq \mathbf{A}_t\left(\hat{\underline{\theta}}_i\right) \begin{bmatrix} \underline{\mu}_{t-1}^g + \mathbf{S}_t \cdot \left(\hat{\underline{\theta}}_i - \underline{\mu}_{t-1}^{\theta}\right) \\ \hat{\underline{\theta}}_i \end{bmatrix} + \underline{\mu}_t^w\left(\hat{\underline{\theta}}_i\right) \,, \tag{18}$$

$$\mathbf{C}_i^p \triangleq \mathbf{A}_t\left(\hat{\underline{\theta}}_i\right) \begin{bmatrix} \mathbf{C}_{t-1}^g - \mathbf{S}_t \mathbf{C}_{t-1}^{\theta g} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{A}_t\left(\hat{\underline{\theta}}_i\right)^{\mathrm{T}} + \mathbf{C}_t^w\left(\hat{\underline{\theta}}_i\right) \tag{19}$$

and $\mathbf{S}_t = \mathbf{C}_{t-1}^{g\theta}(\mathbf{C}_{t-1}^{\theta})^{-1}$.

4.2 Update

In order to incorporate the new observations $\underline{y}_t$ in a very computationally efficient manner, the update is performed in two steps. For this purpose, we decompose the joint distribution into an observed and an unobserved part according to

$$p\left(\underline{z}_t|\underline{y}_{1:t}\right) = \int \underbrace{p\left(\underline{g}, \underline{\theta}_t^- | \sigma, \underline{g}_t\right)}_{\text{unobserved}} \cdot \underbrace{p\left(\sigma, \underline{g}_t | \underline{y}_{1:t}\right)}_{\text{observed}} \mathrm{d}\underline{g}_t \,,$$

where $\underline{\theta}_t^-$ indicates the vector of all hyperparameters excluding σ. The observable state $\underline{o}_t^{\mathrm{T}} \triangleq [\sigma, \underline{g}_t^{\mathrm{T}}]$ directly affects the observations $\underline{y}_t$ according to (1) and thus, can be directly updated. In the second step, correlation in the joint distribution $p\left(\underline{z}_{t-1}, \underline{g}_t | \underline{y}_{1:t-1}\right)$ is exploited for updating the unobservable part $\underline{u}_{t-1}^{\mathrm{T}} \triangleq \left[\underline{g}^{\mathrm{T}}, (\underline{\theta}_t^-)^{\mathrm{T}}\right]$, where the decomposition of mean $\underline{\mu}_t^p$ and covariance $\mathbf{C}_t^p$ regarding $\underline{u}_{t-1}$ and $\underline{o}_t$ is according to

$$\underline{\mu}_t^p = \begin{bmatrix} \underline{\mu}_{t-1}^u \\ \underline{\mu}_t^o \end{bmatrix} , \mathbf{C}_t^p = \begin{bmatrix} \mathbf{C}_{t-1}^u & \mathbf{C}_t^{uo} \\ \mathbf{C}_t^{ou} & \mathbf{C}_t^o \end{bmatrix} .$$

This two-step procedure reduces the computational demand significantly as smaller matrices are multiplied and inverted.

Update Observable State The sigma point technique utilized for the inference step is one way for updating the observable state. However, to not introduce approximation errors, the process model (1) is reformulated to

$$y_i = g(x_i) + \sigma \cdot v \quad , \quad v \sim \mathcal{N}(0,1) \,, \tag{20}$$

with v being uncorrelated with σ. The model (20) is equivalent to (1) since $\varepsilon = \sigma \cdot v$ with identical mean and variance. Here, the standard deviation σ of the observation noise is made explicitly accessible. This simplifies the update, as the mean and covariance of the observations as well as the cross-covariance between observable state and observations can be calculated exactly in closed form according to

$$\underline{\mu}_t^y = \mathrm{E}\left\{\underline{y}_t\right\} = \mathrm{E}\left\{\underline{g}_t + \sigma \cdot \underline{v}\right\} = \mathrm{E}\left\{\underline{g}_t\right\} \,, \tag{21}$$

$$\mathbf{C}_t^y = \mathrm{cov}\left\{\underline{y}_t\right\} = \mathrm{cov}\left\{\underline{g}_t\right\} + \mathrm{E}\{\sigma^2\} \cdot \mathbf{I} \,, \tag{22}$$

$$\mathbf{C}_t^{oy} = \operatorname{cov}\left\{\underline{o}_t, \underline{y}_t\right\} = \operatorname{cov}\left\{\underline{o}_t, \underline{g}_t\right\} , \tag{23}$$

with $\mathrm{E}\left\{\sigma^2\right\} = \operatorname{var}\{\sigma\} + \mathrm{E}\{\sigma\}^2$, where $\mathrm{E}\{\sigma\}$, $\mathrm{E}\left\{\underline{g}_t\right\}$ are the elements of $\underline{\mu}_t^o$ and $\operatorname{var}\{\sigma\}$, $\operatorname{cov}\left\{\underline{g}_t\right\}$, $\operatorname{cov}\left\{\underline{o}_t, \underline{g}_t\right\}$ are the elements of $\mathbf{C}_t^o$.

Assuming that the observed state $\underline{o}_t$ and the observations $\underline{y}_t$ are jointly Gaussian distributed—a typical assumption in Gaussian filters like the unscented Kalman filter—updating the observed state can be performed by conditioning on the observations, which yields the desired conditional distribution $p\left(\sigma, \underline{g}_t \middle| \underline{y}_{1:t}\right) \approx \mathcal{N}\left(\underline{\mu}_t^e, \mathbf{C}_t^e\right)$ with mean and covariance according to

$$\underline{\mu}_t^e = \underline{\mu}_t^o + \mathbf{G}_t \cdot \left(\underline{y}_t - \underline{\mu}_t^y\right) , \tag{24}$$

$$\mathbf{C}_t^e = \mathbf{C}_t^o - \mathbf{G}_t \cdot \mathbf{C}_t^y \cdot \mathbf{G}_t^{\mathrm{T}} , \tag{25}$$

with gain matrix $\mathbf{G}_t = \mathbf{C}_t^{o,y} \left(\mathbf{C}_t^y\right)^{-1}$ and $\underline{\mu}_t^y$, $\mathbf{C}_t^y$, $\mathbf{C}_t^{oy}$ according to (21)–(23).

Update Joint State By means of (24) and (25) it is now possible to update the unobserved part. Therefore, the results in [1] are exploited, which yields the distribution $p\left(\underline{g}, \underline{\theta}_t^- \middle| \sigma, \underline{g}_t\right) = \mathcal{N}\left(\underline{\mu}_t^u, \mathbf{C}_t^u\right)$ with mean and covariance

$$\underline{\mu}_t^u = \underline{\mu}_{t-1}^u + \mathbf{L}_t \cdot \left(\underline{\mu}_t^e - \underline{\mu}_t^o\right) , \tag{26}$$

$$\mathbf{C}_t^u = \mathbf{C}_{t-1}^u + \mathbf{L}_t \cdot \left(\mathbf{C}_t^e - \mathbf{C}_t^o\right) \cdot \mathbf{L}_t^{\mathrm{T}} , \tag{27}$$

where $\mathbf{L}_t = \mathbf{C}_t^{uo} \left(\mathbf{C}_t^o\right)^{-1}$. It is worth mentioning that this update has the same structure as the backward pass of the Rauch-Tung-Striebel smoother [12]. To finalize the update step and thus to obtain the joint posterior distribution $p\left(\underline{z}_t \middle| \underline{y}_{1:t}\right) = \mathcal{N}\left(\underline{z}_t; \underline{\mu}_t^z, \mathbf{C}_t^z\right)$ with updated basis vectors and hyperparameters, the results in (24)–(27) are combined according to

$$\underline{\mu}_t^z = \begin{bmatrix} \underline{\mu}_t^u \\ \underline{h}^{\mathrm{T}} \cdot \underline{\mu}_t^e \end{bmatrix} , \mathbf{C}_t^z = \begin{bmatrix} \mathbf{C}_t^u & \mathbf{L}_t \mathbf{C}_t^e \underline{h} \\ \underline{h}^{\mathrm{T}} \mathbf{C}_t^e \mathbf{L}_t^{\mathrm{T}} & \underline{h}^{\mathrm{T}} \mathbf{C}_t^e \underline{h} \end{bmatrix} \tag{28}$$

with $\underline{h}^{\mathrm{T}} \triangleq [1,0,0,\ldots,0]$. The first row in (28) corresponds to marginalizing out $\underline{g}_t$.

Algorithm 2 Recursive Gaussian Process with hyperparameter learning (RGP*)

```
   ▷ Inference
1: Draw sigma points $(\omega_i, \underline{\hat{\theta}}_i)$, $i = 1, \ldots, s$ from $\mathcal{N}\left(\underline{\mu}^\theta_{t-1}, \mathbf{C}^\theta_{t-1}\right)$
2: For each sigma point determine $\underline{\mu}^p_i$, $\mathbf{C}^p_i$ by means of (18) and (19)
3: Calculate $\underline{\mu}^p_t$ and $\mathbf{C}^p_t$ according to (16) and (17)
   ▷ Update
4: Calculate $\underline{\mu}^y_t$, $\mathbf{C}^y_t$, and $\mathbf{C}^{oy}_t$ according to (21)–(23)
5: Update mean $\underline{\mu}^e_t$ and covariance $\mathbf{C}^e_t$ of observed state
   by means of (24) and (25), respectively
6: Update mean $\underline{\mu}^u_t$ and covariance $\mathbf{C}^u_t$ of unobserved state
   by means of (26) and (27), respectively
7: Calculate mean $\underline{\mu}^z_t$ and covariance $\mathbf{C}^z_t$ of joint state $\underline{z}_t$ by means of (28)
```

The simultaneous regression and hyperparameter learning approach proposed above is named recursive GP with learning (RGP*) and is summarized in Algorithm 2.

5 Discussion

To predict the latent function $g(.)$ for non basis vectors, e.g., for plotting the estimate, the steps 1–4 have to be performed. Similarly, additional basis vectors can be introduced, e.g., in order to refine the resolution or improve the accuracy. For this purpose, the inference step has to be applied, where $\mathbf{X}_t$ now contains the location of the additional basis vectors.

Directly modeling some of the hyperparameters by means of a Gaussian distribution may not be appropriate in some cases. For instance, the length-scale hyperparameters of the SE kernel in (2) have to be positive. To account for such constraints, a standard trick in GP regression is to transform the hyperparameters first and then to train the transformed parameters. After training, the inverse transformation is applied in order to obtain the original hyperparameters. In case of positive hyperparameters, the logarithm for transforming and the exponential function as inverse transformation are common. RGP* can directly be used to also train/estimate transformed hyperparameters.

Assuming Gaussian noise ε in (1) is not reasonable for every application. Capturing a non-Gaussian distribution by the proposed methods can for instance be achieved via warping as proposed in [16]. Alternatively, $p\left(\underline{g}|\underline{y}_{1:t-1}\right)$ could be represented by means of a mixture of Gaussians.

RGP is exact in all computations, i.e., no approximations are employed despite the fact that only a fixed set of basis vectors is used. RGP* instead requires two approximations: the sigma point approximation of $\underline{\theta}_t$ in the inference step and the jointly Gaussian assumption of observations and observed state in the update step. Furthermore, if σ is not correlated with any other element of the joint state $\underline{z}_t$, it will not be correlated with $\underline{g}_t$ in the inference step. This correlation however is important to learn σ as well. This drawback can be compensated by ensuring that the initial covariance $\mathbf{C}_0^\theta$ has non-zero off-diagonal elements. In the simulation (see Section 6), $\mathbf{C}_0^\theta$ is almost a diagonal matrix, except of the values that describe the correlation between σ and all other hyperparameters. Here, full (positive) correlation is assumed, i.e., the values of the corresponding line and column in $\mathbf{C}_0^\theta$ are chosen in such a way that the sum of the off-diagonal values is equal to the diagonal value–this satisfies the row sum and the column sum criterion. The initial cross-covariance $\mathbf{C}_0^{g,\theta}$ is assumed to be a zero matrix.

The computation and memory costs of RGP* for a single time step t scale with $\mathcal{O}\left(s \cdot n_t \cdot (m+r)^2 + n_t^3\right)$ and with $\mathcal{O}\left((m+r)^2\right)$, respectively, where s is the number of sigma points, n_t is the number of observations at step t, m is the number of basis vectors, and r is the dimension of $\underline{\theta}$. In case of the employed unscented transform to generate the sigma points, $s = 2r+1$ and thus scales linearly with the number of hyperparameters. The costs of RGP are even less and scale with $\mathcal{O}\left(n_t \cdot m^2\right)$ for computations and $\mathcal{O}\left(m^2\right)$ for memory. If at each step t the same number of observations is processed, than the computational and memory costs are constant for each step for both RGP and RGP*. Furthermore and in contrast to a full GP the computational and memory costs do not increase over time, i.e., when more and more observations become available.

6 Results

The proposed approaches are compared to existing GP algorithms: a standard GP, SOGP, and SPGP. All methods are implemented in MATLAB by means of the GPML toolbox (`http://www.gaussianprocess.org/gpml/code`).

For all experiments, two different covariance functions are considered: the SE kernel (2) as well as the sum of SE kernel and the so-called neural network (NN) kernel

$$k\left(\underline{x}, \underline{x}'\right) = \beta^2 \sin^{-1}\left(\frac{\underline{x}^{\mathrm{T}}\Lambda^{-2}\underline{x}'}{\sqrt{f(\underline{x})f(\underline{x}')}}\right) \tag{29}$$

with $f\left(\underline{x}\right) = 1 + \underline{x}^{\mathrm{T}}\Lambda^{-2}\underline{x}$, signal variance β^2, and the diagonal matrix Λ of characteristic length-scales. While the SE kernel is stationary, i.e., it is a function of $\Delta\underline{x} \triangleq \underline{x} - \underline{x}'$, the NN kernel is non-stationary. Thus, also the sum of SE and NN kernel—denoted as SE+NN in the following—allows modeling non-stationary processes.

To train the GP methods, at every step t a set of observations $\underline{y}_t$ at inputs $\mathbf{X}_t$ is randomly selected from a data set. At the end of training, the regression performance of the different approaches is evaluated and compared using a randomly selected test data set. As performance criteria we employ the root mean square error (rmse), the negative log-likelihood (nll), and the total computation time. For each experiment, 50 Monte Carlo (MC) runs are performed.

Full GP and RGP require off-line learning of the hyperparameters. Therefore, 100 input-observation pairs are selected randomly and evidence maximization is performed [11]. Then, the actual training takes place. SOGP and RGP* instead, learn the hyperparameters on-line by exploiting the data provided at step t, while SPGP learns the hyperparameters off-line *after* collecting all data. The number of active set elements is identical for all approaches.

6.1 Synthetic Data

At first data generated by means of two different synthetic functions are considered. The first function

$$y = \tfrac{x}{2} + \tfrac{25\cdot x}{1+x^2}\cdot\cos(x) + \varepsilon \quad , \quad \varepsilon \sim \mathcal{N}(0, 0.1) \tag{30}$$

is smooth but non-stationary. It is similar to the growth model proposed in [9]. At each step t, 40 input-observation pairs are selected randomly from the interval $[-10, 10]$. In total 100 steps are performed. The sparse representation comprises 50 elements, which are placed equidistant on the interval $[-10, 10]$. As second function we consider

$$y = \mathcal{N}(0.6, 0.04) + \mathcal{N}(0.15, 0.0015) + 4\cdot H(0.3) + \varepsilon \ , \tag{31}$$

where $H(.)$ is the Heaviside step function with $H(a) = 0$ if $x \leq a$ and $H(a) = 1$ if $x > a$. This function has a discontinuity at $x = 0.3$ and was considered as a benchmark in [18]. The noise ε has variance $\sigma^2 = 0.16$. A total of 70 steps are performed, with 50 data points per step drawn from $[-2, 2]$. On this interval, 30 active set elements are placed equidistant.

In Figure 1a on the next page, an exemplary regression result of RGP* with SE+NN kernel is depicted. The true function is accurately reconstructed. As shown in Figure 1b, the hyperparameters are adjusted over time and converge. This leads to improved regression results compared to the other hyperparameter learning approaches SOGP and SPGP as can be seen in Table 1. This holds for both covariance functions, whereas SE+NN yields better results as it is possible to capture the non-stationarity thanks to the non-stationary NN kernel (29). Compared to a full GP, RGP* is slightly inferior. The off-line hyperparameter optimization provides optimal results and RGP* cannot improve further. With the optimal hyperparameters however, RGP performs close to a full GP but with significantly lower runtime.

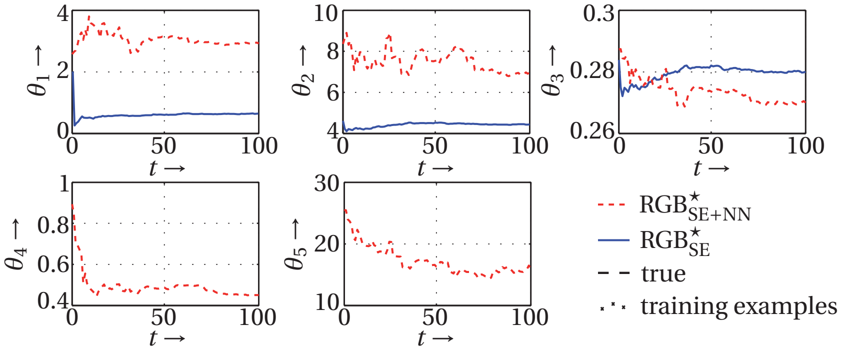
The results in Table 2 indicate that off-line hyperparameter optimization is not always optimal. Here, the hyperparameters learned by RGP* result in better estimates compared to all other algorithms with at the same time lower com-

Table 1: Average rmse, nll, and runtime for function (30).

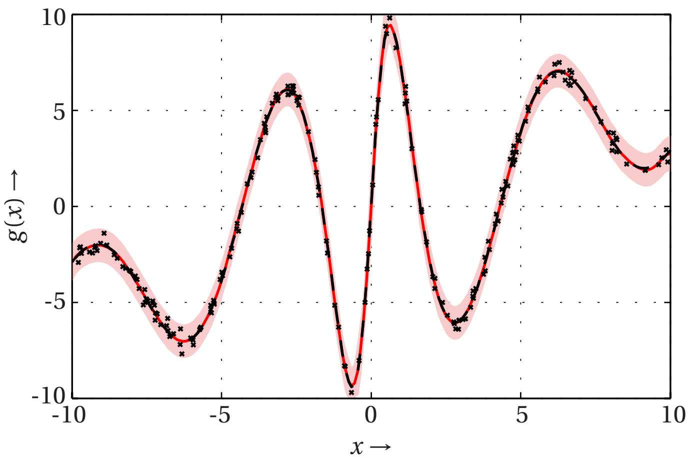
	SE			SE+NN		
	rmse	nll	time in s	rmse	nll	time in s
Full GP	0.31 ± 0.02	0.25 ± 0.05	0.82	0.30 ± 0.02	0.24 ± 0.06	1.46
RGP	0.31 ± 0.02	0.26 ± 0.06	0.16	0.31 ± 0.03	0.24 ± 0.06	0.11
RGP*	0.37 ± 0.02	0.41 ± 0.14	0.65	0.35 ± 0.05	0.34 ± 0.12	1.81
SOGP	1.18 ± 0.03	7.49 ± 0.4	0.93	0.44 ± 0.03	0.80 ± 0.13	2.58
SPGP	0.54 ± 0.02	1.15 ± 0.11	13.05	0.39 ± 0.02	0.51 ± 0.09	24.40

Table 2: Average rmse, nll, and runtime for function (31).

	SE			SE+NN		
	rmse	nll	time in s	rmse	nll	time in s
Full GP	1.38 ± 0.41	1.70 ± 0.33	0.92	1.04 ± 0.43	1.41 ± 0.35	2.57
RGP	1.40 ± 0.38	1.98 ± 0.40	0.45	1.12 ± 0.35	1.86 ± 0.19	0.48
RGP*	0.98 ± 0.11	1.48 ± 0.23	0.38	0.88 ± 0.10	1.39 ± 0.15	1.14
SOGP	1.68 ± 0.07	2.89 ± 0.17	0.8	1.68 ± 0.07	2.88 ± 0.17	2.21
SPGP	1.63 ± 0.10	1.91 ± 0.06	5.6	1.65 ± 0.07	1.93 ± 0.05	10.85



(a) Evolution of the hyperparameters of RGP* with SE (blue, solid) and SE+NN covariance function (red, dashed), where $\theta_1 = l_1$ (length-scale), $\theta_2 = \alpha_1$ (signal standard deviation), $\theta_3 = \sigma$ (noise standard deviation) are the parameters of the SE kernel and $\theta_4 = l_2$ (length-scale), $\theta_5 = \beta$ (signal standard deviation) are the parameters of the NN kernel.



(b) True function (black, dashed line), regression result by RGP* with SE+NN covariance function together with 99% confidence area, and the training examples of step $t = 100$ (black crosses).

Figure 1: Exemplary regression result of proposed approach for function (30).

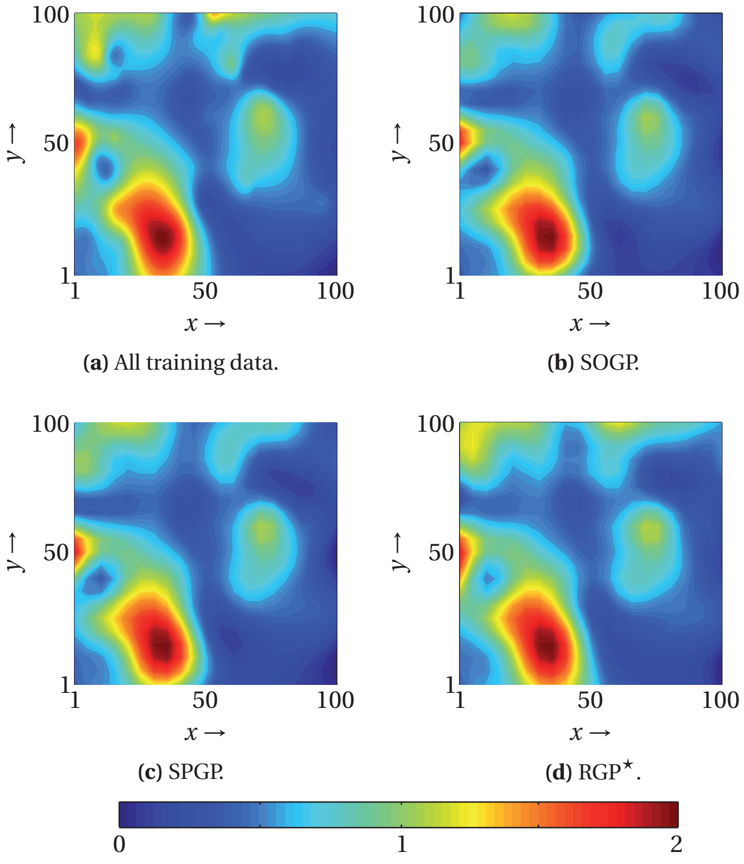


Figure 2: Particulate matter estimates of different approaches.

putational load. It is worth mentioning that RGP* is the only sparse approach that really exploits the properties of the SE+NN kernel resulting in an improved regression compared to the SE kernel.

6.2 Particulate Matter Data

Here, satellite observations of the Global Monitoring for Environment and Security program (`http://www.gmes.info/`, data via `ftp://data-portal.ecmwf.`

int/) are considered. The inputs are 2D measurement locations and the observations are particulate matter (PM_{10}) measurements at these locations. The data set recorded at October 10, 2011 comprises 10,000 elements. For each MC run, 100 active set elements are distributed randomly over the input domain.

Merely the SE covariance function is employed as the SE+NN covariance function does not lead to any improvement. The average performance is listed in Table 3. While all GP approaches except of SOGP have a similar rmse, there is a strong difference in terms of nll. Here, RGP* performs best as it provides lower prediction covariances and thus, is more certain about its estimates. The deviation of SOGP can be explained by Figure 2 on the next page. Occasionally, SOGP fails in accurately modeling the high PM_{10} concentrations in the upper part and it overfits around region (10,40). SPGP behaves similarly, but with much lower deviation from the ground truth. RGP* instead models the concentrations very accurately and even outperforms the full GP, as better hyperparameters are learned from the data.

Table 3: Average rmse, nll, and runtime for PM_{10} data.

	rmse	nll	time in s
Full GP	0.59 ± 0.01	1337.37 ± 217.92	0.18
RGP	0.59 ± 0.02	82.20 ± 24.48	0.06
RGP*	0.56 ± 0.01	45.47 ± 9.41	0.89
SOGP	0.86 ± 0.01	598.34 ± 20.22	1.58
SPGP	0.57 ± 0.01	114.83 ± 22.44	13.87

7 Conclusion and Future Work

Two novel approaches for on-line GP regression have been proposed. Both approaches are especially well suited for streaming data. Given the hyperparameters, the first approach provides similar performance than a full GP, but with significantly lower computation and memory costs. If the hyperparameters are unknown, our second approach is able to learn the hyperparameters on-line from data. Here, the regression performance is comparable to or even better than state-of-the-art approaches, but with lower computational demand. Future work is devoted to adjust the location and number of the basis vectors over time.

References

[1] Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck. Gaussian Filtering using State Decomposition Methods. In *Proceedings of the 12th International Conference on Information Fusion (Fusion)*, pages 579–586, Seattle, Washington, July 2009.

[2] Lehel Csató and Manfred Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, March 2002.

[3] Jouni Hartikainen and Simo Särkkä. Kalman Filtering and Smoothing Solutions to Temporal Gaussian Process Regression Models. In *Proceedings of the IEEE Intl. Workshop on Machine Learning for Signal Processing*, pages 379–384, August 2010.

[4] James Hensman, Nicolo Fusi, and Neil D. Lawrence. Gaussian Processes for Big Data. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*, Bellevue, Washington, USA, 2013.

[5] Marco F. Huber. Recursive Gaussian Process Regression. In *Proceedings of the 38th International Conference on Acoustics, Sound, and Signal Processing (ICASSP)*, pages 3362–3366, Vancouver, BC, Canada, May 2013.

[6] Kazufumi Ito and Kaiqi Xiong. Gaussian Filters for Nonlinear Filtering Problems. *IEEE Transactions on Automatic Control*, 45(5):910–927, May 2000.

[7] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*. Dover Publications, Inc., 2007.

[8] Simon J. Julier and Jeffrey K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[9] Genshiro Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics* , 5(1), 1996.

[10] Joaquin Quiñonero-Candela and Carl E. Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

[11] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[12] H. E. Rauch, F. Tung, and C. T. Striebel. Maximum Likelihood Estimates of Linear Dynamic Systems. *AIAA Journal*, 3(8):1445–1450, August 1965.

[13] Steven Reece and Stephen Roberts. An Introduction to Gaussian Processes for the Kalman Filter Expert. In *Proceedings of the 13th International Conference on Information Fusion*, Edinburgh, UK, 2010.

[14] Alex J. Smola and Peter Bartlett. Sparse Greedy Gaussian Process Regression. In T. K. Leen, T. G. Diettrich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 619–625. The MIT Press, 2001.

[15] Ed Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1259–1266. The MIT Press, 2006.

[16] Edward Snelson, Carl Edward Rasmussen, and Zoubin Ghahramani. Warped Gaussian Processes. In *Advances in Neural Information Processing Systems 16*. The MIT Press, 2004.

[17] Volker Tresp. A Bayesian Committee Machine. *Neural Computation*, 12(11):2719–2741, 2000.

[18] Sally A. Wood, Wenxin Jian, and Martin Tanner. Bayesian mixture of splines for spatially adaptive nonparameteric regression. *Biometrika*, 89(3):513–528, 2002.

Paper L

Optimal Stochastic Linearization for Range-Based Localization

Authors: Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck

Edited version of the paper:

F. Beutler, M. F. Huber, and U. D. Hanebeck. Optimal Stochastic Linearization for Range-Based Localization. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5731–5736, Taipei, Taiwan, October 2010.

Optimal Stochastic Linearization for Range-Based Localization

Frederik Beutler*, Marco F. Huber**, and Uwe D. Hanebeck*

* Intelligent Sensor-Actuator-Systems Laboratory (ISAS)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{beutler|uwe.hanebeck}@ieee.org

** Variable Image Acquisition and Processing Research Group
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB
Karlsruhe, Germany
marco.huber@ieee.org

Abstract

In range-based localization, the trajectory of a mobile object is estimated based on noisy range measurements between the object and known landmarks. In order to deal with this uncertain information, a Bayesian state estimator is presented, which exploits optimal stochastic linearization. Compared to standard state estimators like the Extended or Unscented Kalman Filter, where a point-based Gaussian approximation is used, the proposed approach considers the entire Gaussian density for linearization. By employing the common assumption that the state and measurements are jointly Gaussian, the linearization can be calculated in closed form and thus analytic expressions for the range-based localization problem can be derived.

1 Introduction

In applications such as mobile robot navigation or telepresence, the position of a moving object is often localized based on range measurements between the object and known landmarks. These ranges can for example be measured by times of arrival or field strengths [10].

Existing range-based localization algorithms can be divided into two classes. Approaches of the first class assume exact (or almost exact) range measurements.

As long as this assumption is satisfied, closed-form localization approaches as those in [1, 2, 4, 6, 11], gradient descent algorithms, or methods based on linearization via Taylor-series expansion perform very well. However, these approaches merely allow for a static localization, i.e., a separate localization is performed at every time step. Furthermore, accurate range measurements require specialized and expensive hardware.

Dealing with inaccurate measurements that may arise for example from signal strength information or ultrasonic range finders requires range-based localization approaches from the second class. Based on probabilistic models that capture measurement uncertainties—for instance arising from measurement noise or modeling errors—the object's position and velocity can be estimated by means of a Bayesian estimator in a recursive fashion. This allows for dynamic localization, i.e., the combination of dead reckoning and static localization, for a smoother and more robust localization.

Generally, a closed-form evaluation of the equations of the Bayesian estimator is not possible due to nonlinearities in the measurement model and the object's dynamics model. Thus, approximate estimators like the extended Kalman filter (EKF) [9] or the unscented Kalman filter (UKF) [5] are typically employed, particularly for range-based localization.

In this paper, we proposed an analytically solvable estimator. For this purpose, the standard measurement model consisting of the Euclidean norm between object position and landmark position affected by additive noise is slightly modified by moving the noise into the Euclidean norm and by considering squared ranges. Based on this modification, we derive an analytic expression of the first two moments, i.e., mean and covariance, characterizing the object's position and velocity estimates. This analytic moment calculation (AMC) can be considered as Gaussian estimation employing stochastic linearization. In contrast to the point-based Gaussian estimators such as the EKF or the UKF, the proposed AMC algorithm considers the entire Gaussian density for linearization, leading to more accurate localization results.

The structure of the paper is as follows. The problem formulation in Section 2 provides the modified measurement model and the object's dynamics model. In Section 3, the general form of the proposed state estimator is described. Based on the modified measurement model, the moments can be calculated in closed form, which is shown in Section 4. In Section 5, the proposed algorithm is compared with the EKF, the UKF, and a closed-form solution via simulations and experiments. Conclusions and an outlook to future work are given in Section 6.

2 Problem Formulation

In this paper, dynamic localization of a mobile object is considered. The dynamic state of the object is described by means of the state vector $\underline{\boldsymbol{x}}_k^{\mathrm{T}} = \left[\underline{\boldsymbol{x}}_{k,P}^{\mathrm{T}}, \underline{\boldsymbol{x}}_{k,V}^{\mathrm{T}}\right]$ comprising the object's position $\underline{\boldsymbol{x}}_{k,P} \in \mathbb{R}^3$ and velocity $\underline{\boldsymbol{x}}_{k,V} \in \mathbb{R}^3$ in three-dimensional space. Here, $k = 0, 1, \ldots$ is the discrete time index.

2.1 Dynamics Model

The dynamic behavior—the motion—of the object is described by means of the linear discrete-time dynamic system

$$\underline{\boldsymbol{x}}_{k+1} = \mathbf{A} \cdot \underline{\boldsymbol{x}}_k + \underline{\boldsymbol{w}}_k \, , \tag{1}$$

where the noise $\underline{\boldsymbol{w}}_k$ is assumed to be zero-mean white Gaussian. For a *position velocity model* [12], the matrix $\mathbf{A}$ and the covariance of the process noise $\mathbf{C}^w$ are given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & T \cdot \mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} , \quad \mathbf{C}^w = \begin{bmatrix} \frac{T^3}{3}\mathbf{C}_c^w & \frac{T^2}{2}\mathbf{C}_c^w \\ \frac{T^2}{2}\mathbf{C}_c^w & T \cdot \mathbf{C}_c^w \end{bmatrix} ,$$

respectively, where T is the sampling time and $\mathbf{I}$ is the identity matrix of adequate dimension. $\mathbf{C}_c^w$ is the process noise of the covariance from the continuous time system model, where $\mathbf{C}_c^w = \mathrm{diag}\left(\begin{bmatrix} C_{c,x}^w & C_{c,y}^w & C_{c,z}^w \end{bmatrix}\right)$ with $C_{c,\xi}^w$ being the variance of dimension $\xi \in \{x, y, z\}$.

2.2 Measurement Models

For improving the object's state estimate, range measurements to N landmarks at the known positions $\underline{S}_i \in \mathbb{R}^3$ with $i = 1, \ldots, N$ are incorporated. The nonlinear relation between the object position and the landmark position is given by

$$\boldsymbol{r}_{k,i} = \left\| \underline{S}_i - \underline{\boldsymbol{x}}_{k,P} \right\|_2 \, , \tag{2}$$

where $\boldsymbol{r}_{k,i}$ is the Euclidean distance between object and landmark. $\|.\|_2$ is the Euclidean norm.

In a real scenario, the ranges cannot be measured exactly, i.e., measurement uncertainty has to be considered, which is usually done by incorporating a noise process into (2). Two possibilities arise for incorporation. In the first case given by

$$\boldsymbol{r}_{k,i} = \left\| \underline{S}_i - \underline{\boldsymbol{x}}_{k,P} \right\|_2 + \underline{\boldsymbol{v}}_{k,i} \ , \tag{3}$$

the noise process $\boldsymbol{v}_{k,i}$ directly affects the range $\boldsymbol{r}_{k,i}$, which is the standard model. In the second case

$$\boldsymbol{r}_{k,i} = \left\| \underline{S}_i - \underline{\boldsymbol{x}}_{k,P} - \underline{\boldsymbol{v}}_{k,i} \right\|_2 \ , \tag{4}$$

which is called *noise before non-linearity* [3], the noise process affects the difference between object and landmark position. This measurement model can be interpreted in such a way that the positions of the landmarks are uncertain. In both measurement models, the noise process is assumed to be zero-mean white Gaussian.

In this paper, we will focus on the second model (4) mainly for two reasons. First, the standard model (3) is only appropriate in situations where the distance $\boldsymbol{r}_{k,i}$ is large compared to the variance of the noise $\boldsymbol{v}_{k,i}$. Otherwise, negative ranges are possible, which is not true in reality. This problem cannot occur in the second measurement model. Second, the model in (4) allows analytic moment calculation as will be shown in the following.

3 Recursive State Estimation

Both the measurement model (4) and the system model (1) are utilized in a Bayesian estimation framework for recursively estimating the state $\underline{\boldsymbol{x}}_k$. For this purpose, two alternating steps, i.e., prediction and filtering, are performed.

3.1 Prediction Step

In the prediction step, we are interested in calculating the predicted mean $\underline{\mu}_{k+1}^p$ and covariance $\mathbf{C}_{k+1}^p$ of the state. Thanks to the linear system model (1), the prediction can be performed in closed form by means of the prediction step of the Kalman filter. Assuming that the result of the previous filter step is Gaussian and thus given by the density function $f^e(\underline{x}_k) \triangleq \mathcal{N}(\underline{x}_k; \underline{\mu}_k^e, \mathbf{C}_k^e)$, the mean and the covariance of the predicted density $f^p(\underline{x}_{k+1}) \triangleq \mathcal{N}(\underline{x}_{k+1}; \underline{\mu}_{k+1}^p, \mathbf{C}_{k+1}^p)$ are given by

$$\begin{aligned} \underline{\mu}_{k+1}^p &= \mathbf{A} \cdot \underline{\mu}_k^e \ , \\ \mathbf{C}_{k+1}^p &= \mathbf{A} \cdot \mathbf{C}_k^e \cdot \mathbf{A}^{\mathrm{T}} + \mathbf{C}^w \ , \end{aligned}$$

respectively. It is worth mentioning that the approach proposed in this paper is not restricted to linear system models. The techniques derived in the following for the filter step can also be used for special types of nonlinear dynamics. If the system model consists of a linear combination of trigonometric functions and/or polynomials, as it is the case for example in differential drive or bicycle kinematics, an analytic calculation of the predicted mean and covariance is still possible.

3.2 Filter Step

In the filter step, the current range measurement $\underline{\hat{r}}_k = [\hat{r}_{k,1}, \ldots, \hat{r}_{k,N}]^{\mathrm{T}}$ is used for updating the result of the prediction step $f^p(\underline{x}_k)$ according to Bayes' rule

$$f^e(\underline{x}_k) = c_k \cdot f(\underline{\hat{r}}_k | \underline{x}_k) \cdot f^p(\underline{x}_k) \ ,$$

where $c_k = 1/\int f(\underline{\hat{r}}_k|\underline{x}_k) \cdot f^p(\underline{x}_k)\,\mathrm{d}\underline{x}_k$ is a normalization constant and $f(\underline{\hat{r}}_k|\underline{x}_k)$ is the likelihood defined by (4). Here, $\underline{\hat{r}}_k$ is a realization of the random vector $\underline{\boldsymbol{r}}_k$.

Generally, for nonlinear measurement models, a closed-form calculation of the density $f^e(\underline{x}_k)$ as well as the mean $\underline{\mu}_k^e$ and covariance $\mathbf{C}_k^e$ is not possible. Instead, appropriate approximations have to be applied. For the special measurement model in (4) however, mean $\underline{\mu}_k^e$ and covariance $\mathbf{C}_k^e$ can be calculated analytically if we assume that state and measurement are jointly Gaussian[1]. This assumption is only true if there is a linear relationship between $\underline{\boldsymbol{x}}_k$ and $\underline{\boldsymbol{r}}_k$. Otherwise, it is an approximation and corresponds to a special type of linearization. While typically applied linearization techniques like first-order Taylor-series expansion or unscented transformation [5], which are merely point-based, our approach

1 This assumption is common in Gaussian filters like the EKF or the UKF.

considers the entire predicted density. In doing so, it is possible to calculate the mean $\underline{\mu}_k^e$ and the covariance $\mathbf{C}_k^e$ according to

$$\begin{aligned} \underline{\mu}_k^e &= \underline{\mu}_k^p + \mathbf{C}_k^{x,r} \cdot \left(\mathbf{C}_k^r\right)^{-1} \cdot \left(\underline{\hat{r}}_k - \underline{\mu}_k^r\right) , \\ \mathbf{C}_k^e &= \mathbf{C}_k^p - \mathbf{C}_k^{x,r} \cdot \left(\mathbf{C}_k^r\right)^{-1} \cdot \left(\mathbf{C}_k^{x,r}\right)^{\mathrm{T}} , \end{aligned} \tag{5}$$

respectively, where $\underline{\mu}_k^r$ is the predicted measurement value, $\mathbf{C}_k^r$ is the covariance of the predicted measurement and $\mathbf{C}_k^{x,r}$ is the cross-covariance between state and range measurement. In the following, an analytic calculation of these quantities is provided in order to allow for the closed-form evaluation of (5). For the rest of the paper, the time index k and the superscripts at the symbol f of density functions are omitted for brevity.

4 Analytic Moment Calculation (AMC)

4.1 Modified Measurement Equation

In order to obtain an analytic expression for the required quantities in (5), the range-based measurement equation (4) is squared, which results in

$$\boldsymbol{d}_i \triangleq (\boldsymbol{r}_i)^2 = \left(\underline{S}_i - \underline{\boldsymbol{x}}_P\right)^{\mathrm{T}} \cdot \left(\underline{S}_i - \underline{\boldsymbol{x}}_P\right) - 2 \cdot \left(\underline{S}_i - \underline{\boldsymbol{x}}_P\right)^{\mathrm{T}} \cdot \underline{\boldsymbol{v}}_i + \underline{\boldsymbol{v}}_i^{\mathrm{T}} \cdot \underline{\boldsymbol{v}}_i , \tag{6}$$

where $\boldsymbol{d}_i$ is a squared range assumed to be measured by $\hat{d}_i = \hat{r}_i^2$. Thus, the modified measurement equation (6) can be described in short term via

$$\boldsymbol{d}_i = h_i\left(\underline{\boldsymbol{x}}_P, \underline{\boldsymbol{v}}_i\right) \tag{7}$$

for a single measurement to landmark i and via

$$\underline{\boldsymbol{d}} = \underline{h}\left(\underline{\boldsymbol{x}}_P, \underline{\boldsymbol{v}}\right) \tag{8}$$

for measurements to all landmarks. It is important to note that the moments of the squared ranges (indicated with superscript d) instead of the normal range measurements (indicated with superscript r) have to be used in (5), which yields

$$\begin{aligned} \underline{\mu}^e &= \underline{\mu}^p + \mathbf{C}^{x,d} \cdot \left(\mathbf{C}^d\right)^{-1} \cdot \left(\underline{\hat{d}} - \underline{\mu}^d\right) , \\ \mathbf{C}^e &= \mathbf{C}^p - \mathbf{C}^{x,d} \cdot \left(\mathbf{C}^d\right)^{-1} \cdot \left(\mathbf{C}^{x,d}\right)^{\mathrm{T}} . \end{aligned} \tag{9}$$

4.2 Moment Calculation

Based on (8), the moments $\underline{\mu}^d$, $\mathbf{C}^d$, and $\mathbf{C}^{x,d}$ of the squared range measurement $\underline{\boldsymbol{d}}$ can be calculated in closed form, if the random vectors $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{v}}$ are assumed as stemming from Gaussians $\mathcal{N}\big(\underline{x};\underline{\mu}^p,\mathbf{C}^p\big)$ and $\mathcal{N}\big(\underline{v};\underline{0},\mathbf{C}^v\big)$. The mean and covariance of the state are given by

$$\underline{\mu}^p = \begin{bmatrix} \underline{\mu}^{x_P} \\ \underline{\mu}^{x_V} \end{bmatrix} , \quad \mathbf{C}^p = \begin{bmatrix} \mathbf{C}^{x_P} & \mathbf{C}^{x_P,x_V} \\ \mathbf{C}^{x_V,x_P} & \mathbf{C}^{x_V} \end{bmatrix} ,$$

respectively, where the state consists of the position and the velocity. The measurement noise $\underline{\boldsymbol{v}}$ is zero-mean with a covariance of

$$\mathbf{C}^v = \begin{bmatrix} \mathbf{C}^v_1 & \cdots & \mathbf{C}^v_{1,j} & \cdots & \mathbf{C}^v_{1,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{C}^v_{i,1} & \cdots & \mathbf{C}^v_{i,j} & \cdots & \mathbf{C}^v_{i,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{C}^v_{N,1} & \cdots & \mathbf{C}^v_{N,j} & \cdots & \mathbf{C}^v_{N} \end{bmatrix} ,$$

where $\mathbf{C}^v_{i,j}$ is the three-by-three-dimensional covariance matrix between the ith and jth landmark. It is worth mentioning that the landmarks are assumed to be spatially correlated, which allows general applicability of the proposed algorithm. Assuming uncorrelatedness would simplify the following derivations.

Mean

For calculating the mean, it can be utilized that

$$\underline{\mu}^d = \mathrm{E}\{\underline{\boldsymbol{d}}\} = \int \underline{d} \cdot f(\underline{d}) \; \mathrm{d}\underline{d} = \iiint \underline{d} \cdot f(\underline{x},\underline{d},\underline{v}) \; \mathrm{d}\underline{x}\,\mathrm{d}\underline{d}\,\mathrm{d}\underline{v} \qquad (10)$$

holds (see for instance [7]). With (8) and Bayes' rule, the joint density function $f(\underline{x},\underline{d},\underline{v})$ of $\underline{\boldsymbol{x}}$, $\underline{\boldsymbol{d}}$ and $\underline{\boldsymbol{v}}$ can be written according to

$$\begin{aligned} f(\underline{x},\underline{d},\underline{v}) &= f(\underline{d}|\underline{x},\underline{v}) \cdot f(\underline{x},\underline{v}) \\ &= \delta\big(\underline{d} - \underline{h}(\underline{x},\underline{v})\big) \cdot f(\underline{x},\underline{v}), \end{aligned} \qquad (11)$$

where $\delta(\underline{x} - \underline{\mu})$ is the Dirac delta distribution at position $\underline{\mu}$.

As white measurement noise is assumed, the random vectors $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{v}}$ are independent and thus, the joint density $f(\underline{x},\underline{v})$ is given by $f(\underline{x},\underline{v}) = f(\underline{x}) \cdot f(\underline{v})$.

Plugging (11) into (10) and utilizing the sifting property of the Dirac delta distribution results in

$$\begin{aligned}\underline{\mu}^d &= \mathrm{E}_{\underline{x},\underline{v}}\left\{\underline{h}\left(\underline{\boldsymbol{x}}_P,\underline{\boldsymbol{v}}\right)\right\} = \iint \underline{h}\left(\underline{x}_P,\underline{v}\right)\cdot f(\underline{x})\cdot f(\underline{v})\ \mathrm{d}\underline{x}\,\mathrm{d}\underline{v}\\ &= \iint \underline{h}\left(\underline{x}_P,\underline{v}\right)\cdot f\left(\underline{x}_P\right)\cdot f(\underline{v})\ \mathrm{d}\underline{x}_P\,\mathrm{d}\underline{v} = \mathrm{E}_{\underline{x}_P,\underline{v}}\left\{\underline{h}\left(\underline{\boldsymbol{x}}_P,\underline{\boldsymbol{v}}\right)\right\}\ , \end{aligned} \tag{12}$$

where the velocity $\underline{x}_V$ is marginalized, because the variable $\underline{x}_V$ has no influence on the mean. Hence, the mean of $\underline{\boldsymbol{d}}$ can be calculated directly based on the nonlinear function $\underline{h}(\cdot)$ and the density of $\underline{\boldsymbol{x}}_P$ and $f(\underline{v})$. By substituting (7) in (12), the predicted measurement μ_i^d for the ith range between landmark i and the target is given by

$$\begin{aligned}\mu_i^d &= \mathrm{E}_{\underline{x}_P,\underline{v}}\left\{h_i\left(\underline{\boldsymbol{x}}_P,\underline{\boldsymbol{v}}\right)\right\}\\ &= \mathrm{E}\left\{\left(\underline{S}_i-\underline{\boldsymbol{x}}_P\right)^{\mathrm{T}}\cdot\left(\underline{S}_i-\underline{\boldsymbol{x}}_P\right)\right\} - 2\cdot\mathrm{E}\left\{\left(\underline{S}_i-\underline{\boldsymbol{x}}_P\right)^{\mathrm{T}}\cdot\underline{\boldsymbol{v}}_i\right\} + \mathrm{E}\left\{\underline{\boldsymbol{v}}_i^{\mathrm{T}}\cdot\underline{\boldsymbol{v}}_i\right\}\\ &= \left(\underline{S}_i-\underline{\mu}^{x_P}\right)^{\mathrm{T}}\cdot\left(\underline{S}_i-\underline{\mu}^{x_P}\right) + \mathrm{Tr}\left(\mathbf{C}^{x_P}\right) + \mathrm{Tr}\left(\mathbf{C}_i^{v}\right)\ .\end{aligned}$$

Covariance

For calculating the covariance $\mathbf{C}^d$ of the measurement process, it is exploited that

$$\begin{aligned}\mathbf{C}^d &= \mathrm{E}\left\{\left(\underline{\boldsymbol{d}}-\underline{\mu}^d\right)\cdot\left(\underline{\boldsymbol{d}}-\underline{\mu}^d\right)^{\mathrm{T}}\right\}\\ &= \iiint \left(\underline{d}-\underline{\mu}^d\right)\cdot\left(\underline{d}-\underline{\mu}^d\right)^{\mathrm{T}}\cdot f\left(\underline{x}_P,\underline{v},\underline{d}\right)\ \mathrm{d}\underline{x}_P\,\mathrm{d}\underline{v}\,\mathrm{d}\underline{d}\\ &= \iint \underline{h}\left(\underline{x}_P,\underline{v}\right)\cdot\underline{h}\left(\underline{x}_P,\underline{v}\right)^{\mathrm{T}}\cdot f\left(\underline{x}_P\right)\cdot f(\underline{v})\ \mathrm{d}\underline{x}_P\,\mathrm{d}\underline{v} - \underline{\mu}^d\cdot\left(\underline{\mu}^d\right)^{\mathrm{T}}\\ &= \mathrm{E}\left\{\underline{h}\left(\underline{\boldsymbol{x}}_P,\underline{\boldsymbol{v}}\right)\cdot\underline{h}\left(\underline{\boldsymbol{x}}_P,\underline{\boldsymbol{v}}\right)^{\mathrm{T}}\right\} - \underline{\mu}^d\cdot\left(\underline{\mu}^d\right)^{\mathrm{T}}\ .\end{aligned} \tag{13}$$

Similar to the mean, the covariance does not depend on the velocity. With (13) and (7), the covariance matrix of $\underline{\boldsymbol{d}}$ consists of the single entries $C_{i,j}^d$ given by

$$C_{i,j}^d = \mathrm{E}_{\underline{x}_P,\underline{v}_i,\underline{v}_j}\left\{h_i\left(\underline{\boldsymbol{x}}_P,\underline{\boldsymbol{v}}_i\right)\cdot h_j\left(\underline{\boldsymbol{x}}_P,\underline{\boldsymbol{v}}_j\right)\right\} - \mu_i^d\cdot\mu_j^d. \tag{14}$$

The expected value $\mathrm{E}_{\underline{x}_P,\underline{v}_i,\underline{v}_j}\{\cdot\}$ in (14) can be decomposed into nine summands representing the product $h_i\big(\underline{\boldsymbol{x}}_P,\underline{\boldsymbol{v}}_i\big)\cdot h_j\big(\underline{\boldsymbol{x}}_P,\underline{\boldsymbol{v}}_j\big)$. In the following, the solution of the expected value $\mathrm{E}_{\underline{x}_P,\underline{v}_i,\underline{v}_j}\{\cdot\}$ for each of the nine summands is derived. The first term is given by

$$\begin{aligned}\mathrm{E}_{\underline{x}_P}\left\{\big(\underline{S}_i-\underline{\boldsymbol{x}}_P\big)^{\mathrm{T}}\cdot\big(\underline{S}_i-\underline{\boldsymbol{x}}_P\big)\cdot\big(\underline{S}_j-\underline{\boldsymbol{x}}_P\big)^{\mathrm{T}}\cdot\big(\underline{S}_j-\underline{\boldsymbol{x}}_P\big)\right\}=&\\ \underline{A}^{\mathrm{T}}\cdot\underline{A}\cdot\underline{B}^{\mathrm{T}}\cdot\underline{B}+(\underline{A}^{\mathrm{T}}\cdot\underline{A}+\underline{B}^{\mathrm{T}}\cdot\underline{B})\cdot\mathrm{Tr}(\mathbf{C}^{x_P})+&\\ 4\cdot\underline{A}^{\mathrm{T}}\cdot\mathbf{C}^{x_P}\cdot\underline{B}+(\mathrm{Tr}(\mathbf{C}^{x_P}))^2+2\cdot\underline{1}_M^{\mathrm{T}}\cdot(\mathbf{C}^{x_P}\odot\mathbf{C}^{x_P})\cdot\underline{1}_M\,,&\end{aligned}\qquad(15)$$

where $\underline{A}\triangleq\underline{S}_i-\underline{\mu}^{x_P}$, $\underline{B}\triangleq\underline{S}_j-\underline{\mu}^{x_P}$, and $\odot$ is the Hadamard (element-wise) product. $\underline{1}_M$ is a vector consisting of ones, where the variable $M=3$ stands for the three-dimensional space.

In four of the summands, the noise $\underline{\boldsymbol{v}}$ occurs in first or third order and thus the expected value is zero, because the noise process $\underline{\boldsymbol{v}}$ is zero-mean and it is uncorrelated to the target position $\underline{\boldsymbol{x}}_P$. The remaining four expected values are given by

$$\mathrm{E}_{\underline{x}_P,\underline{v}_j}\left\{\big(\underline{S}_i-\underline{\boldsymbol{x}}_P\big)^{\mathrm{T}}\cdot\big(\underline{S}_i-\underline{\boldsymbol{x}}_P\big)\cdot\underline{\boldsymbol{v}}_j^{\mathrm{T}}\cdot\underline{\boldsymbol{v}}_j\right\}=\big(\underline{A}^{\mathrm{T}}\cdot\underline{A}+\mathrm{Tr}\big(\mathbf{C}^{x_P}\big)\big)\cdot\mathrm{Tr}\big(\mathbf{C}_j^{v}\big)\,,\qquad(16)$$

$$\mathrm{E}_{\underline{x}_P,\underline{v}_i}\left\{\underline{\boldsymbol{v}}_i^{\mathrm{T}}\cdot\underline{\boldsymbol{v}}_i\cdot\big(\underline{S}_j-\underline{\boldsymbol{x}}_P\big)^{\mathrm{T}}\cdot\big(\underline{S}_j-\underline{\boldsymbol{x}}_P\big)\right\}=\big(\underline{B}^{\mathrm{T}}\cdot\underline{B}+\mathrm{Tr}\big(\mathbf{C}^{x_P}\big)\big)\cdot\mathrm{Tr}\big(\mathbf{C}_i^{v}\big)\,,\qquad(17)$$

$$\mathrm{E}_{\underline{x}_P,\underline{v}_i,\underline{v}_j}\left\{\big(\underline{S}_i-\underline{\boldsymbol{x}}_P\big)^{\mathrm{T}}\cdot\underline{\boldsymbol{v}}_i\cdot\big(\underline{S}_j-\underline{\boldsymbol{x}}_P\big)^{\mathrm{T}}\cdot\underline{\boldsymbol{v}}_j\right\}=\Big(\underline{B}^{\mathrm{T}}\cdot\mathbf{C}_{i,j}^{v}\cdot\underline{A}+\underline{1}_M^{\mathrm{T}}\cdot\big(\mathbf{C}^{x_P}\odot\mathbf{C}_{i,j}^{v}\big)\cdot\underline{1}_M\Big)\,,\qquad(18)$$

$$\mathrm{E}_{\underline{v}_i,\underline{v}_j}\left\{\underline{\boldsymbol{v}}_i^{\mathrm{T}}\cdot\underline{\boldsymbol{v}}_i\cdot\underline{\boldsymbol{v}}_j^{\mathrm{T}}\cdot\underline{\boldsymbol{v}}_j\right\}=\mathrm{Tr}\big(\mathbf{C}_i^{v}\big)\cdot\mathrm{Tr}\big(\mathbf{C}_j^{v}\big)+2\cdot\underline{1}_M^{\mathrm{T}}\cdot\big(\mathbf{C}_{i,j}^{v}\odot\mathbf{C}_{i,j}^{v}\big)\cdot\underline{1}_M\,.\qquad(19)$$

Plugging the results of the five non-zero expected values (15)–(19) into (14), the entry i,j of the covariance matrix $\mathbf{C}^d$ is given by

$$\begin{aligned}C_{i,j}^d=&\,4\cdot\underline{A}^{\mathrm{T}}\cdot\mathbf{C}^{x_P}\cdot\underline{B}+2\cdot\underline{1}_M^{\mathrm{T}}\cdot\big(\mathbf{C}^{x_P}\odot\mathbf{C}^{x_P}\big)\cdot\underline{1}_M\\&+4\Big(\underline{B}^{\mathrm{T}}\cdot\mathbf{C}_{i,j}^{v}\cdot\underline{A}+\underline{1}_M^{\mathrm{T}}\cdot\big(\mathbf{C}^{x_P}\odot\mathbf{C}_{i,j}^{v}\big)\cdot\underline{1}_M\Big)+2\cdot\underline{1}_M^{\mathrm{T}}\cdot\big(\mathbf{C}_{i,j}^{v}\odot\mathbf{C}_{i,j}^{v}\big)\cdot\underline{1}_M\,,\end{aligned}$$

which can be simplified to

$$C_{i,j}^d=4\cdot\underline{A}^{\mathrm{T}}\cdot\mathbf{C}_{i,j}^{'}\cdot\underline{B}+2\cdot\underline{1}_M^{\mathrm{T}}\cdot\Big(\mathbf{C}_{i,j}^{'}\odot\mathbf{C}_{i,j}^{'}\Big)\cdot\underline{1}_M$$

with covariance matrix $\mathbf{C}_{i,j}^{'}\triangleq\mathbf{C}^{x_P}+\mathbf{C}_{i,j}^{v}$.

Cross-Covariance

Finally, the cross-covariance $\mathbf{C}^{x,d}$ is required, which consists of the cross-covariance between the position $\mathbf{C}^{x_P,d}$ and the velocity $\mathbf{C}^{x_V,d}$. Similar to the covariance $\mathbf{C}^d$ in (13), the cross-covariance $\mathbf{C}^{x,d}$ is calculated by

$$\mathbf{C}^{x,d} = \mathrm{E}_{\underline{x},\underline{d}}\left\{\left(\underline{\boldsymbol{x}} - \underline{\mu}^p\right)\cdot\left(\underline{\boldsymbol{d}} - \underline{\mu}^d\right)^{\mathrm{T}}\right\} = \mathrm{E}_{\underline{x},\underline{v}}\left\{\underline{\boldsymbol{x}}\cdot\underline{h}(\underline{\boldsymbol{x}},\underline{\boldsymbol{v}})^{\mathrm{T}}\right\} - \underline{\mu}^p\cdot\left(\underline{\mu}^d\right)^{\mathrm{T}} . \qquad (20)$$

For calculating the cross-covariance $\mathbf{C}^{x_P,d}$ of the position, (20) is used. The cross-covariance of the ith column is calculated by

$$\begin{aligned}\underline{C}_i^{x_P,d} &= \mathrm{E}_{\underline{x}_P,\underline{v}_i}\left\{\underline{\boldsymbol{x}}_P\cdot h_i\left(\underline{\boldsymbol{x}}_P,\underline{\boldsymbol{v}}_i\right)\right\} - \underline{\mu}^{x_P}\cdot\mu_i^d \\ &= -2\cdot\mathbf{C}^p\cdot\begin{bmatrix}\mathbf{I}_{M,M}\\ \mathbf{0}_{M,M}\end{bmatrix}\cdot\underline{A} \\ &= -2\cdot\mathbf{C}^{x_P}\cdot\underline{A}\,, \qquad (21)\end{aligned}$$

where $\mathbf{I}_{M,M}$ is the identity matrix and $\mathbf{0}_{M,M}$ is a zero matrix, both with dimension $M\times M$. Analogously, the cross-covariance $\mathbf{C}^{x_V,d}$ of the velocity is given by

$$\begin{aligned}\underline{C}_i^{x_V,d} &= \mathrm{E}_{\underline{x}_P,\underline{x}_V,\underline{v}_i}\left\{\underline{\boldsymbol{x}}_V\cdot h_i\left(\underline{\boldsymbol{x}}_P,\underline{\boldsymbol{v}}_i\right)\right\} - \underline{\mu}^{x_V}\cdot\mu_i^d = \mathbf{C}^{x_V,x_P}\cdot\left(\mathbf{C}^{x_P}\right)^{-1}\cdot\underline{C}_i^{x_P,d} \\ &= -2\cdot\mathbf{C}^{x_V,x_P}\cdot\underline{A}\,.\end{aligned}$$

with $\mathbf{C}^{x_V,x_P}$ being the cross-covariance between velocity and position taken from the predicted covariance $\mathbf{C}^p$. The last equality follows from substituting $\underline{C}_i^{x_P,d}$ by (21).

4.3 Summary

The proposed algorithm makes use of a state estimator for recursively calculating the position and the velocity of the object. The state estimator consists of two steps, the prediction and filter step, as described above. In the following, a short wrap-up for the filter step is given in vector-matrix notation allowing for a straightforward and computationally efficient implementation.

The filter step provides an estimate of the mean $\underline{\mu}^e$ and covariance matrix $\mathbf{C}^e$ according to (9). Here, the unknown moments $\underline{\mu}^d$, $\mathbf{C}^d$, and $\mathbf{C}^{x,d}$ are given by

$$\begin{aligned}
\underline{\mu}^d &= (\mathbf{V} \odot \mathbf{V})^{\mathrm{T}} \cdot \underline{1}_M + \underline{1}_N \cdot \mathrm{Tr}\left(\mathbf{P} \cdot \mathbf{C}^p \cdot \mathbf{P}^{\mathrm{T}}\right) + \mathbf{O}^{\mathrm{T}} \cdot \mathrm{diag}\left(\mathbf{C}^v\right) , \\
\mathbf{C}^d &= \mathbf{O}^{\mathrm{T}} \cdot \left(4 \cdot \left(\mathrm{vec}(\mathbf{V}) \cdot \mathrm{vec}(\mathbf{V})^{\mathrm{T}}\right) \odot \mathbf{T} + 2 \cdot \mathbf{T} \odot \mathbf{T}\right) \cdot \mathbf{O} , \\
\mathbf{C}^{x,d} &= -2 \cdot \mathbf{C}^p \cdot \mathbf{P}^{\mathrm{T}} \cdot \mathbf{V} ,
\end{aligned} \tag{22}$$

with

$$\begin{aligned}
\mathbf{S} &\triangleq \begin{bmatrix} \underline{S}_1 & \dots & \underline{S}_N \end{bmatrix} , \\
\mathbf{P} &\triangleq \begin{bmatrix} \mathbf{I}_{M,M} & \mathbf{0}_{M,M} \end{bmatrix} , \\
\mathbf{V} &\triangleq \mathbf{S} - \left(\underline{1}_N\right)^{\mathrm{T}} \otimes \left(\mathbf{P} \cdot \underline{\mu}^p\right) , \\
\mathbf{O} &\triangleq \mathbf{I}_{N,N} \otimes \underline{1}_M , \\
\mathbf{T} &\triangleq \mathbf{C}^v + \mathbf{1}_{N,N} \otimes \left(\mathbf{P} \cdot \mathbf{C}^p \cdot \mathbf{P}^{\mathrm{T}}\right) ,
\end{aligned}$$

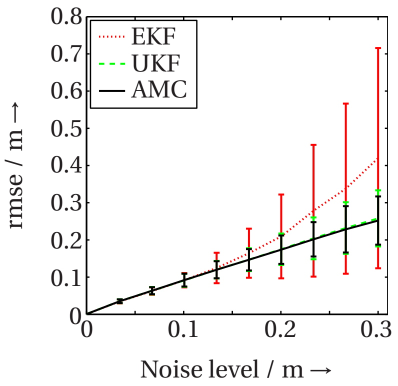
where $\otimes$ is the Kronecker product, vec($\mathbf{V}$) is the vectorized version of the matrix $\mathbf{V}$, and $\mathbf{1}_{N,N}$ is a one matrix. The variable $M = 3$ stands for the three-dimensional space and N for the number of landmarks. Furthermore, the measured ranges $\underline{\hat{r}}$ have to be squared according to $\underline{\hat{d}} = \underline{\hat{r}} \odot \underline{\hat{r}}$.

4.4 Computational Complexity

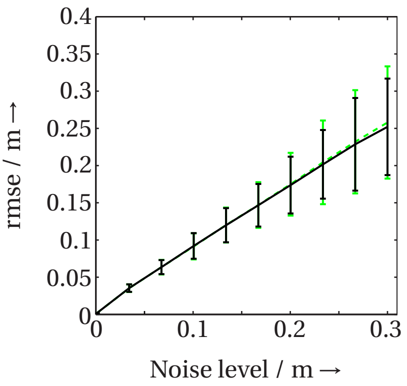
In order to calculate the required moments $\underline{\mu}^d$, $\mathbf{C}^d$, and $\mathbf{C}^{x,d}$ for the filter step, the computational complexity of the proposed approach is in $\mathcal{O}\left(N^2 \cdot M^3\right)$. For comparison, the computational complexity of calculating the matrix root in the unscented Kalman filter is in $\mathcal{O}\left(N^3 \cdot M^3\right)$.

5 Experiments

In this section, simulations and real-world experiments are used for comparing the proposed approach (AMC) with standard state estimators like the extended Kalman filter (EKF) and the unscented Kalman filter (UKF). Furthermore, a closed-form solution (CFS) [4] is considered for the real world experiment. All three estimators (AMC, EKF, and UKF) make use of the measurement equation (4) and the system model (1).



(a) Result of the three estimators AMC, UKF, and EKF.



(b) Zoom of the result for AMC and UKF only.

Figure 1: The average rmse and its standard deviation for different noise levels.

5.1 Simulation

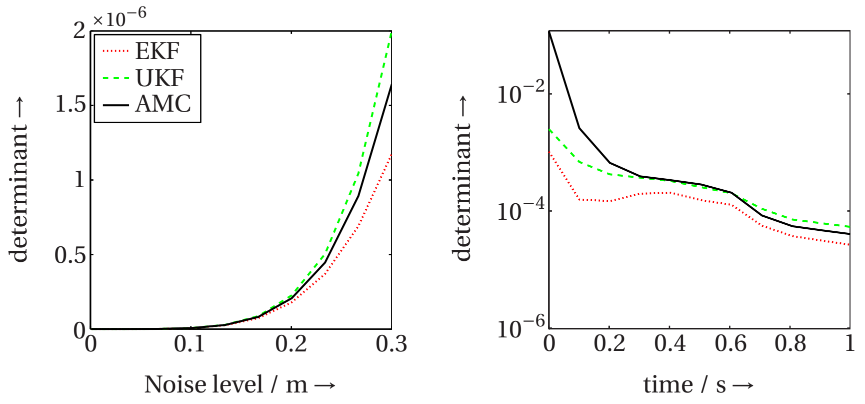
To compare the performance of the three algorithms (AMC, EKF, UKF), range measurements to four landmarks with positions

$$\mathbf{S} = \begin{bmatrix} \underline{S}_1 & \dots & \underline{S}_4 \end{bmatrix} = \begin{bmatrix} -2 & -2 & 2 & 2 \\ -2 & 2 & -2 & 2 \\ 0 & 0 & 0 & 2 \end{bmatrix} \mathrm{m}$$

are performed. For generating the noisy range measurements, ten different noise levels $\mathbf{C}_i^v = \mathbf{I}_{3,3} \cdot \sigma_n^2$ for each landmark $i = 1, \dots, 4$ are considered, where $\sigma_n = \frac{n-1}{30}$ m with $n = 1, \dots, 10$. For each noise level, 1000 random object trajectories are generated. Each trajectory consists of 100 measurements to all landmarks.

The Gaussian density representing the initial state at time step $k = 0$ has zero mean and an initial covariance $\mathbf{C}_0 = 10 \cdot \mathbf{I}_{6,6}$. The sampling time is $T = 0.1$s. The process noise is $C_{c,x}^w = 0.01$, $C_{c,y}^w = 0.01$, and $C_{c,z}^w = 0.0001$.

For each random trajectory, the root mean square error (rmse) between the estimate and the ground truth is calculated. In Figure 1, the average rmse and its standard deviation over the 1000 test runs for all the different noise levels is shown.



(a) Average of the determinants over all test runs.

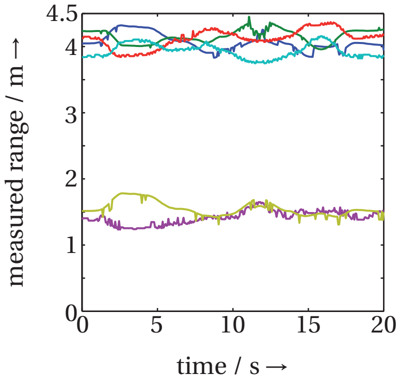
(b) Determinant for one single run at noise level 0.3 m.

Figure 2: Determinant of the covariance matrix calculated by AMC, UKF, and EKF for different noise levels.

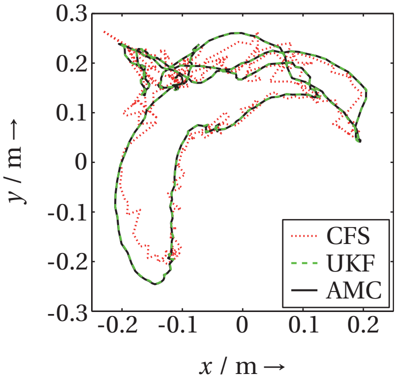
For small noise, all three estimators perform similar. If the noise increases, the rmse of the EKF increases much stronger compared to the other two estimators. For a high noise level, the UKF and the proposed approach present comparable results, where the average rmses and the standard deviations of the AMC are slightly smaller.

All three estimators have to evaluate (5), but compared to AMC and EKF, the UKF additionally requires matrix roots for determining the sigma points. Consequently, the computational demand of the UKF is much higher. For calculating the required moments in (22), only vector-matrix products and no additional matrix inversions or roots are required. Of course, the complexity of the EKF is lower compared to the AMC, but for a high noise level, the AMC performs significantly better.

The average determinant of the covariance matrix of the position estimate $\underline{\boldsymbol{x}}_{k,P}$ of all test runs is shown in Figure 2a. In Figure 2b, one single test run at noise level 0.3 m exemplarily demonstrates the evolution of the determinant of the covariance over the time. It can be seen that the covariance of the EKF decreases too quickly. Due to the linearization based on first-order Taylor-series expansions, the determinant of the EKF is too small and thus the EKF is too certain about its



(a) Measured ranges between one microphone and six landmarks.



(b) Estimated trajectories of the AMC, UKF, and CFS.

Figure 3: The measured ranges and estimated trajectories.

estimate. Hence, the estimation results are inconsistent, which is often a problem when using an EKF. On the other hand, sample-based approaches as the UKF or analytic approaches as the AMC overcome this problem. The determinant of the AMC is smaller compared to the determinant of the UKF. Furthermore, as described before, the rmse of the AMC is smaller as well. All together, the AMC is more informative compared to the UKF.

5.2 Experiment

The experiment considers a tracking system for extended range telepresence [8] for generating the range data. Six loudspeakers emit signals that are received by four microphones attached to the object. Based on the emitted and received signals, ranges between each microphone and each loudspeaker are calculated. The measured ranges for one microphone are shown in Figure 3a.

The initial mean is zero and the initial covariance is $\mathbf{C}_0 = 10 \cdot \mathbf{I}_{6,6}$. The process noise was set to $C^w_{c,x} = 0.04$, $C^w_{c,y} = 0.04$, and $C^w_{c,z} = 0.01$. The sampling time is $T = 0.0625$s. The standard deviation of the measurement noise is assumed to be 0.1m.

It can be seen in Figure 3b that a closed-form solution as in [4] provides poor results, if the range measurements are noisy. On the other hand, state estimators like the AMC or UKF can deal with noisy range measurements. The results of the state estimators (AMC, UKF) are similar, since the noise level is small (please recall Section 5.1). The average of the absolute position error of all microphone pairs is 0.0336 m for the AMC, 0.0337 m for the UKF, 0.0342 m for the EKF, and 0.0423 m for the closed-form solution (CFS). Even if the AMC merely provides a slightly better result than the UKF and the EKF, this experiment demonstrates that the AMC also works in a real-world scenario.

6 Conclusions

This paper presents a Bayesian state estimator for range-based localization. Assuming that the object's state estimate is Gaussian, the required moments, i.e., mean and covariance are calculated analytically. Compared to well-known closed-form solutions assuming exact (or almost exact) range measurements, the proposed approach allows taking subspace measurements into account. Furthermore, AMC facilitates dynamic localization and provides information about the object's position and velocity uncertainty.

Due to the assumption that state and measurement are jointly Gaussian, AMC provides an optimal stochastic linearization, which takes the entire Gaussian density into account. Other Gaussian estimators based on linearization merely consider points, i.e., EKF uses first-order Taylor-series expansion around the mean and the UKF calculates sigma-points for stochastic linearization. As demonstrated in the simulations and experiments, AMC leads to more accurate and more consistent localization results compared to the EKF or UKF. The computational demand of AMC is lower than that of the UKF, because no matrix roots have to be calculated. Furthermore, no parameter adaption for sigma-point calculation is necessary.

It is intended to utilize the analytic expressions for the mean and covariance for improving existing results in research fields such as sensor placement and scheduling or simultaneous localization and mapping (SLAM).

References

[1] James J. Caffery, Jr. A New Approach to the Geometry of TOA Location. In *Proceedings of 55th IEEE Vehicular Technology Conference*, pages 1942–1949, 2000.

[2] Kwok-Wai Cheung and Hing Cheung So. A Multidimensional Scaling Framework for Mobile Location Using Time-of-Arrival Measurements. *IEEE Transactions on Signal Processing*, 53(2):460–470, February 2005.

[3] Martin Clark and Richard Vinter. A New Class of Moment Matching Filters for Nonlinear Tracking and Estimation Problems. In *2006 IEEE Nonlinear Statistical Signal Processing Workshop*, pages 108–112, September 2006.

[4] Uwe D. Hanebeck and Günther Schmidt. Closed-Form Elliptic Location with an Arbitrary Array Topology. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3070–3073, 1996.

[5] Simon J. Julier and Jeffrey K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[6] Dimitris E. Manolakis. Efficient Solution and Performance Analysis of 3-D Position Estimation by Trilateration. *IEEE Transactions on Aerospace and Electronic Systems*, 32(4):1239–1248, October 1996.

[7] Athanasios Papoulis and S. Unnikrishna Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill Science/Engineering/Math, 4th edition, 2002.

[8] Patrick Rößler, Frederik Beutler, Uwe D. Hanebeck, and Norbert Nitzsche. Motion Compression Applied to Guidance of a Mobile Teleoperator. In *Proceedings of the 2005 IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2495–2500, 2005.

[9] Stanley F. Schmidt. *Advances in Control Systems, Volume 3*, chapter Application of State-Space Methods to Navigation Problems, pages 293–340. Academic Press, New York, 1966.

[10] Fernando Seco, Antonio R. Jiménez, Carlos Prieto, Javier Roa, and Katerina Koutsou. A Survey of Mathematical Methods for Indoor Localization. In *IEEE International Symposium on Intelligent Signal Processing*, pages 9–14, August 2009.

[11] Federico Thomas and Lluís Ros. Revisiting trilateration for robot localization. *IEEE Transactions on Robotics*, 21:93–101, 2005.

[12] Greg Welch, B. Danette Allen, Adrian Ilie, and Gary Bishop. Measurement Sample Time Optimization for Human Motion Tracking/Capture Systems. In *Proceedings of Trends and Issues in Tracking for Virtual Environments, Workshop at the IEEE Virtual Reality 2007 Conference*, 2007.

Paper M

Semi-Analytic Stochastic Linearization for Range-Based Pose Tracking

Authors: Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck

Edited version of the paper:

F. Beutler, M. F. Huber, and U. D. Hanebeck. Semi-Analytic Stochastic Linearization for Range-Based Pose Tracking. In *Proceedings of the 2010 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 44–49, Salt Lake City, UT, USA, September 2010.

Semi-Analytic Stochastic Linearization for Range-Based Pose Tracking

Frederik Beutler*, Marco F. Huber**, and Uwe D. Hanebeck*

* Intelligent Sensor-Actuator-Systems Laboratory (ISAS)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{beutler|uwe.hanebeck}@ieee.org

** Variable Image Acquisition and Processing Research Group
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB
Karlsruhe, Germany
marco.huber@ieee.org

Abstract

In range-based pose tracking, the translation and rotation of an object with respect to a global coordinate system has to be estimated. The ranges are measured between the target and the global frame. In this paper, an intelligent decomposition is introduced in order to reduce the computational effort for pose tracking. Usually, decomposition procedures only exploit conditionally linear models. In this paper, this principle is generalized to conditionally integrable substructures and applied to pose tracking. Due to a modified measurement equation, parts of the problem can even be solved analytically.

1 Introduction

In many applications, the estimation of an object pose, i.e., the translation and the rotation, is essential, e.g., in large-scale telepresence (see Figure 1 and [11]). In [11], the pose of a human has to be tracked in order to steer the teleoperator. For tracking the user's pose, several emitters are located at known positions in a global coordinate system. They are emitting signals that are received by several sensors attached to the target frame. Based on the emitted and received signals, ranges between emitters and sensors can be determined. Due to disturbances and a nonlinear measurement equation, which describes the relationship between measured ranges and pose, an exact estimator cannot be applied and so

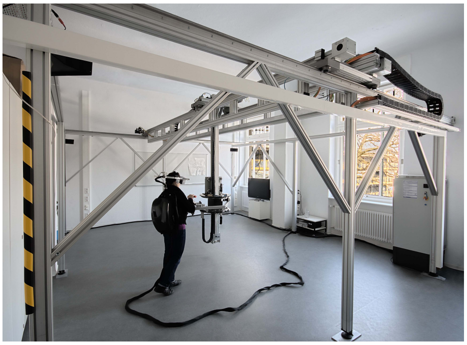


Figure 1: Person using large-scale telepresence system.

approximative estimators have to be used. Algorithms for estimating the pose based on range measurements are closed-form solutions [2], gradient descent algorithms [11], or state estimators [3]. Popular state estimators rely on the Gaussian assumption [6–8], e.g., the unscented Kalman filter, where all involved random variables are described by mean and covariance. Furthermore, the random variable for the measurement and for the state are assumed to be jointly Gaussian distributed. In order to calculate the mean and covariance, sample-based approaches are used. For an efficient implementation, the structure of the measurement and system equation can be exploited and so the number of sample points can be reduced. For conditionally linear substructures, the reader is referred to [3] and [9]. In order to reduce the number of sample points, the decomposition in conditionally linear substructures can be generalized to conditionally integrable substructures. In this case, nonlinear parts of the problem can be solved in closed form.

Compared to the previous approach in [3], where the density of the translation and rotation has to be approximated for the filter step, merely the density of the rotation is processed approximately in the proposed approach, while the remaining part can be calculated in closed form. In doing so, the number of sample points can be decreased.

The structure of the paper is as follows. In the problem formulation in Section 2, the measurement equation (see Section 2.1) and system equation (see Section 2.2) for pose tracking are described. The proposed approach makes use of a state estimator, the *generalized Gaussian assumed density filter*, which is described in Section 2.3. This filter consists of a prediction step and a filter step.

In the filter step, the assumption that measurements and the state are jointly Gaussian distributed is applied. The proposed approach for pose tracking is shown in Section 3. First, the measurement equation is modified in Section 3.1. Based on this modified measurement equation, parts of the problem can be solved in closed form, where first the decomposition is explained in Section 3.2. This decomposition is then used in order to calculate the mean (Section 3.3), the covariance (Section 3.4), and the cross-covariance (Section 3.5), which is required for the filter step. A short wrap-up of the filter step is described in Section 3.6. The proposed approach is compared to the standard decomposition via simulations in Section 4. Finally, the paper closes with conclusions.

2 Problem Formulation

In pose tracking, the translation $\underline{\boldsymbol{t}}$ and the rotation $\underline{\boldsymbol{r}}$ of an extended object have to be estimated in three-dimensional space. Due to the fact that the object is in motion, it is essential to consider the dynamic behavior of the object by means of the translation velocity $\underline{\dot{\boldsymbol{t}}}$ and angular velocity $\underline{\boldsymbol{\omega}}$. In order to estimate the state of the object

$$\underline{\boldsymbol{x}}_k \triangleq \begin{bmatrix} \underline{\boldsymbol{t}}_k^{\mathrm{T}} & \underline{\dot{\boldsymbol{t}}}_k^{\mathrm{T}} & \underline{\boldsymbol{r}}_k^{\mathrm{T}} & \underline{\boldsymbol{\omega}}_k^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$$

at time step $k = 0, 1, \ldots$, a state estimator is used.

2.1 Measurement Equation

The measurement equation describes the relationship between the measurement and the state. In range-based pose estimation, the measured ranges depend on known landmarks located on the extended target, known landmarks in a static global coordinate system, and the unknown translation and rotation of the object with respect to the global coordinate system. The relationship between measured ranges and the translation and rotation is given by

$$\boldsymbol{d}_{i,j,k} = \left\| \underline{L}_j - \mathbf{D}\left(\underline{\boldsymbol{r}}_k\right) \cdot \underline{M}_i - \underline{\boldsymbol{t}}_k - \underline{\boldsymbol{v}}_{i,j,k} \right\|_2 , \tag{1}$$

where $\mathbf{D}(\cdot)$ is the rotation matrix parametrized by the vector $\underline{\boldsymbol{r}}_k$, which describes the rotation. $\underline{L}_j$ is the position of the jth landmark with respect to the global coordinate system and $\underline{M}_i$ is the position of the ith landmark with respect to the target coordinate system. $\underline{\boldsymbol{v}}_{i,j,k}$ is the measurement noise between landmark $\underline{L}_j$ and landmark $\underline{M}_i$. $\boldsymbol{d}_{i,j,k}$ is the measured range between these two landmarks, while $\underline{\boldsymbol{d}}_k$ comprises all possible measurements between global and target landmarks in the following.

The rotation matrix can be parameterized by quaternions, Euler angles, roll-pitch-yaw, or a rotation vector [1]. In this paper, the parameterization of the rotation matrix is based on the rotation vector [3] according to

$$\mathbf{D}(\underline{\boldsymbol{r}}_k) = \mathbf{I} + \frac{\sin(\|\underline{\boldsymbol{r}}_k\|)}{\|\underline{\boldsymbol{r}}_k\|} \cdot \mathbf{R}(\underline{\boldsymbol{r}}_k) + \frac{1-\cos(\|\underline{\boldsymbol{r}}_k\|)}{\|\underline{\boldsymbol{r}}_k\|^2} \cdot \mathbf{R}(\underline{\boldsymbol{r}}_k) \cdot \mathbf{R}(\underline{\boldsymbol{r}}_k) \,,$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{R}$ is a skew-symmetric matrix given by

$$\mathbf{R}(\underline{\boldsymbol{r}}_k) = \begin{bmatrix} 0 & -\boldsymbol{r}_z & \boldsymbol{r}_y \\ \boldsymbol{r}_z & 0 & -\boldsymbol{r}_x \\ -\boldsymbol{r}_y & \boldsymbol{r}_x & 0 \end{bmatrix}$$

for the three-dimensional space.

2.2 System equation

The system equation describes how the state evolves over time. In state estimation, the system equation is used to predict the state at the next time step when a new measurement is taken. In the considered example, two separate motion models are assumed, one for the translation and the second for the rotation. The discrete-time motion model for the translation is given by a linear equation according to

$$\begin{bmatrix} \underline{\boldsymbol{t}}_{k+1} \\ \underline{\dot{\boldsymbol{t}}}_{k+1} \end{bmatrix} = \mathbf{A} \cdot \begin{bmatrix} \underline{\boldsymbol{t}}_{k} \\ \underline{\dot{\boldsymbol{t}}}_{k} \end{bmatrix} + \underline{\boldsymbol{w}}_k^t \,, \tag{2}$$

where the process noise $\underline{\boldsymbol{w}}_k^t$ is assumed to be Gaussian distributed with covariance $\mathbf{Q}^t$ [12]. The matrix $\mathbf{A}$ and the covariance $\mathbf{Q}^t$ are given by

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & T \cdot \mathbf{I} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} , \quad \mathbf{Q}^t = \begin{bmatrix} \frac{T^3}{3}\mathbf{Q}_c^t & \frac{T^2}{2}\mathbf{Q}_c^t \\ \frac{T^2}{2}\mathbf{Q}_c^t & T \cdot \mathbf{Q}_c^t \end{bmatrix} ,$$

where T is the sampling time and $\mathbf{Q}_c^t = \operatorname{diag}\left(\begin{bmatrix} Q_{c,x}^t & Q_{c,y}^t & Q_{c,z}^t \end{bmatrix}\right)$ is the covariance of the process noise from the continuous-time system model.

The evolution of the rotation vector over time is described by means of a nonlinear equation [4, 5]

$$\underline{\boldsymbol{r}}_{k+1} = \underline{\boldsymbol{r}}_k + \underbrace{T \cdot \left(\mathbf{I} + 0.5 \cdot \mathbf{R}(\underline{\boldsymbol{r}}_k) + a(\underline{\boldsymbol{r}}_k) \cdot \mathbf{R}(\underline{\boldsymbol{r}}_k) \cdot \mathbf{R}(\underline{\boldsymbol{r}}_k)\right)}_{\triangleq \Lambda(\underline{\boldsymbol{r}}_k)} \cdot \underline{\boldsymbol{\omega}}_k \,, \tag{3}$$

with

$$a(\underline{\boldsymbol{r}}) \triangleq \frac{1 - 0.5 \cdot \left\|\underline{\boldsymbol{r}}_k\right\|}{\left\|\underline{\boldsymbol{r}}_k\right\|^2} \cdot \cot\left(\frac{\left\|\underline{\boldsymbol{r}}_k\right\|}{2}\right) .$$

The system model for the angular velocity $\underline{\boldsymbol{\omega}}_k$ is assumed to be

$$\underline{\boldsymbol{\omega}}_{k+1} = \underline{\boldsymbol{\omega}}_k + \underline{\boldsymbol{w}}_k^w \,,$$

where $\underline{\boldsymbol{w}}_k^w$ is the process noise that affects the angular velocity. The process noise has zero mean and is Gaussian distributed with covariance $\mathbf{Q}^w$.

By combining (2) and (3), the system model for the pose tracking scenario can be written as

$$\underline{\boldsymbol{x}}_{k+1} = \begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \Lambda(\underline{\boldsymbol{r}}_k) \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \cdot \underline{\boldsymbol{x}}_k + \underline{\boldsymbol{w}}_k \,, \tag{4}$$

where the covariance of the process noise $\underline{\boldsymbol{w}}_k$ comprises the covariances of the process noises from the translation model $\mathbf{Q}^t$ and the rotation model $\mathbf{Q}^w$ according to

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}^t & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{Q}^w \end{bmatrix} .$$

It is worth mentioning that the length of the rotation vector has to be limited to the interval $[-\pi, \pi]$. Thus, if the norm of the rotation vector $||\underline{\boldsymbol{r}}_k||$ exceeds π, the transformation $\underline{\boldsymbol{r}}_k^{\text{new}} = \underline{\boldsymbol{r}}_k \cdot \left(1 - {}^{2\pi}/_{||\underline{\boldsymbol{r}}_k||}\right)$ has to be applied to the rotation vector.[1]

2.3 Recursive Gaussian State Estimation

Generally, in state estimation, filtering and prediction are performed alternating in a recursive fashion. The two steps depend on the measurement equation (1)

1 In the following, the time index k is omitted.

and the system equation (4), respectively. Furthermore, it is assumed that the state $\underline{\boldsymbol{x}}$ is Gaussian and thus, can be described completely by means of a mean vector $\underline{\mu}^x$ and a covariance matrix $\mathbf{C}^x$ according to

$$\underline{\mu}^x = \begin{bmatrix} \underline{\mu}^t \\ \underline{\mu}^{\dot{t}} \\ \underline{\mu}^r \\ \underline{\mu}^\omega \end{bmatrix} , \quad \mathbf{C}^x = \begin{bmatrix} \mathbf{C}^t & \mathbf{C}^{t,\dot{t}} & \mathbf{C}^{t,r} & \mathbf{C}^{t,\omega} \\ \mathbf{C}^{\dot{t},t} & \mathbf{C}^{\dot{t}} & \mathbf{C}^{\dot{t},r} & \mathbf{C}^{\dot{t},\omega} \\ \mathbf{C}^{r,t} & \mathbf{C}^{r,\dot{t}} & \mathbf{C}^r & \mathbf{C}^{r,\omega} \\ \mathbf{C}^{\omega,t} & \mathbf{C}^{\omega,\dot{t}} & \mathbf{C}^{\omega,r} & \mathbf{C}^\omega \end{bmatrix} .$$

Prediction Step

In the prediction step, the estimated mean $\underline{\mu}^{x,e}$ and covariance $\mathbf{C}^{x,e}$ of the previous filter step as well as the probabilistic model according to the system equation (4) are used in order to determine the predicted mean $\underline{\mu}^{x,p}$ and covariance $\mathbf{C}^{x,p}$. Due to the fact that the system model (4) is conditionally linear, i.e., if the rotation vector is set to a fixed value, the system model becomes linear and the prediction for each value can be performed by using the well-known Kalman predictor equation. Based on the predicted quantities for each fixed value, the predicted mean and covariance are calculated [3].

Filter Step

In the filter step, the state is updated based on the actual measurement value $\hat{\underline{z}}$ by using Bayesian inference. Due to the nonlinear measurement equation (1), the filter step cannot be solved in closed form. However, by assuming that the joint density of the measurement and the state is Gaussian, the estimated mean $\underline{\mu}^{x,e}$ and covariance $\mathbf{C}^{x,e}$ can be efficiently calculated by using

$$\begin{aligned} \underline{\mu}^{x,e} &= \underline{\mu}^{x,p} + \mathbf{C}^{x,d} \cdot \left(\mathbf{C}^d\right)^{-1} \left(\hat{\underline{d}} - \underline{\mu}^d\right) , \\ \mathbf{C}^{x,e} &= \mathbf{C}^{x,p} - \mathbf{C}^{x,d} \cdot \left(\mathbf{C}^d\right)^{-1} \cdot \mathbf{C}^{d,x} , \end{aligned} \tag{5}$$

where, $\mathbf{C}^d$ and $\underline{\mu}^d$ are the covariance and the mean of the range measurement $\underline{\boldsymbol{d}}$, respectively, and $\mathbf{C}^{x,d}$ is the cross-covariance between the state and the measurement. These three quantities depend on the considered nonlinear measurement equation (1) as well as on the involved densities of the state $\underline{\boldsymbol{x}}$ and the noise $\underline{\boldsymbol{v}}$. In general, these quantities cannot be calculated in closed form. However, some classes of nonlinear functions lead to analytic expressions [10].

3 Semi-Analytic Linearization

In the following, first the measurement equation is modified to a polynomial function and then the density of the state is decomposed in order to solve parts of the problem in closed form.[2]

3.1 Modified Measurement Equation

For an analytical solution of the filter step in closed from, the measurement equation (1) is squared, which results in

$$\left(\boldsymbol{d}_{i,j}\right)^2 = \left(\underline{g}_{i,j}(\underline{\boldsymbol{r}}) - \underline{\boldsymbol{t}} - \underline{\boldsymbol{v}}_{i,j}\right)^{\mathrm{T}} \cdot \left(\underline{g}_{i,j}(\underline{\boldsymbol{r}}) - \underline{\boldsymbol{t}} - \underline{\boldsymbol{v}}_{i,j}\right) , \tag{6}$$

with

$$\underline{g}_{i,j}(\underline{\boldsymbol{r}}) \triangleq \underline{L}_j - \mathbf{D}(\underline{\boldsymbol{r}}) \cdot \underline{M}_i \ . \tag{7}$$

Furthermore, it is assumed that squared ranges $\boldsymbol{z}_{i,j} \triangleq \left(\boldsymbol{d}_{i,j}\right)^2$ are measured, i.e., if a measurement $\hat{d}_{i,j}$ is taken, this measurement is mapped to the new measurement $\hat{z}_{i,j}$ by means of

$$\hat{z}_{i,j} = \left(\hat{d}_{i,j}\right)^2 \ . \tag{8}$$

For every possible range measurement between landmarks in global and target coordinate system, the nonlinear measurement equation can be compactly written as

$$\begin{aligned} \underline{\boldsymbol{z}} &= \underline{h}\left(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}, \underline{g}(\underline{\boldsymbol{r}})\right) \\ &= \mathbf{O}^{\mathrm{T}} \cdot \left(\left(\underline{g}\left(\underline{\boldsymbol{r}}\right) - \underline{1}_{N \cdot M} \otimes \underline{\boldsymbol{t}} - \underline{\boldsymbol{v}}\right) \odot \left(\underline{g}\left(\underline{\boldsymbol{r}}\right) - \underline{1}_{N \cdot M} \otimes \underline{\boldsymbol{t}} - \underline{\boldsymbol{v}}\right)\right) , \end{aligned} \tag{9}$$

where $\otimes$ is the Kronecker product, $\odot$ the element-wise product, $\underline{1}$ is the one-vector, D is the dimension of $\underline{\boldsymbol{t}}$, and $\mathbf{O} = \mathbf{I}_{N \cdot M} \otimes \underline{1}_D$. The vector $\underline{g}(\cdot)$ in (9) comprises all possible combinations of (7) for $i = 1, \ldots, N$ and $j = 1, \ldots, M$, where N is the number of landmarks of the target frame and M is the number of landmarks of the global coordinate system. Accordingly, $\underline{g}(\cdot)$ is given by

$$\underline{g}(\underline{\boldsymbol{r}}) = \left[\underline{g}_{1,1}(\underline{\boldsymbol{r}})^{\mathrm{T}} \quad \cdots \quad \underline{g}_{N,1}(\underline{\boldsymbol{r}})^{\mathrm{T}} \quad \cdots \quad \underline{g}_{N,M}(\underline{\boldsymbol{r}})^{\mathrm{T}}\right]^{\mathrm{T}} . \tag{10}$$

2 In the following, the indices for the predicted and estimated state are omitted.

The term $\underline{\boldsymbol{v}}$ in (9) is the Gaussian measurement noise with zero mean and covariance matrix

$$\mathbf{C}^{v} = \begin{bmatrix} \mathbf{C}^{v}_{1} & \cdots & \mathbf{C}^{v}_{1,j} & \cdots & \mathbf{C}^{v}_{1,N\cdot M} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{C}^{v}_{i,1} & \cdots & \mathbf{C}^{v}_{i,j} & \cdots & \mathbf{C}^{v}_{i,N\cdot M} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{C}^{v}_{N\cdot M,1} & \cdots & \mathbf{C}^{v}_{N\cdot M,j} & \cdots & \mathbf{C}^{v}_{N\cdot M} \end{bmatrix} ,$$

where the submatrix $\mathbf{C}^{v}_{i,j}$ describes the noise covariance between the landmarks from the different coordinate systems.

Based on the squared range measurements and the combined measurement equation (9), the unknown quantities $\underline{\mu}^{d}$, $\mathbf{C}^{d}$, and $\mathbf{C}^{x,d}$ in (5) can be replaced by

$$\underline{\mu}^{z} = \mathrm{E}\left\{\underline{h}\left(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}, \underline{g}(\underline{\boldsymbol{r}})\right)\right\} , \tag{11}$$

$$\mathbf{C}^{z} = \mathrm{E}\left\{\left(\underline{h}\left(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}, \underline{g}(\underline{\boldsymbol{r}})\right) - \underline{\mu}^{z}\right)\cdot\left(\underline{h}\left(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}, \underline{g}(\underline{\boldsymbol{r}})\right) - \underline{\mu}^{z}\right)^{\mathrm{T}}\right\} , \tag{12}$$

$$\mathbf{C}^{x,z} = \mathrm{E}\left\{\left(\underline{\boldsymbol{x}} - \underline{\mu}^{x,p}\right)\cdot\left(\underline{h}\left(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}, \underline{g}(\underline{\boldsymbol{r}})\right) - \underline{\mu}^{z}\right)^{\mathrm{T}}\right\} , \tag{13}$$

respectively.

3.2 Decomposition

The quantities in (11)–(13), however, still cannot be calculated in closed form. To facilitate this, it is necessary to decompose the problem into parts, which can be solved analytically. For this purpose the density of the state $f(\underline{x})$ has to be written as

$$f(\underline{x}) = f(\underline{t}, \underline{\dot{t}}, \underline{w}|\underline{r})\cdot f(\underline{r}) . \tag{14}$$

Furthermore, the density of the rotation $f(\underline{r})$ is approximated by means of a sample-based representation according to

$$f(\underline{r}) \approx \sum_{u=1}^{L} w_u \cdot \delta\left(\underline{r} - \underline{\mu}_u\right) , \tag{15}$$

where L is the number of sample points, $\underline{\mu}_u$ are the sample positions, w_u are the sample weights, and $\delta(\cdot)$ the Dirac delta distribution. For determining the sample points $\underline{\mu}_u$, several sampling schemes such as the unscented transform [8], Gauss-Hermite quadrature [7], or Gaussian shape approximation [6] can be applied.

Using (15) in (14), for a single sample point $\underline{\mu}_u$ the condition density $f(\underline{x}^a|\underline{r})$ with

$$\underline{x}^a \triangleq \begin{bmatrix} \underline{t}^{\mathrm{T}} & \underline{\dot{t}}^{\mathrm{T}} & \underline{w}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} ,$$

can be written as $f\left(\underline{x}^a|\underline{\mu}_u\right) = \mathcal{N}\left(\underline{x}^a; \underline{\mu}_u^a, \mathbf{C}^a\right)$ with mean vector and covariance matrix according to

$$\underline{\mu}_u^a = \begin{bmatrix} \underline{\mu}_u^t \\ \underline{\mu}_u^{\dot{t}} \\ \underline{\mu}_u^{\omega} \end{bmatrix} = \begin{bmatrix} \underline{\mu}^t \\ \underline{\mu}^{\dot{t}} \\ \underline{\mu}^{\omega} \end{bmatrix} + \begin{bmatrix} \mathbf{C}^{t,r} \\ \mathbf{C}^{\dot{t},r} \\ \mathbf{C}^{\omega,r} \end{bmatrix} \cdot \left(\mathbf{C}^r\right)^{-1} \cdot \left(\underline{\mu}_u - \underline{\mu}^r\right) ,$$

$$\mathbf{C}^a = \begin{bmatrix} \mathbf{C}_t^a & \mathbf{C}_{t,\dot{t}}^a & \mathbf{C}_{t,\omega}^a \\ \mathbf{C}_{\dot{t},t}^a & \mathbf{C}_{\dot{t}}^a & \mathbf{C}_{\dot{t},\omega}^a \\ \mathbf{C}_{\omega,t}^a & \mathbf{C}_{\omega,\dot{t}}^a & \mathbf{C}_{\omega}^a \end{bmatrix} = \begin{bmatrix} \mathbf{C}^t & \mathbf{C}^{t,\dot{t}} & \mathbf{C}^{t,\omega} \\ \mathbf{C}^{\dot{t},t} & \mathbf{C}^{\dot{t}} & \mathbf{C}^{\dot{t},\omega} \\ \mathbf{C}^{\omega,t} & \mathbf{C}^{\omega,\dot{t}} & \mathbf{C}^{\omega} \end{bmatrix} - \begin{bmatrix} \mathbf{C}^{t,r} \\ \mathbf{C}^{\dot{t},r} \\ \mathbf{C}^{\omega,r} \end{bmatrix} \left(\mathbf{C}^r\right)^{-1} \begin{bmatrix} \mathbf{C}^{t,r} \\ \mathbf{C}^{\dot{t},r} \\ \mathbf{C}^{\omega,r} \end{bmatrix}^{\mathrm{T}} . \tag{16}$$

3.3 Mean

In order to calculate the required mean $\underline{\mu}^z$ in (11), (14) and (15) are used to obtain

$$\underline{\mu}^z = \mathrm{E}\left\{\underline{h}\left(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}, \underline{g}(\underline{\boldsymbol{r}})\right)\right\} \approx \sum_{u=1}^{L} w_u \iint \underline{h}\left(\underline{t}, \underline{v}, \underline{g}(\underline{r})\right) \cdot f\left(\underline{x}^a|\underline{\mu}_u\right) \cdot \delta\left(\underline{r} - \underline{\mu}_u\right) \cdot f(\underline{v})\, \mathrm{d}\underline{v}\, \mathrm{d}\underline{x} .$$

Thanks to the sifting property of the Dirac delta distribution, the integration w.r.t. the rotation variable $\underline{r}$ can be solved. As the nonlinear function in (9) then only depends on the translation, the variable comprising the velocities (translational and angular) can be simply marginalized. The predicted mean is then given by

$$\begin{aligned} \underline{\mu}^z &\approx \sum_{u=1}^{L} w_u \iint \underline{h}\left(\underline{t}, \underline{v}, \underline{g}\left(\underline{\mu}_u\right)\right) \cdot f\left(\underline{x}^a|\underline{\mu}_u\right) \cdot f(\underline{v})\ \mathrm{d}\underline{v}\, \mathrm{d}\underline{x}^a \\ &= \sum_{u=1}^{L} w_u \underbrace{\iint \underline{h}\left(\underline{t}, \underline{v}, \underline{g}\left(\underline{\mu}_u\right)\right) \cdot f\left(\underline{t}|\underline{\mu}_u\right) \cdot f(\underline{v})\ \mathrm{d}\underline{v}\, \mathrm{d}\underline{t}}_{=\mathrm{E}\left\{\underline{h}\left(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}, \underline{g}\left(\underline{\mu}_u\right)\right)\right\}} . \end{aligned} \tag{17}$$

To resolve the remaining integrals, a new variable $\underline{\boldsymbol{q}}_u$ is introduced, which is defined by

$$\underline{\boldsymbol{q}}_u \triangleq \underline{g}(\underline{\mu}_u) - \underline{1}_{N \cdot M} \otimes \underline{\boldsymbol{t}} - \underline{\boldsymbol{v}} \, .$$

Due to the linear relation, the variable $\underline{\boldsymbol{q}}_u$ is Gaussian distributed with mean and covariance

$$\begin{aligned} \underline{\mu}_u^q &= \underline{g}\left(\underline{\mu}_u\right) - \underline{1}_{N \cdot M} \otimes \underline{\mu}_u^t \, , \\ \mathbf{C}^q &= \mathbf{C}^v + \left(\underline{1}_{N \cdot M} \cdot \left(\underline{1}_{N \cdot M}\right)^{\mathrm{T}}\right) \otimes \mathbf{C}_t^a \, , \end{aligned} \tag{18}$$

respectively, where $\underline{\mu}_u^t$ and $\mathbf{C}_t^a$ are the quantities from $\underline{\mu}_u^a$ and $\mathbf{C}^a$ that are related to the translation (16). Based on this new random variable $\underline{\boldsymbol{q}}_u$, the expected value in (17) for a fixed value for $\underline{\mu}_u$ is calculated as

$$\mathrm{E}\left\{\underline{h}\left(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}, \underline{g}\left(\underline{\mu}_u\right)\right)\right\} \overset{(9)}{=} \mathrm{E}\left\{\mathbf{O}^{\mathrm{T}} \cdot \left(\underline{\boldsymbol{q}}_u \odot \underline{\boldsymbol{q}}_u\right)\right\} = \mathbf{O}^{\mathrm{T}} \cdot \left(\underline{\mu}_u^q \odot \underline{\mu}_u^q + \mathrm{diag}\left(\mathbf{C}^q\right)\right) \, . \tag{19}$$

Summing up this result of the expected value for every fixed value $\underline{\mu}_u$, $u = 1, \ldots, L$, gives the desired mean $\underline{\mu}^z$ according to

$$\underline{\mu}^z = \sum_{u=1}^{L} w_u \cdot \mathbf{O}^{\mathrm{T}} \cdot \left(\underline{\mu}_u^q \odot \underline{\mu}_u^q + \mathrm{diag}\left(\mathbf{C}^q\right)\right) \, . \tag{20}$$

3.4 Covariance

Similar to (17), the covariance $\mathbf{C}^z$ in (12) is given by

$$\mathbf{C}^z = \sum_{u=1}^{L} w_u \cdot \mathrm{E}\left\{\left(\underline{h}\left(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}, \underline{g}\left(\underline{\mu}_u\right)\right) - \underline{\mu}^z\right) \cdot \left(\underline{h}\left(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}, \underline{g}\left(\underline{\mu}_u\right)\right) - \underline{\mu}^z\right)^{\mathrm{T}}\right\} \, , \tag{21}$$

where the expected value can be resolved to

$$\begin{aligned} \mathrm{E}\left\{\left(\underline{h}(\cdot) - \underline{\mu}^z\right) \cdot \left(\underline{h}(\cdot) - \underline{\mu}^z\right)^{\mathrm{T}}\right\} &= \mathrm{E}\left\{\underline{h}(\cdot) \cdot \underline{h}(\cdot)^{\mathrm{T}}\right\} - \underline{\mu}^z \cdot \left(\underline{\mu}^z\right)^{\mathrm{T}} \\ &\overset{(9)}{=} \mathbf{O}^{\mathrm{T}} \cdot \underbrace{\mathrm{E}\left\{\left(\underline{\boldsymbol{q}}_u \odot \underline{\boldsymbol{q}}_u\right) \cdot \left(\underline{\boldsymbol{q}}_u^{\mathrm{T}} \odot \underline{\boldsymbol{q}}_u^{\mathrm{T}}\right)\right\}}_{= \mathrm{E}\left\{\left(\underline{\boldsymbol{q}}_u \cdot \underline{\boldsymbol{q}}_u^{\mathrm{T}}\right) \odot \left(\underline{\boldsymbol{q}}_u \cdot \underline{\boldsymbol{q}}_u^{\mathrm{T}}\right)\right\}} \cdot \mathbf{O} - \underline{\mu}^z \cdot \left(\underline{\mu}^z\right)^{\mathrm{T}} \, , \end{aligned} \tag{22}$$

where the Cartesian product of the squared variable $\underline{\boldsymbol{q}}_u$ is written as the squared Cartesian product of $\underline{\boldsymbol{q}}_u$.

The expected value of the matrix $\mathrm{E}\left\{\left(\underline{\boldsymbol{q}}_u \cdot \underline{\boldsymbol{q}}_u^{\mathrm{T}}\right) \odot \left(\underline{\boldsymbol{q}}_u \cdot \underline{\boldsymbol{q}}_u^{\mathrm{T}}\right)\right\}$ can be calculated separately. An entry of the matrix at row i and column j corresponds to the value $\mathrm{E}\left\{\boldsymbol{q}_{u,i}^2 \cdot \boldsymbol{q}_{u,j}^2\right\}$, which is a fourth-order non-central moment given by

$$\mathrm{E}\left\{\boldsymbol{q}_{u,i}^2 \cdot \boldsymbol{q}_{u,j}^2\right\} = \left(\left(\mu_{u,i}^q\right)^2 + C_{i,i}^q\right) \cdot \left(\left(\mu_{u,j}^q\right)^2 + C_{j,j}^q\right) + 4 \cdot \mu_{u,i}^q \cdot \mu_{u,j}^q \cdot C_{i,j}^q + 2 \cdot \left(C_{i,j}^q\right)^2 ,$$

where $\mu_{u,i}^q$ is the ith entry of the vector $\underline{\mu}_u^q$ and $C_{i,j}^q$ is the entry at row i, column j of the matrix $\mathbf{C}^q$ for $i = 1, \ldots, N \cdot M \cdot D$ and $j = 1, \ldots, N \cdot M \cdot D$. Accordingly, the right-hand expected value in (22) can be expressed analytically via

$$\begin{aligned} \mathrm{E}\left\{\underline{h}(\cdot) \cdot \underline{h}(\cdot)^{\mathrm{T}}\right\} = {} & \mathrm{E}\left\{\underline{h}(\cdot)\right\} \cdot \mathrm{E}\left\{\underline{h}(\cdot)\right\}^{\mathrm{T}} + \\ & \mathbf{O}^{\mathrm{T}} \cdot \left(4 \cdot \left(\underline{\mu}_u^q \cdot \left(\underline{\mu}_u^q\right)^{\mathrm{T}}\right) \odot \mathbf{C}^q + 2 \cdot \mathbf{C}^q \odot \mathbf{C}^q\right) \cdot \mathbf{O} , \end{aligned} \tag{23}$$

where the expected value $\mathrm{E}\left\{\underline{h}(\cdot)\right\}$ is given by (19).

3.5 Cross-Covariance

By again exploiting the similarity to (17), the cross-covariance $\mathbf{C}^{x,z}$ is

$$\mathbf{C}^{x,z} = \sum_{u=1}^{L} w_u \cdot \begin{bmatrix} \mathrm{E}\left\{\left(\underline{\boldsymbol{t}} - \underline{\mu}^t\right) \cdot \left(\underline{h}\left(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}, \underline{g}\left(\underline{\mu}_u\right)\right) - \underline{\mu}^z\right)^{\mathrm{T}}\right\} \\ \mathrm{E}\left\{\left(\underline{\dot{\boldsymbol{t}}} - \underline{\mu}^{\dot{t}}\right) \cdot \left(\underline{h}\left(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}, \underline{g}\left(\underline{\mu}_u\right)\right) - \underline{\mu}^z\right)^{\mathrm{T}}\right\} \\ \mathrm{E}\left\{\left(\underline{\mu}_u - \underline{\mu}^r\right) \cdot \left(\underline{h}\left(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}, \underline{g}\left(\underline{\mu}_u\right)\right) - \underline{\mu}^z\right)^{\mathrm{T}}\right\} \\ \mathrm{E}\left\{\left(\underline{\boldsymbol{\omega}} - \underline{\mu}^{\omega}\right) \cdot \left(\underline{h}\left(\underline{\boldsymbol{t}}, \underline{\boldsymbol{v}}, \underline{g}\left(\underline{\mu}_u\right)\right) - \underline{\mu}^z\right)^{\mathrm{T}}\right\} \end{bmatrix} . \tag{24}$$

with its respective sub-cross-covariance matrices. Expanding each product in (24) leads to the following not yet evaluated expected values

$$\begin{aligned} \mathrm{E}\left\{\underline{\boldsymbol{t}} \cdot \underline{h}(\cdot)^{\mathrm{T}}\right\} &= -2 \cdot \mathbf{C}_t^a \cdot \mathbf{S}_u + \underline{\mu}_u^t \cdot \mathrm{E}\left\{\underline{h}(\cdot)\right\}^{\mathrm{T}} , \\ \mathrm{E}\left\{\underline{\dot{\boldsymbol{t}}} \cdot \underline{h}(\cdot)^{\mathrm{T}}\right\} &= -2 \cdot \mathbf{C}_{\dot{t},t}^a \cdot \mathbf{S}_u + \underline{\mu}_u^{\dot{t}} \cdot \mathrm{E}\left\{\underline{h}(\cdot)\right\}^{\mathrm{T}} , \\ \mathrm{E}\left\{\underline{\boldsymbol{\omega}} \cdot \underline{h}(\cdot)^{\mathrm{T}}\right\} &= -2 \cdot \mathbf{C}_{\omega,t}^a \cdot \mathbf{S}_u + \underline{\mu}_u^{w} \cdot \mathrm{E}\left\{\underline{h}(\cdot)\right\}^{\mathrm{T}} , \end{aligned} \tag{25}$$

with $\mathbf{S}_u \triangleq \left[\underline{g}_{1,1}\left(\underline{\mu}_u\right) \quad \cdots \quad \underline{g}_{N,1}\left(\underline{\mu}_u\right) \cdots \quad \underline{g}_{N,M}\left(\underline{\mu}_u\right)\right] - \underline{1}_{N\cdot M}^{\mathrm{T}} \otimes \underline{\mu}_u^t$. The quantities $\underline{\mu}_u^{\dot{t}}, \mathbf{C}_{\dot{t},t}^a, \underline{\mu}_u^w$, and $\mathbf{C}_{\omega,t}^a$ are those entries of $\underline{\mu}^a$ and $\mathbf{C}^a$ that are related to the translation and angular velocity, respectively (see (16)).

3.6 Summary

In the following, all steps of the filter are summarized:

1. Determine the sample points $\underline{\mu}_u$ with corresponding weights w_u of the density of the rotation $f(\underline{r})$, where $u = 1,\ldots,L$.
2. Calculate the conditional means $\underline{\mu}_u^t$, $\underline{\mu}_u^{\dot{t}}$, and $\underline{\mu}_u^w$, the conditional covariance $\mathbf{C}_t^a$, and conditional cross-covariances $\mathbf{C}_{\dot{t},t}^a$ and $\mathbf{C}_{\omega,t}^a$ for all sample points (see (16)).
3. Calculate the mean $\underline{\mu}_u^q$ and covariance $\mathbf{C}^q$ of the random variable $\underline{\boldsymbol{q}}_u$ for all sample points (see (18)).
4. Calculate the expected values $\mathrm{E}\{\underline{h}(\cdot)\}$ (see (19)) and $\mathrm{E}\left\{\underline{h}(\cdot)\cdot\underline{h}(\cdot)^{\mathrm{T}}\right\}$ (see (23)) as well as $\mathrm{E}\left\{\underline{\boldsymbol{t}}\cdot\underline{h}(\cdot)^{\mathrm{T}}\right\}$, $\mathrm{E}\left\{\underline{\dot{\boldsymbol{t}}}\cdot\underline{h}(\cdot)^{\mathrm{T}}\right\}$, and $\mathrm{E}\left\{\underline{\boldsymbol{\omega}}\cdot\underline{h}(\cdot)^{\mathrm{T}}\right\}$ (see (25)) for all sample points.
5. Combine the results of the expected values for determining the mean $\underline{\mu}^z$ (see (20)), the covariance $\mathbf{C}^z$ (see (21)), and the cross-covariance $\mathbf{C}^{x,z}$ (see (24)).
6. Square the measured ranges $\underline{\hat{d}}$ (see (8)).
7. Perform the filter step in order to calculate the estimated mean and covariance (see (5)).

3.7 Computational Complexity

The determination of the sample points has the highest computational cost of calculating the required mean $\underline{\mu}^z$, covariance $\mathbf{C}^z$, and cross-covariance $\mathbf{C}^{x,z}$. For the proposed approach, the sample points only have to be calculated for the rotation. In this case, the computational complexity is in $\mathcal{O}\left(R^3 + D^3 \cdot N^2 \cdot M^2\right)$, where R is the dimension of the rotation vector. For comparison, if the decomposition would be based on separating the directly and the indirectly observed parts of the state—see [3] for details—the computational complexity would be in

$\mathcal{O}\big((R+D+N\cdot M)^3\big)$. This significantly higher complexity is caused by the need to additionally sample the translation and the noise, as both are nonlinearly transformed.

4 Simulation Results

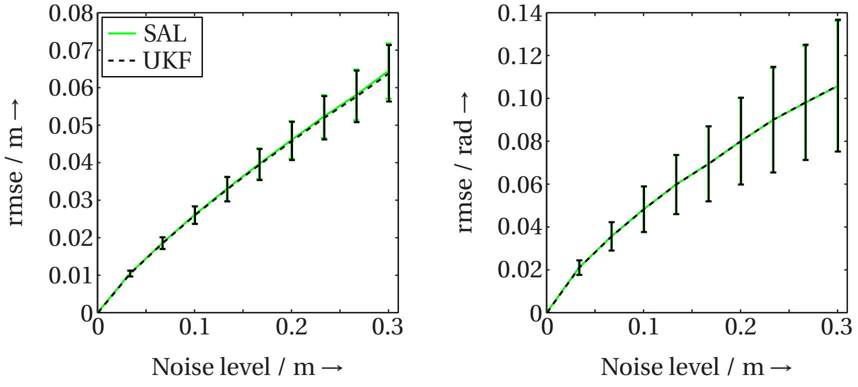
In the simulation, a two-dimensional coordinate system is considered containing four emitters and four sensors with positions

$$\begin{bmatrix} \underline{M}_1^{\mathrm{T}} \\ \underline{M}_2^{\mathrm{T}} \\ \underline{M}_3^{\mathrm{T}} \\ \underline{M}_4^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} -0.2 & -0.2 \\ -0.2 & 0.2 \\ 0.2 & -0.2 \\ 0.2 & 0.2 \end{bmatrix} \mathrm{m}\,, \quad \begin{bmatrix} \underline{L}_1^{\mathrm{T}} \\ \underline{L}_2^{\mathrm{T}} \\ \underline{L}_3^{\mathrm{T}} \\ \underline{L}_4^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} -2 & -2 \\ -2 & 2 \\ 2 & -2 \\ 2 & 2 \end{bmatrix} \mathrm{m}$$

with respect to the global coordinate system and the target frame, respectively. At different noise levels ranging from $[0.000001,\ldots,0.3]$ meters, 1000 random trajectories are generated, where the sampling time was $T = 0.1\,\mathrm{s}$. The noise process is assumed as isotropic. The measured ranges were generated with (1).

In the simulation, the proposed approach denoted by *Semi-Analytic Linearization* (SAL) is compared to a standard estimator. As a standard estimator, the unscented Kalman filter (UKF) is used, with decomposition into directly and indirectly observed parts as explained above. In this case, the density of the translation and the rotation has to be approximated with samples. Furthermore, due to the fact that the measurement noise is mapped through the nonlinear transformation, it has to be approximated with samples as well. For the UKF, 71 sample points are used to approximate the density of the translation, rotation, and the measurement noise. On the other hand, the proposed approach only has to approximate the rotation by sample points, where for determining the sample points the approach presented in [6] is used. Here, 5 sample points are used to approximate the density for the rotation.

The system equation (4) for a two-dimensional coordinate system becomes linear and thus, the prediction step can be solved by using the Kalman prediction step. This is exploited for both estimators. The covariance of the continuous process noise for the velocities (translation and rotation) is set to $\mathbf{Q}_c^t = \mathrm{diag}\big(\begin{bmatrix}0.1 & 0.1\end{bmatrix}\big)$ and $Q_c^w = 0.1$, respectively. The initial covariance of the translation is $\mathbf{C}_0^t = \mathrm{diag}\big(\begin{bmatrix}10 & 10\end{bmatrix}\big)$, of the translation velocity $\mathbf{C}_0^{\dot{t}} = \mathrm{diag}\big(\begin{bmatrix}10 & 10\end{bmatrix}\big)$, of the angle



(a) Rmse of the translation. **(b)** Rmse of the rotation.

Figure 2: Simulation results for the two estimators SAL and UKF. The average and the standard deviation of the rmse for different noise levels.

$C_0^r = 0.001$, and of the angular velocity $C_0^w = 0.0001$. The initial mean is initialized with zeros.

In Figure 2, the average and the standard deviation of the root mean square error (rmse) over 1000 trajectories for each noise level are plotted. The performance of the two estimators is nearly identical. Regarding the computational effort, the proposed approach only has to determine sample points for one dimension, which can be implemented very efficiently. On the other hand, the UKF calculates a matrix root of the covariance of the noise and the state (translation and rotation), which has a high computational effort considering the size of the combined covariance matrix, which is 35×35. In the simulation, the proposed approach is *three times faster* than the standard approach.

5 Conclusions

In this paper, a state estimator for range-based pose tracking is presented, which relies on an intelligent decomposition of the proposed problem. This state estimator exploits the Gaussian assumption and makes use of a modified measurement equation, where the measurement noise process is inside of the nonlinearity. Due to the modified measurement equation, the filter step is separated into an analytically integrable and an approximate part. In the approximate part, the density of the rotation is represented by samples. For every sample point, the required moments for the filter step are then calculated by analytic moment calculation. In doing so, the computational demand for the approximation is drastically reduced compared to a standard decomposition, which relies on conditionally linear substructures.

The new approach was evaluated in a two-dimensional simulation and compared to the standard approach. Regarding the rmse, the performance of the two estimators is similar. However, in the two-dimensional simulation example, only a one-dimensional density of the rotation has to be approximated for the proposed approach, which is feasible for an embedded system, compared to the standard estimator, where the matrix root of a large covariance matrix has to be calculated. In summary, if the decomposition in integrable substructure is exploited, the number of sample points and thus, the computational complexity can be drastically reduced.

References

[1] Nicholas Ayache. *Artifical Vision for Mobile Robots: Stereo Vision and Multisensory Perception.* Massachusetts Institute of Technology, 1991.

[2] Frederik Beutler and Uwe D. Hanebeck. Closed-Form Range-Based Posture Estimation Based on Decoupling Translation and Orientation. In *Proceedings of the 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2005)*, volume 4, pages 989–992, Philadelphia, Pennsylvania, March 2005.

[3] Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck. Gaussian Filtering using State Decomposition Methods. In *Proceedings of the 12th International Conference on Information Fusion (Fusion)*, pages 579–586, Seattle, Washington, July 2009.

[4] Frederik Beutler, Marco F. Huber, and Uwe D. Hanebeck. Instantaneous Pose Estimation using Rotation Vectors. In *Proceedings of the 34th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3413–3416, Taipei, Taiwan, April 2009.

[5] John E. Bortz. A New Mathematical Formulation for Strapdown Inertial Navigation. *IEEE Transactions on Aerospace and Electronic Systems*, AES-7(1):61–66, January 1971.

[6] Marco F. Huber and Uwe D. Hanebeck. Gaussian Filter based on Deterministic Sampling for High Quality Nonlinear Estimation. In *Proceedings of the 17th IFAC World Congress*, Seoul, Republic of Korea, July 2008.

[7] Kazufumi Ito and Kaiqi Xiong. Gaussian Filters for Nonlinear Filtering Problems. *IEEE Transactions on Automatic Control*, 45(5):910–927, May 2000.

[8] Simon J. Julier and Jeffrey K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[9] Mark R. Morelande and Bill Moran. An Unscented Transformation for Conditionally Linear Models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages III–1417–III–1420, April 2007.

[10] Athanasios Papoulis and S. Unnikrishna Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill Science/Engineering/Math, 4th edition, 2002.

[11] Patrick Rößler, Frederik Beutler, Uwe D. Hanebeck, and Norbert Nitzsche. Motion Compression Applied to Guidance of a Mobile Teleoperator. In *Proceedings of the 2005 IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2495–2500, 2005.

[12] Greg Welch, B. Danette Allen, Adrian Ilie, and Gary Bishop. Measurement Sample Time Optimization for Human Motion Tracking/Capture Systems. In *Proceedings of Trends and Issues in Tracking for Virtual Environments, Workshop at the IEEE Virtual Reality 2007 Conference*, 2007.

Paper N

On-line Dispersion Source Estimation using Adaptive Gaussian Mixture Filter

Authors: Marco F. Huber

Edited version of the paper:

M. F. Huber. On-line Dispersion Source Estimation using Adaptive Gaussian Mixture Filter. In *Proceedings of the 19th IFAC World Congress*, pages 1059–1066, Cape Town, South Africa, August 2014.

On-line Dispersion Source Estimation using Adaptive Gaussian Mixture Filter

Marco F. Huber

AGT International
Darmstadt, Germany
marco.huber@ieee.org

Abstract

The reconstruction of environmental events has gained increased interest in the recent years. In this paper, the focus is on estimating the location and strength of a gas release from distributed measurements. The estimation is formulated as Bayesian inverse problem, which utilizes a Gaussian plume forward model. A novel recursive estimation algorithm based on statistical linearization and Gaussian mixture densities with adaptive component number selection is used in order to allow at the same time accurate and computationally efficient source estimation. The proposed solution is compared against state-of-the-art methods via simulations and a real-word experiment.

1 Introduction

If a hazardous gas has been released—either accidentally or deliberately—into atmosphere, it is of paramount importance to gain knowledge of this event at an early stage in order to increase the effectiveness of counter measures for protecting the public and for mitigating the harmful effects. By means of so-called *atmospheric dispersion models* (ADMs), it is possible to predict the concentration spread of the released gas in space and time. These models, however, merely provide reliable predictions, if the characteristics of the gas source are known precisely. To determine or estimate the source characteristics it necessary to solve an *inverse problem*, where one has to infer the location and strength of the gas release from concentration measurements of spatially distributed sensors.

In general, solution methods of the source estimation problem can be classified into *forward* and *backward* methods [15]. Forward methods employ an forward-running ADM multiple times in order to find an estimate of the source that best describes the given concentration measurements. Here, the mostly used techniques are based on Bayesian inference in combination with Monte Carlo sampling. Sequential Monte Carlo methods as described in [21] or [23] employ a set of samples or particles that forms the posterior probability distribution of the source parameters. This distribution is updated by means of Bayes' rule whenever new concentration measurements from sensors are available. In contrast to this online procedure, Markov chain Monte Carlo (MCMC) methods process all acquired concentration measurements in a batch in order to determine the posterior distribution. For this purpose, samples are drawn from the posterior distribution by simulating a Markov chain that has the desired posterior distribution as its stationary distribution. Given a properly constructed Markov chain it can be shown that MCMC reaches the stationary distribution after a typically large number of sampling steps. Application of MCMC to source estimation can be found for instance in [4, 19], and [8].

Backward methods instead perform only one model run in the reverse direction from the sensors to the source. Commonly used techniques are backtracking, where an inverse version of an ADM is utilized (see e.g. [10]), and variational methods, where a cost function between model predictions and concentration measurements is optimized (see e.g. [16, 22]). The backward approach is preferred over forward methods, when the number of sources is larger than the number of sensors [15].

In this paper, a novel forward approach is proposed that aims at performing online source estimation, i.e., the current source estimate is updated at run-time whenever sensors provide new concentration measurements. For this purpose, a Gaussian plume dispersion model is employed, which allows predicting the gas dispersion in closed-form with low computational overhead. This forward model is employed in a recursive Bayesian inference framework to allow for uncertainties arising from modeling errors and sensor noise. The resulting statistical inverse problem, however, cannot be solved in closed form due to nonlinearities in the Gaussian plume model. To overcome this issue, the so-called *adaptive Gaussian mixture filter* (AGMF) proposed in [11] is employed. Here, the posterior distribution is approximated via a sum of Gaussians, which is known to be a universal function approximator [14]. To limit the computational demand but still perform accurate estimation, the number of Gaussian components is adapted at run-time depending on the nonlinearity of the dispersion model.

The paper is structured as follows: In the next section, a general ADM and its special case, the Gaussian plume model, are introduced. The recursive Bayesian estimation problem is stated in Section 3, while Section 4 describes the AGMF. A comparison of the proposed source estimation methods with state-of-the-art is provided in Section 5. The paper closes with a conclusion.

2 Problem Formulation

In this section, a physical model describing the spatial dispersion of a substance in atmosphere is derived. It models the transportation of a substance from an emitting source to regions of low concentration under consideration of various environmental conditions.

2.1 General Dispersion Model

In the following, $c(\underline{x}, t)$ is the concentration of the substance at position $\underline{x} = [x, y, z]^{\mathrm{T}} \in \mathbb{R}^3$ and at time $t \geq 0$. The concentration follows the *advection-diffusion equation*

$$\frac{\partial c(\underline{x}, t)}{\partial t} = \nabla \cdot \left(\mathbf{K} \nabla c(\underline{x}, t) - \underline{v} \cdot c(\underline{x}, t)\right) + s(\underline{x}, t) \tag{1}$$

with $\nabla \triangleq [\partial/\partial x, \partial/\partial y, \partial/\partial z]^{\mathrm{T}}$ (see e.g. [9]). The term $\mathbf{K}\nabla c(\underline{x}, t)$ describes the diffusion according to Fick's law with diffusion matrix $\mathbf{K}(\underline{x}, t)$ and the term $\underline{v} \cdot c(\underline{x}, t)$ represents linear advection due to wind with velocity $\underline{v}(\underline{x}, t)$. Finally, $s(\underline{x}, t)$ is a source or sink term.

Analytical solutions of (1), i.e., functions $c(\underline{x}, t)$ satisfying the equation, exist merely for some special cases. In the following, one important special case utilized throughout the paper is introduced.

2.2 Gaussian Plume

In order to obtain a closed-form solution, it is necessary to make several assumptions:

1. The substance is emitted at a constant *rate* $q > 0$ from a single point source at location $\underline{x}_s \triangleq [x_s, y_s, z_s]^{\mathrm{T}}$. Thus, the source term $s(\underline{x}, t)$ in (1) becomes
$$s(\underline{x}, t) = q \cdot \delta(x - x_s) \cdot \delta(y - y_s) \cdot \delta(z - z_s) \ ,$$
where $\delta(x - x_s)$ is the Dirac delta localized at x_s.
2. The wind is constant with velocity $v \geq 0$ and the wind direction draws an angle ϕ with the x-axis so that $\underline{v} = v \cdot [\cos\phi, \sin\phi, 0]^{\mathrm{T}}$.
3. The diffusion is a function of the downwind distance only. Furthermore, it is assumed that the diffusion in wind direction can be neglected.
4. The terrain is flat and the ground cannot be penetrated by the substance.
5. The solution is steady state, i.e., time independent.

Based on these assumptions and additional boundary conditions that force vanishing concentrations at infinite distance from the source and at upwind distances, (1) has the time-invariant solution

$$c(\underline{x}) = \frac{q}{2\pi \cdot v \cdot \sigma_y \sigma_z} \cdot \exp\Big(-\frac{(1+2\sin(\phi)\cos(\phi)) \cdot (y-y_s)^2}{2\sigma_y^2}\Big) \cdot \Big[\exp\Big(-\frac{(z-z_s)^2}{2\sigma_z^2}\Big) + \exp\Big(-\frac{(z+z_s)^2}{2\sigma_z^2}\Big)\Big] \ , \tag{2}$$

which is the well-known *Gaussian plume* dispersion model (for a detailed derivation see [22]). Here, σ_y and σ_z are the so-called standard deviations of the Gaussian concentration distribution. They are both functions of x, y, ϕ and they depend on the stability of the atmosphere.

The Gaussian plume model (2) is employed as it is widely used and suitable for describing short range substance releases. Furthermore, being an analytical model, it allows for an on-line and computationally light-weight estimation of the unknown parameters.

In this paper, the focus is on estimating the source rate q and location $\underline{x}_s$ from a set of spatially distributed concentration measurements. It is assumed that the measurements become available sequentially over time, i.e., batch or off-line estimation is impractical. Additional parameters like wind speed or direction are assumed to be known, as they can be provided reliably from external sources like weather stations. However, the approach proposed in this paper can be easily extended to estimate also these additional parameters.

3 Recursive Estimation

The Gaussian plume model forms an instance of a so-called *forward model*

$$\underline{z} = \underline{g}(\underline{\theta}) \;, \tag{3}$$

where the output or observations $\underline{z}$ are defined based on physical transformations $\underline{g}(.)$ and model parameters $\underline{\theta}$. In the considered problem, $\underline{z}$ corresponds to a set of concentration measurements, $\underline{g}$ is the Gaussian plume model (2), and $\underline{\theta}^{\mathrm{T}} \triangleq [q, \underline{x}_s^{\mathrm{T}}]$ comprises the source rate as well as the source location.

As the goal is to determine the parameters $\underline{\theta}$, an *inverse problem* of (3) needs to be considered, where $\underline{\theta}$ is estimated given the observed concentrations $\underline{z}$. Inverse problems are typically difficult to solve: they are often ill-conditioned, ambiguities exist—the observations $\underline{z}$ can be explained by different parameters $\underline{\theta}$—, and an inverse transformation $\underline{g}^{-1}$ often is not available.

3.1 Bayesian Estimation

For solving inverse problems, deterministic or probabilistic approaches can be applied. In this paper, a probabilistic approach employing *Bayesian inference* is considered. In doing so, uncertainties arising for instance from sensor noise or modeling errors can be incorporated.

According to Bayes' theorem (see e.g. [18]), the so-called *posterior density* $p(\underline{\theta}|\underline{z})$ of $\underline{\theta}$ is calculated according to

$$p(\underline{\theta}|\underline{z}) = \frac{p(\underline{z}|\underline{\theta}) \cdot p(\underline{\theta})}{p(\underline{z})} \;, \tag{4}$$

which is the conditional probability of the unknown model parameters given the measurements. This density function represents the solution of the inverse problem. In (4), $p(\underline{z}|\underline{\theta})$ is the *likelihood*, $p(\underline{\theta})$ is the *prior density*, and $p(\underline{z}) = \int p(\underline{z}|\underline{\theta}) \cdot p(\underline{\theta})\, \mathrm{d}\underline{\theta}$ is a normalization constant.

By inspecting (4) it can be seen that all concentration measurements are processed at once. Under weak assumptions however, Bayes' theorem also allows an recursive calculation of the posterior distribution. This is especially useful, when the concentration measurements z_k are acquired over time at discrete time steps t_k with $k = 1,2,\ldots$ and one is interested in constantly updating the posterior. Assuming that the concentration measurements z_k are conditionally independent given the model parameters, (4) can be re-formulated in a recursion

$$p(\underline{\theta}|z_{1:k}) = \frac{p(z_k|\underline{\theta}) \cdot p(\underline{\theta}|z_{1:k-1})}{p(z_k|z_{1:k-1})} \tag{5}$$

which commences from the prior $p(\underline{\theta})$ and where $z_{1:k} \triangleq (z_1, z_2, \ldots, z_k)$ is the collection of all measurements up to and including z_k.

3.2 Measurement Model

While the prior reflects the a priori knowledge of the user on the source, the likelihood $p(z_k|\underline{\theta})$ relates a concentration measurement to the unknown source parameters. In order to define the likelihood, one can make use of the Gaussian plume model derived above, assuming that this model represents the underlying dispersion mechanism. For this purpose, suppose that the k-th measurement z_k is acquired by a sensor at location $\underline{x}_r \triangleq [x_r, y_r, z_r]^{\mathrm{T}}$ at time t_k. The resulting measurement model is given by

$$z_k = c(\underline{x}_r; \underline{\theta}) + v_k \,, \quad v_k \sim p(v_k) \triangleq \mathcal{N}(0, \sigma_v^2) \,, \tag{6}$$

where $c(\underline{x}_r; \underline{\theta})$ is the true concentration value according to (2) and v_k is the sensor's noise, which is assumed to be zero-mean Gaussian with variance σ_v^2 and independent in time and space. The measurement model (6) can be turned into a statistical model according to

$$p(z_k|\underline{\theta}) = \int \underbrace{p(z_k|v_k, \underline{\theta})}_{\delta(z_k - c(\underline{x}_r; \underline{\theta}) - v_k)} \cdot p(v_k)\, \mathrm{d}v_k = \mathcal{N}\left(z_k; c(\underline{x}_r; \underline{\theta}), \sigma_v^2\right) \,,$$

where the second equality follows from the sifting property of the Dirac delta distribution $\delta(.)$. This completes the derivation of the Bayesian estimation formalism.

It is worth mentioning that in case of multiple sources, the concentration measurement can be written as

$$z_k = \sum_{i=1}^{N} c_i(\underline{x}_r; \underline{\theta}_i) + v_k \,, \tag{7}$$

where the superposition of the concentration values $c_i(\underline{x}_r; \underline{\theta}_i)$ of the sources $i = 1, \ldots, N$ is exploited (see [22]). The likelihood for multiple sources can be derived analogously as described above.

3.3 Approximate Estimation

Unfortunately, the Bayesian formalism in (5) is merely of limited practical use as a closed-form solution of the recursion is not available in general. Except for some special cases like the linear Gaussian one, an approximate solution is inevitable. This holds also for the considered source estimation problem due to the nonlinear Gaussian plume model (2) that forms the likelihood. Typical approximations of nonlinear Bayesian estimation problems rely on Monte Carlo methods like particle filters (see e.g. [3]). However, the obtained results are not reproducible and scaling for high dimensional parameters is an issue. Alternatives are Gaussian filters like the extended Kalman filter (EKF) or the unscented Kalman filter (UKF) (see e.g. [18]), where the posterior is approximated by means of a Gaussian distribution. These approaches scale well with high dimensions, but the approximation can be poor in case of strong nonlinearities. In this paper, an approximate Bayesian estimator named *adaptive Gaussian mixture filter* (AGMF) is employed that provides Gaussian mixture posteriors. As a result, significantly better approximations compared to Gaussian filters are obtained with an at the same time low computational demand and good scalability.

4 Adaptive Gaussian Mixture Filter

In this section, a brief introduction of the AGMF proposed by [11] is given. It provides a Gaussian mixture representation of the posterior $p(\underline{\theta} | z_{1:k})$ according to

$$p(\underline{\theta} | z_{1:k}) = \sum_{i=1}^{L} \omega_{k,i} \cdot \mathcal{N}(\underline{\theta}; \underline{\hat{\theta}}_{k,i}, \mathbf{C}_{k,i}) ,$$

where L is the number of mixture components, $\omega_{k,i}$ are non-negative weights summing up to one, and $\underline{\hat{\theta}}_{k,i}$, $\mathbf{C}_{k,i}$ are the mean and covariance matrix, respectively, of the i-th Gaussian component. By substituting this mixture representation into the Bayesian recursion (5), one has to evaluate two terms: the product of the Gaussian mixture with the likelihood (3.2) and the integration required for the normalization constant $p(z_k | z_{1:k-1})$. In order to obtain the desired quantities, the AGMF performs four steps as depicted in Figure 1: linearization, splitting, filtering, and reduction. Each step is explained in the following, where the time index k is omitted for improved readability.

4.1 Statistical Linearization

The product of likelihood and mixture boils down to multiple products between the likelihood and each Gaussian component of the mixture. These individual products also appear in standard nonlinear Gaussian filters like the EKF or UKF. The AGMF utilizes the same technique also used in the UKF in order to provide an approximate solution of the individual products. By means of so-called *statistical linearization*, the nonlinearity inducted by the Gaussian plume model is transformed into a linear one, i.e., the nonlinear model (6) is approximated by means of the linear model

$$z \approx \mathbf{H}_i \cdot \underline{\theta} + b_i + v \, . \tag{8}$$

The terms $\mathbf{H}_i$ and b_i are obtained via statistical linear regression, where the Gaussian plume model $c(.)$ is evaluated at a set of weighted regression points $\mathcal{L} \triangleq \{\alpha_j^{(i)}, \underline{\theta}_j^{(i)}\}_{j=0\ldots N}$ with non-negative weights $\alpha_j^{(i)}$. The regression points are drawn *deterministically* from the i-th Gaussian component in such a way that sample mean and covariance of $\mathcal{L}$ coincide with the mean $\underline{\hat{\theta}}_i$ and covariance $\mathbf{C}_i$. The solution of the linear regression yields

$$\mathbf{H}_i = \left(\mathbf{C}^{\theta z}\right)^{\mathrm{T}} \mathbf{C}_i^{-1} \quad \text{and} \quad b_i = \hat{z} - \mathbf{H}_i \cdot \underline{\hat{\theta}}_i \tag{9}$$

for the required terms in (8), where $\mathcal{L}$ is used to approximate the mean and cross-covariance of z according to

$$\hat{z} \approx \sum_{j=0}^{N} \alpha_j^{(i)} \cdot z_j^{(i)} \, , \quad \mathbf{C}^{\theta z} \approx \sum_{j=0}^{N} \alpha_j^{(i)} \cdot \left(\underline{\theta}_j^{(i)} - \underline{\hat{\theta}}\right) \cdot \left(z_j^{(i)} - \hat{z}\right)^{\mathrm{T}} ,$$

respectively, with $z_j^{(i)} = c\left(.; \underline{\theta}_j^{(i)}\right)$ for $j = 0 \ldots N$. The terms in (9) minimize the square of the error

$$e_i \triangleq c\left(.; \underline{\theta}\right) - \mathbf{H}_i \cdot \underline{\theta} - b_i \tag{10}$$

in such a way that the error has zero mean and variance

$$\sigma_{e,i}^2 = \sigma_{z,i}^2 - \mathbf{H}_i \mathbf{C}_i \mathbf{H}_i^{\mathrm{T}} - \sigma_v^2 \, , \tag{11}$$

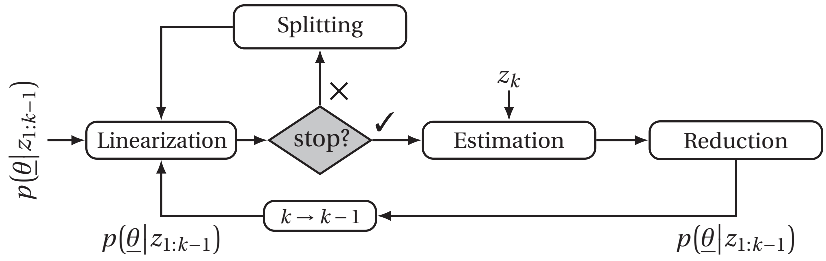

$p(\underline{\theta}|z_{1:k-1})$
Splitting
Linearization
stop?
×
✓
z_k
Estimation
Reduction
$k \to k-1$
$p(\underline{\theta}|z_{1:k-1})$
$p(\underline{\theta}|z_{1:k-1})$

Figure 1: Flow chart of the adaptive Gaussian mixture filter.

where $\sigma_{z,i}^2 \approx \sum_j \alpha_j^{(i)} \cdot \big(z_j^{(i)} - \hat{z}\big)\big(z_j^{(i)} - \hat{z}\big)^{\mathrm{T}} + \sigma_v^2$. The error variance (11) gives a good indication of the linearization error as it is merely zero iff $c(.)$ is affine. The Gaussian plume model approaches an affine function with an increasing distance between the sensor location $\underline{x}_r$ and the gas source.

4.2 Splitting

As statistical linearization (8) has to be performed for every component of the Gaussian mixture $p(\underline{\theta}|z_{1:k-1})$, it is called *local linearization* in the following. It typically provides a better estimation performance compared to a single *global* linearization (see [2]). A further performance improvement can be achieved by increasing the number of mixture components. It was proven in [1] that Gaussian mixture filters relying on a local linearization converge towards the optimal estimate when increasing the component number.

The AGMF adds additional components to the mixture by *splitting* existing ones, i.e., an existing component is replaced by several new components that share some statistics with the original component and that have lower weights and covariances. To select a component for splitting, AGMF takes the local linearization error induced by each component into account. Here, the statistical linearization described above already provides a measure via the error variance σ_e^2 in (11). Besides the linearization error, also the importance of a mixture component, which is given by its weight ω_i, is curcial. Both ingredients are combined in the so-called *selection criterion*

$$i^* = \arg \min_{i=1\ldots L} \{s_i\}\ , \ s_i \triangleq \omega_i^{\gamma} \cdot \Big(1 - \exp\big(-\sigma_{e,i}^2\big)\Big)^{1-\gamma} \tag{12}$$

for selecting component i^* for splitting. Here, the term $1-\exp\left(-\sigma_{e,i}^2\right)$ normalizes the error $\sigma_{e,i}^2$ of the i-th component to the interval [0,1]. The criterion considers the component weight and linearization error through a geometric interpolation with parameter $\gamma \in [0,1]$, where for $\gamma = 0$ only the linearization error is the determining factor and for $\gamma = 1$ it is the weight.

In order to trade the reduction of the linearization error off against controlling the computational load and the growth of the number of components, the Gaussian component selected for splitting is replaced by two Gaussians only in each splitting round. As splitting is performed recursively by the AGMF, the newly introduced components can be split again in the next rounds if the linearization error is still too high.

Let $\omega \cdot \mathcal{N}\left(\underline{\theta};\hat{\underline{\theta}},\mathbf{C}\right)$ be the component considered for splitting. It is replaced by two components according to

$$\omega \cdot \mathcal{N}\left(\underline{\theta};\hat{\underline{\theta}},\mathbf{C}\right) \approx \sum_{n=1}^{2} \omega_n \cdot \mathcal{N}\left(\underline{\theta};\hat{\underline{\theta}}_n,\mathbf{C}_n\right) ,$$

with parameters

$$\begin{aligned} &\omega_1 = \omega_2 = \tfrac{\omega}{2} , \\ &\hat{\underline{\theta}}_1 = \hat{\underline{\theta}} + \sqrt{\lambda}\cdot\mu\cdot\underline{\varepsilon} , \quad \hat{\underline{\theta}}_2 = \hat{\underline{\theta}} - \sqrt{\lambda}\cdot\mu\cdot\underline{\varepsilon} , \\ &\mathbf{C}_1 = \mathbf{C}_2 = \mathbf{C} - \lambda\cdot\mu\cdot\underline{\varepsilon}\,\underline{\varepsilon}^{\mathrm{T}} , \end{aligned} \tag{13}$$

where $\mu \in [-1,1]$ is a free parameter. The parametrization in (13) ensures moment preservation, i.e., the original Gaussian component and its split counterpart have the same mean and covariance. Furthermore, λ and $\underline{\varepsilon}$ in (13) are a particular eigenvalue and eigenvector, respectively, of $\mathbf{C}$. Splitting is merely performed along eigenvectors of $\mathbf{C}$, which is computationally cheap and numerically stable compared to arbitrary splitting directions. Among all possible eigenvectors, the one is chosen that currently induces the highest error according to (10), i.e., splitting is performed along the eigenvector where the Gaussian plume model posses the strongest nonlinearity.

As indicated in Figure 1, in every splitting round a *stopping criterion* is evaluated. Splitting stops, if at least one of the three following thresholds is reached:

Error threshold: The value s_i in the selection criterion (12) drops below $s_{\max} \in [0,1]$ for *every* component.

Component threshold: The number of mixture components excels $L_{\max}$.

Deviation threshold: The deviation between the original Gaussian mixture $p(\underline{\theta})$ and the mixture obtained via splitting $\tilde{p}(\underline{\theta})$ excels $d_{\max} \in [0,1]$.

The deviation considered for the latter threshold is determined by means of the normalized intergral squared distance measure

$$D\left(p(\underline{\theta}), \tilde{p}(\underline{\theta})\right) = \frac{\int \left(p(\underline{\theta}) - \tilde{p}(\underline{\theta})\right)^2 \mathrm{d}\underline{\theta}}{\int p(\underline{\theta})^2 \mathrm{d}\underline{\theta} + \int \tilde{p}(\underline{\theta})^2 \mathrm{d}\underline{\theta}} \in [0,1] \ .$$

Since splitting always introduces an approximation error to the original mixture, continuously monitoring the deviation limits this error.

4.3 Filtering

Let $\omega_i^s \cdot \mathcal{N}(\underline{\theta}; \hat{\underline{\theta}}_i^s, \mathbf{C}_i^s)$ be the Gaussians resulting from splitting with $i = 1 \ldots L^s$ and $L^s \in [L, L_{\max}] \subset \mathbb{N}$. Given the concentration measurement z_k, the recursive Bayesian update according to (5) boils down to a bank of Kalman filter updates thanks to the locally linearized models (8). Thus, the update of each (prior) Gaussian component gives rise to the parameters of the corresponding component of the posterior Gaussian mixture $p(\underline{\theta}|z_{1:k})$ according to

$$\begin{aligned}
\omega_i &= c \cdot \omega_i^s \cdot \mathcal{N}(z_k; \hat{z}_i, \sigma_{z,i}^2) \ , \\
\hat{\underline{\theta}}_i &= \hat{\underline{\theta}}_i^s + \mathbf{K}_i \cdot (z_k - \hat{z}_i) \ , \\
\mathbf{C}_i &= \mathbf{C}_i^s - \mathbf{K}_i \mathbf{H}_i \mathbf{C}_i^s \ ,
\end{aligned} \tag{14}$$

with predicted measurement $\hat{z}_i = \mathbf{H}_i \cdot \hat{\underline{\theta}}_i^s + b_i$, $\mathbf{H}_i$ and b_i according to (9), Kalman gain $\mathbf{K}_i = \mathbf{C}_i^s \mathbf{H}_i^{\mathrm{T}} / \sigma_{z,i}^2$, and innovation variance $\sigma_{z,i}^2 = \mathbf{H}_i \mathbf{C}_i^s \mathbf{H}_i^{\mathrm{T}} + \sigma_v^2 + \sigma_{e,i}^2$ with $\sigma_{e,i}^2$ being the linearization error variance (11). In the calculation of the weight ω_i in (14), $c = 1 / \sum_i \omega_i^s \cdot \mathcal{N}\left(z_k; \hat{z}_i, \sigma_{z,i}^2\right)$ is a normalization constant.

4.4 Reduction

As the number of components of $p(\underline{\theta}|z_{1:k})$ grows due to splitting, it is necessary to bound this growth in order to reduce the computational and memory demand of subsequent estimation steps. For this purpose, one can exploit the redundancy and similarity of Gaussian components. Furthermore, many components will have negligible weights and thus, they can be removed without introducing significant errors. To reduce a Gaussian mixture, many algorithms have been proposed in the recent years (see e.g. [12, 155, 207]). Most of these algorithms require a *reduction threshold* $L_{\text{red}} \ll L^s$ to which the number of components of the given Gaussian mixture has to be reduced. The reduction to L_{red} components closes the calculation of the posterior Gaussian mixture in (5) for time step k.

5 Results

The estimation performance of the AGMF is assessed in the following by means of real data in Section 5.2 and by means of a synthetic example in Section 5.3. However, as the AGMF is a generic estimator allowing specific parametrization depending on the estimation task, the selected setup is described first.

5.1 AGMF Parametrization

The three processing steps linearization, splitting, and reduction allow specialized parametrization. For the statistical linearization, the selection of the regression points is based on the scaled unscented transform [13]. Hence, the points and their weights are given by

$$\begin{aligned}
\underline{\theta}_0 &= \hat{\underline{\theta}} \,, & \alpha_0 &= \tfrac{\lambda^2 - d}{\lambda^2} \,, & & \\
\underline{\theta}_j &= \hat{\underline{\theta}} + \lambda \cdot \underline{P}_j \,, & \alpha_j &= \tfrac{1}{2\lambda^2} \,, & j &= 1 \ldots d \,, \\
\underline{\theta}_{d+j} &= \hat{\underline{\theta}} - \lambda \cdot \underline{P}_j \,, & \alpha_{d+j} &= \alpha_j \,, & j &= 1 \ldots d \,,
\end{aligned}$$

with d being the dimension of $\underline{\theta}$, $\lambda = \nu \cdot \sqrt{d + \kappa}$ being a scaling factor, and $\underline{P}_j$ being the j-th column of the matrix $\mathbf{P} = \sqrt{\mathbf{C}}$, where the matrix square root is calculated via the Cholesky decomposition. The free parameters in λ are chosen to $\nu = 1$, $\beta = 2$, and $\kappa = 0.5$ in the following experiments, which leads to equal weights α_j for all $j = 0 \ldots 2d$.

Both the mixing parameter γ in the selection criterion (12) and the displacement parameter μ in (13) are set to be 0.5. The thresholds for stopping component splitting are chosen to be $s_{\max} = 0$ and $d_{\max} = 1$, which actually disables these thresholds. Only the threshold $L_{\max}$ is active, but set differently depending on the considered experiment.

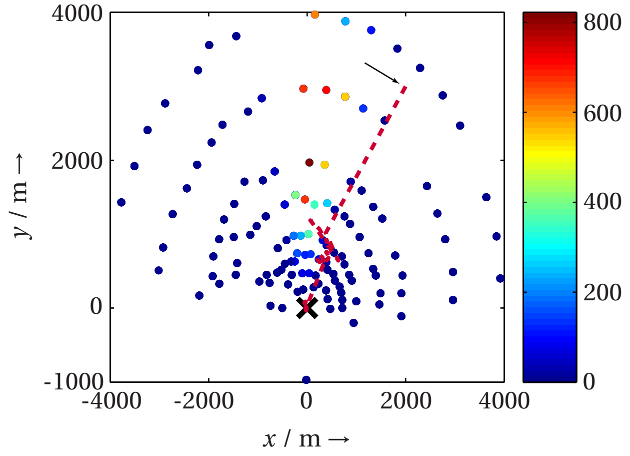
For reducing the Gaussian mixture after performing the filtering step, the reduction algorithm proposed by [17] is employed as it provides a good trade-off between computational demand and reduction error. The reduction threshold L_{red} is set to be $L_{\max}/8$.

5.2 Indianapolis Field Study

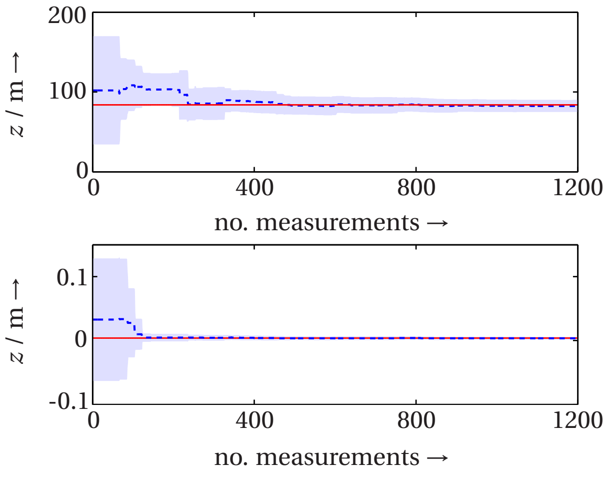
In the first experiment, it is demonstrated how the proposed source estimation solution performs on a real data set. For this purpose, the data acquired during the EPRI Indianapolis field study is considered, where SF_6 tracer was released from a $z_s = 83.8$m stack at a power plant in Indianapolis, Indiana, USA. Data was recorded by 160 ground-level sensors over 19 days in September and October 1985 for 8 to 9 hours every day. Details about the field study and the data can be found in [7].

In Figure 2a, the locations of the sensors and sensors' concentration measurements are depicted for the 19th September 1985. The source is located at the origin and the emission rate of the tracer gas is $q = 0.0041\,\text{g/s}$. Information about wind speed, wind direction, and atmospheric stability was made available by meteorological observations. The initial estimate of the source at time step $k = 0$ is given by a single Gaussian with mean vector $\hat{\underline{\theta}}_0 = [2000, 3000, 102, 0.033]^{\text{T}}$ and covariance matrix $\mathbf{C}_0 = \text{diag}(10^6, 10^6, 500, 0.001)$. Figures 2a and 2b show the convergence of the source estimate towards the true source location over time and with increasing number of concentration measurements, respectively. It is important to note that many sensor measurements (typically 60%-70%) provide a concentration measurement of almost zero as most of the sensor are outside the gas plume, as can be seem in Figure 2a. This explains the step-wise convergence of the estimate and reduction of the variance in Figure 2b.

The posterior density $p(\underline{\theta} | z_{1:k})$ after all $k = 1200$ measurements is depicted in Figure 3. It can be seen that the mean of the estimate is close to the true source parameters. Slight deviation from the ground truth is only observed for the emission rate, but still the true parameters are within the high confidence region of the estimate. Thus, the proposed estimator is not overconfident.



(a) The red dashed line marks the trajectory of the estimated source location $[x, y]^T$, whereas the true location of the source is marked by the black cross. Circular markers denote the sensor locations colored with the measured concentration in ppt.



(b) Estimate of source height z and emission rate q with increasing number of measurements. The shaded area denotes the 3-sigma confidence region and the red line indicates the true value.

Figure 2: Source estimate by AGMF based on the data from the Indianapolis field study.

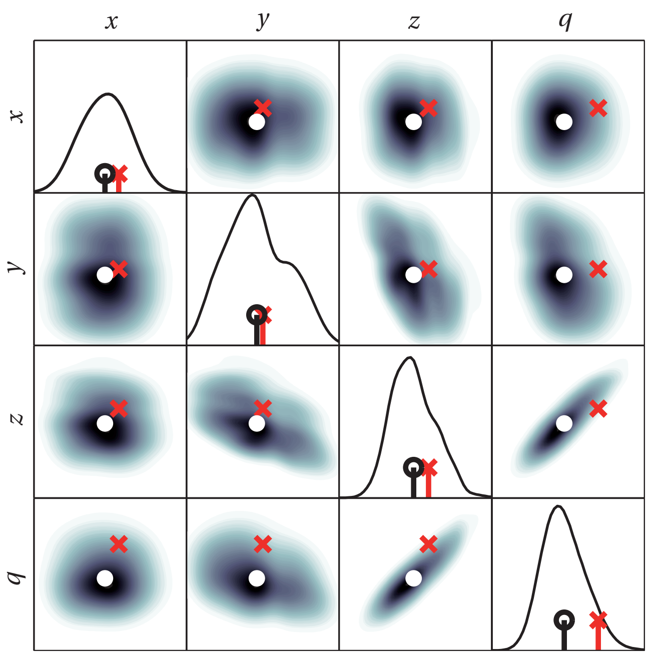


Figure 3: Bivariate posterior densities of the AGMF source estimate. The diagonal plots are the univariate marginal densities. Red crosses indicate the true value, while white and black circles, respectively, denote the mean of the respective density.

5.3 Simulation

In contrast to the previous experiment, a synthetic example is considered here allowing the comparison with state-of-the-art source estimation methods via Monte Carlo simulations. In this example, the locations and emission rates of two sources have to be estimated. Thus, the measurement model (7) for $N = 2$ applies here. The employed simulation parameters are listed in Table 1. The standard deviations σ_y, σ_z of the Gaussian plume model are assumed to take the form

$$\sigma(x) = a \cdot x \cdot (1 + b \cdot x)^{-c} \tag{15}$$

according to [5]. The constants in (15) depend on the atmospheric stability, for which class D is assumed—corresponds to “neutral” in accordance to the

Pasquill-Gifford classification scheme. As described in [6], the corresponding values of the constants for class D are as listed in the last two rows of Table 1.

For comparison, the following estimators are considered:

AGMF$_{16}$	Proposed source estimator with $L_{\max} = 16$.
AGMF$_{32}$	Proposed source estimator with $L_{\max} = 32$.
GMF$_{16}$	Gaussian mixture estimator by [20], i.e., AGMF without adaptation. The number of components remains constant at 16.
GMF$_{32}$	Same estimator as GMF$_{16}$ but with 32 mixture components.
UKF	Unscented Kalman filter proposed by [13], i.e., only a single Gaussian represents the posterior.
GN	Forward off-line method utilizing Gauß-Newton optimization proposed by [16].
MCMC	Backward off-line method utilizing Markov chain Monte Carlo sampling. The number of sampling steps is chosen in such a way that the runtime of MCMC is similar to AGMF$_{32}$.

Table 1: Simulation parameters, where $\mathcal{U}(a,b)$ is a uniform distribution over the interval $[a,b]$.

Parameter	Distribution / Value
location 1st source	$x_{s,1},\ y_{s,1} \sim \mathcal{U}(0\,\mathrm{m}, 100\,\mathrm{m})$
height 1st source	$z_{s,1} \sim \mathcal{N}(4\,\mathrm{m}, 1\,\mathrm{m}^2)$
emission rate 1st source	$q_1 \sim \mathcal{U}(0.005\,\mathrm{g/s}, 0.006\,\mathrm{g/s})$
location 2nd source	$x_{s,1},\ y_{s,2} \sim \mathcal{U}(0\,\mathrm{m}, 100\,\mathrm{m})$
height 2nd source	$z_{s,2} \sim \mathcal{N}(6\,\mathrm{m}, 1\,\mathrm{m}^2)$
emission rate 2nd source	$q_2 \sim \mathcal{U}(0.0075\,\mathrm{g/s}, 0.0085\,\mathrm{g/s})$
wind speed	$u \sim \mathcal{U}(1\,\mathrm{m/s}, 2\,\mathrm{m/s})$
wind direction	$\phi \sim \mathcal{U}(-\pi/4\,\mathrm{rad}, \pi/4\,\mathrm{rad})$
standard deviation σ_y	$a = 0.08,\ b = 0.0001,\ c = 0.5$
standard deviation σ_z	$a = 0.06,\ b = 0.0015,\ c = 0.5$

Table 2: Average distance and average source rate deviation in case of two sources.

	Strong Noise		Med. Noise		Low Noise	
	dist	rate	dist	rate	dist	rate
AGMF$_{16}$	49.9	29.0	37.1	31.9	37.5	27.3
AGMF$_{32}$	45.6	28.5	**34.6**	32.1	**34.2**	28.1
GMF$_{16}$	58.0	27.2	42.6	30.9	61.2	36.8
GMF$_{32}$	52.2	28.2	41.6	31.1	48.8	33.7
UKF	69.9	29.1	74.5	34.7	124.1	59.8
GN	63.8	**21.8**	63.5	**21.3**	64.2	**21.6**
MCMC	**34.3**	25.5	43.0	29.9	75.8	36.3

The initial estimate of each estimator at time step $k = 0$ is Gaussian with mean vector $\underline{\hat{\theta}}_0$ drawn randomly from $\mathcal{N}(\underline{\theta}; \mathbf{C})$ with

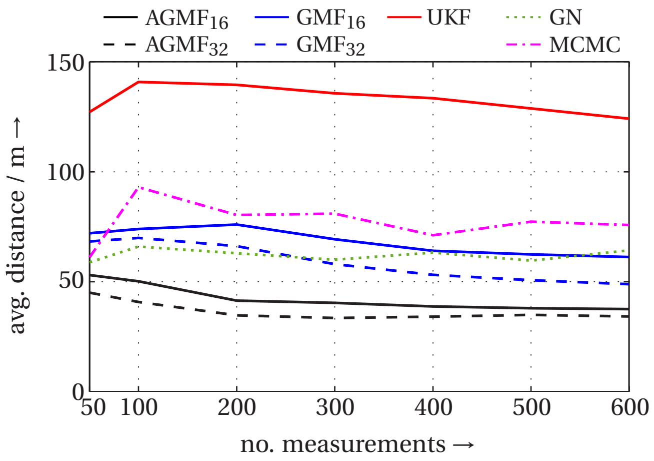
$$\underline{\theta} = \left[x_{s,1}, y_{s,1}, z_{s,1}, q_1, x_{s,2}, y_{s,2}, z_{s,2}, q_2\right]^{\mathrm{T}} , \tag{16}$$

$$\mathbf{C} = \operatorname{diag}\left(50^2, 50^2, 10^2, 0.0025^2, 50^2, 50^2, 10^2, 0.0025^2\right) . \tag{17}$$

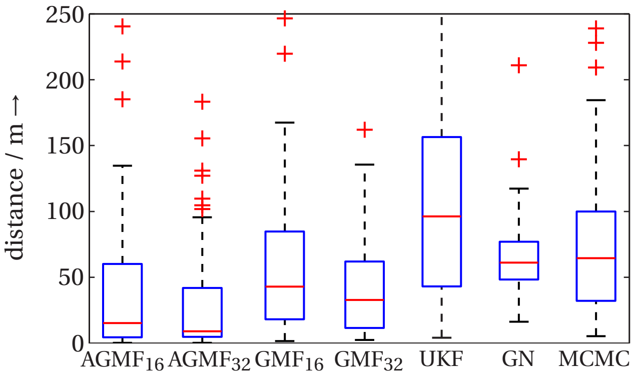
The initial covariance matrix is set to be $\mathbf{C}_0 = \mathbf{C}$.

Three different sensor noise levels for σ are considered: $10\,\mathrm{mg/m^3}$ (strong noise), $5\,\mathrm{mg/m^3}$ (medium noise), and $1\,\mathrm{mg/m^3}$ (low noise). For each noise level, 100 Monte Carlo simulation runs are performed. In each run, 600 concentration measurements are acquired at locations drawn randomly from $\mathcal{U}(0\,\mathrm{m}, 200\,\mathrm{m}) \times \mathcal{U}(0\,\mathrm{m}, 200\,\mathrm{m})$. In Table 2, average values of the distance between source estimates and true source locations for both sources as well as the average absolute deviation between estimated and true emission rate are listed. For low and medium noise, the proposed AGMF provides the lowest source distance, where allowing more components for splitting leads to a slightly better estimation performance. Regarding the emission rate, GN is the most accurate estimator. In case of strong noise, MCMC is the best estimator with respect to the source location. MCMC however, is an off-line estimator, i.e., estimates become available after processing all measurements in a batch, while the AGMF continuously provides estimates at runtime.

The superior estimation performance of the AGMF compared to the other estimators in case of low noise does not vary with the number of concentration measurements as shown in Figure 4a. From 50 to 600 measurements AGMF al-



(a) Average distance depending on the number of measurements. For the batch methods GN and MCMC, for each number of measurements separate estimation runs have been performed.



(b) Median (red line), lower and upper quantiles (blue box), and spread (black lines) of the distances for 600 measurements. Red crosses indicate outliers.

Figure 4: Distances between estimated source position and true source positions for the low noise case.

Table 3: Average and standard deviation of the number of mixture components per time step.

	Strong Noise	Med. Noise	Low Noise
AGMF_{16}	12.46 ± 8.07	9.77 ± 8.93	8.59 ± 8.96
AGMF_{32}	25.13 ± 16.02	19.64 ± 17.85	16.53 ± 17.86

ways provides the smallest distance between estimated and true source location. A similar result is obtained for the other noise levels, except that MCMC always provides a better estimate for the strong noise case.

Figure 4b shows that the majority of the estimates provided by the AGMF is better than the average listed in Table 2. Some outliers significantly lower the average performance. These outliers result from random initializations, where the initial estimated source location is far outside the sensor area $[0\text{m}, 200\text{m}] \times [0\text{m}, 200\text{m}]$. Due to the potentially large distance between estimate and measurement location, the Gaussian plume model become almost linear, which is also indicated by the lineariztion error measure (11). As a consequence no splitting is performed and thus, the AGMF degrades to a simple UKF. By means of allowing also splitting towards the sensor location would resolve this issue and would result in a better performance.

Of all on-line estimators, AGMF performs best. This clearly shows that a single Gaussian and also a fixed Gaussian mixture representation are not sufficient. By means of the adaptation via splitting, a much better source estimation is possible. It is worth mentioning that it is not necessary to utilize the maximum number of components $L_{\max}$ at all time steps in order to provide a meaningful representation of the posterior density. As shown in Table 3, the average number of components per time step is significantly below $L_{\max}$, whereas the lower the measurement noise the lower the required number of components. Thus, the proposed AGMF can carefully control the demand of splits.

6 Conclusion

The state-of-the-art in dispersion source estimation mainly focuses on MCMC methods. While this allows accurate estimates, only batch processing is possible. The proposed adaptive Gaussian mixture filter shows that a similar and in some cases an even better estimation performance can be obtained with an on-line estimator. Therefore, it is of paramount importance to not rely on a single Gaussian or a fixed Gaussian mixture representation. By means of splitting that adapts the Gaussian mixture to the nonlinearity of the Gaussian plume dispersion model, a significant improvement can be achieved.

References

[1] Simo Ali-Löytty. *Gaussian Mixture Filters in Hybrid Positioning.* PhD thesis, Tampere University of Technology, Tampere, Finland, August 2009.

[2] Daniel L. Alspach and Harold W. Sorenson. Nonlinear Bayesian Estimation using Gaussian Sum Approximation. *IEEE Transactions on Automatic Control*, 17(4):439–448, August 1972.

[3] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[4] Mieczyslaw Borysiewicz, Anna Wawrzynczak, and Piotr Kopka. Bayesian-Based Methods for the Estimation of the Unknown Model's Parameters in the Case of the Localization of the Atmospheric Contamination Source. *Foundations of Computing and Decision Sciences*, 37(4):253–270, 2012.

[5] G. A. Briggs. Diffusion estimation for small emissions, Atmospheric Turbulence and Diffusion Laboratory Contribution. Technical Report 79, National Oceanic and Atmospheric Administration, Oak Ridge, 1973.

[6] M. D. Carrascal, M. Puigcerver, and P. Puig. Sensitivity of Gaussian plume model to dispersion specifications. In *Theoretical and Applied Climatology*, volume 48, pages 147–157. Springer, 1993.

[7] Steven Hanna, Joseph Chang, and Helge R. Olesen. *Indianapolis Tracer Data and Meteorological Data*, May 1997.

[8] Bill Hirst, Philip Jonathan, Fernando G. del Cueto, David Randell, and Oliver Kosut. Locating and quantifying gas emission sources using remotely obtained concentration data. *Atmospheric Environment*, 74:141–158, August 2013.

[9] Ekkehard Holzbecher. *Environmental Modeling.* Springer, 2nd edition, 2012.

[10] Frédéric Hourdin and Olivier Talagrand. Eulerian backtracking of atmospheric tracers. I: Adjoint derivation and parametrization of subgrid-scale transport. *Quarterly Journal of the Royal Meteorological Society*, 132(615):567–583, January 2006.

[11] Marco F. Huber. Adaptive Gaussian Mixture Filter Based on Statistical Linearization. In *Proceedings of the 14th International Conference on Information Fusion (Fusion)*, Chicago, Illinois, July 2011.

[12] Marco F. Huber and Uwe D. Hanebeck. Progressive Gaussian Mixture Reduction. In *Proceedings of the 11th International Conference on Information Fusion (Fusion)*, Cologne, Germany, July 2008.

[13] Simon J. Julier and Jeffrey K. Uhlmann. Unscented Filtering and Nonlinear Estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[14] Vladimir Maz'ya and Gunther Schmidt. On approximate approximations using gaussian kernels. *IMA Journal of Numerical Analysis*, 16:13–29, 1996.

[15] K. Shankar Rao. Source estimation methods for atmospheric dispersion. *Atmospheric Environment*, 41:6964–6973, 2007.

[16] Alison Rudd, Alan G. Robins, Jason J. Lepley, and Stephen E. Belcher. An Inverse Method for Determining Source Characteristics for Emergency Response Applications. *Boundary-Layer Meteorology*, 144(1):1–20, July 2012.

[17] Andrew R. Runnalls. Kullback-Leibler Approach to Gaussian Mixture Reduction. *IEEE Transactions on Aerospace and Electronic Systems*, 43(3):989–999, July 2007.

[18] Simo Särkkä. *Bayesian Filtering and Smoothing.* Cambridge University Press, 2013.

[19] Inanc Senocak, Nicolas W. Hengartner, Margaret B. Short, and W. Brent Daniel. Stochastic Event Reconstruction of Atmospheric Contaminant Dispersion Using Bayesian Inference. *Atmospheric Environment*, 42(33):7718–7727, October 2008.

[20] Miroslav Simandl and Jindrich Duník. Sigma Point Gaussian Sum Filter Design Using Square Root Unscented Filters. In *Proceedings of the 16th IFAC World Congress*, Prague, Czech Republic, July 2005.

[21] Michael D. Sohn, Pamela Reynolds, Navtej Singh, and Ashok J. Gadgil. Rapidly Locating and Characterizing Pollutant Releases in Buildings. *Journal of the Air & Waste Management Association*, 52(12):1422–1432, 2002.

[22] John M. Stockie. The Mathematics of Atmospheric Dispersion Modelling. *SIAM Review*, 53(2):349–372, 2011.

[23] Yong Zhang and Li Wang. Particle Filtering Method for Source Localization in Wireless Sensor Network. In *Advanced Technology in Teaching: Selected papers from the 2012 International Conference on Teaching and Computational Science (ICTCS 2012)*, volume 163, pages 517–523. Springer, 2013.

Paper O

Bayesian Active Object Recognition via Gaussian Process Regression

Authors: Marco F. Huber, Tobias Dencker, Masoud Roschani, and Jürgen Beyerer

Edited version of the paper:

M. F. Huber, T. Dencker, M. Roschani, and J. Beyerer. Bayesian Active Object Recognition via Gaussian Process Regression. In *Proceedings of the 15th International Conference on Information Fusion (Fusion)*, pages 1718–1725, Singapore, July 2012.

Bayesian Active Object Recognition via Gaussian Process Regression

Marco F. Huber*, Tobias Dencker**, Masoud Roschani**, and Jürgen Beyerer**

* AGT International
Darmstadt, Germany
marco.huber@ieee.org

** Institute for Anthropomatics
Karlsruhe Institute of Technology (KIT), Germany
tobias.dencker@kit.edu,
masoud.roschani@kit.edu,
juergen.beyerer@kit.edu

Abstract

This paper is concerned with a Bayesian approach of actively selecting camera parameters in order to recognize a given object from a finite set of object classes. Gaussian process regression is applied to learn the likelihood of image features given the object classes and camera parameters. In doing so, the object recognition task can be treated as Bayesian state estimation problem. For improving the recognition accuracy and speed, the selection of appropriate camera parameters is formulated as a sequential optimization problem. Mutual information is considered as optimization criterion, which aims at maximizing the information from camera observations or equivalently at minimizing the uncertainty of the state estimate.

1 Introduction

Research on computer vision mostly focuses on the object or scene observed by the camera system. It is assumed that the parameters of the camera (e.g., position, illumination, or focus) are given or determined off-line in a time-consuming trial-and-error process involving human interaction. Particular operations are then applied on the acquired images in order to solve the considered vision task like recognizing an object. In such *passive* vision systems, the camera parameters aer not adapted on-line. This is in contrast to an *active* vision system, where the next camera observation is carefully planned based on the previously acquired images and prior information about the considered scene.

While various approaches for passive object recognition exist (see e.g. [23] and references therein), active object recognition still is in its early stages. One of the first approaches to active object recognition can be found in [2], where the object models are learned via the eigenspace approach introduced in [15]. The planning algorithm greedily chooses the view that leads to the maximum entropy reduction of the object hypotheses. In [8], from a finite set of views the one maximizing the mutual information between observations and classes is selected. The approach is designed for arbitrary features, but requires approximate mutual information calculation via Monte Carlo sampling, which prevents a direct extension to continuous views. An upper bound of the Jeffrey divergence is employed in [13]. Again, merely a finite set of viewpoints is considered. Reinforcement learning approaches for active object recognition are proposed in [6, 16]. Here, learning the object models and planning is performed simultaneously. A comparison of some of the aforementioned approaches can be found in [5].

The active object recognition method proposed in this paper consists of two parts (see Figure 1 on page 534). In the off-line *learning* part described in Section 4.1, for each object a so-called object model is created. For varying camera parameters, e.g., focus or position, 2D images of each 3D object are generated. Gaussian process regression is then applied on the sample images to learn the object models. As explained in Section 3, Gaussian processes can be considered distributions over functions and thus, allow capturing the variations in images due to noise and errors in image pre-processing.

In the on-line *recognition* part, planning the next-best camera view (see Section 4.3) and Bayesian state estimation (see Section 4.2) are performed alternately. For planning, mutual information is maximized with respect to the camera parameters. Mutual information quantifies the reduction of the uncertainty in the current object estimate given a particular camera parameter. Based on the chosen parameter, the object estimate is updated via Bayesian estimation under consideration of the learned object models.

In contrast to prior art, the proposed method is very general as it is not restricted to specific image features. Furthermore, camera parameters can be arbitrary and continuous valued. All derivations in this paper regarding Bayesian estimation hold for arbitrary Gaussian process kernel functions. The performance of the proposed approach is demonstrated by means of simulations in Section 5.

2 Problem Formulation

In this paper, the object recognition problem is treated in a probabilistic fashion in order to account for uncertainties arising for example from camera noise, occlusion, or feature extraction. Based on a feature vector $\underline{z}_k \in \mathcal{Z} \subseteq \mathbb{R}^{n_z}$ acquired from images at stage $k = 0,1,\ldots$, the goal is to estimate the true latent object class $x \in \mathcal{X} = \{x_1, x_2, \ldots, x_N\} \subset \mathbb{N}$, with N being the finite number of possible object classes. For estimation purposes, the true object class is approximated by a discrete random variable $x_k \in \mathcal{X}$, which forms the object class estimate. By means of the camera parameters $\underline{a}_k \in \mathcal{A} \subseteq \mathbb{R}^{n_a}$ the estimation process can be actively driven. Potential camera parameters are position, orientation, focal length, or exposure time, just to name a few.

The object class estimate x_k given all features and camera parameters up to and including stage k is characterized via the probability distribution $p_{k|k} \triangleq p\big(x_k \big| \underline{z}_{0:k}, \underline{a}_{0:k}\big)$, with $\underline{z}_{0:k} = \big(\underline{z}_0, \underline{z}_1, \ldots, \underline{z}_k\big)$. It is calculated recursively by means of Bayes' equation [20] according to

$$p_{k|k} = \frac{1}{c} \cdot p\big(\underline{z}_k \big| x_k, \underline{a}_k\big) \cdot p_{k|k-1} \,, \tag{1}$$

with normalization constant $c \triangleq p\big(\underline{z}_k \big| \underline{z}_{0:k-1}, \underline{a}_{0:k}\big)$ and the prior distribution $p_{k|k-1} \triangleq p\big(x_k \big| \underline{z}_{0:k-1}, \underline{a}_{0:k-1}\big) = p_{k-1|k-1}$, i.e., the distribution at stage $k-1$. The recursion (1) commences from $p_0 \triangleq p(x_0)$ being the prior distribution of the object class estimate at stage $k=0$. Furthermore, $p\big(\underline{z}_k \big| x_k, \underline{a}_k\big)$ in (1) is the likelihood defined by the nonlinear transformation

$$\underline{z}_k = \underline{h}\big(x_k, \underline{a}_k\big) + \underline{v}_k \,. \tag{2}$$

This *measurement model* with nonlinear measurement function $\underline{h}(.)$ relates the object class to a feature vector given the camera parameters. Here, the measurement noise $\underline{v}_k$ subsumes all uncertainties arising during image acquisition.

So far, the action[1] $\underline{a}_k$ was assumed to be given. But in active object recognition, an action is chosen automatically by the imaging system itself for acquiring high

1 The terms 'state', 'observation', 'action' are used interchangeably for 'object class', 'feature vector', 'camera parameter' from now on.

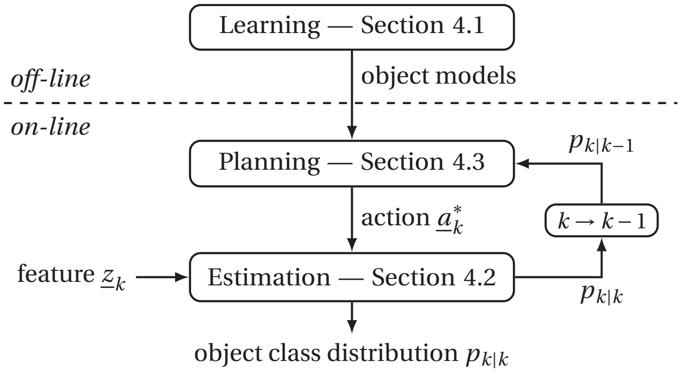


Figure 1: Flow chart of the active object recognition system.

informative observations. For this purpose, the optimization problem

$$\underline{a}_k^* = \arg \max_{\underline{a}_k} \mathrm{I}\left(x_k, \underline{z}_k \middle| \underline{a}_k\right) \tag{3}$$

is formulated to determine the optimal action $\underline{a}_k^*$ to be applied at stage k. Since solving (3) results in the camera parameters to be applied next, it is often referred to as *next-best-view* planning (see e.g. [19]). As target function in (3), the mutual information $\mathrm{I}\left(x_k, \underline{z}_k \middle| \underline{a}_k\right)$ between state and observation given an action is considered. This measure quantifies the amount of information the knowledge of an observation revels about the state and vice versa. It is closely related to Shannon's entropy and zero only iff both variables are independent [4].

For solving the next-best-view problem given by (3), several problems arise: 1) Analytical expressions for the measurement model (2) and the likelihood $p\left(\underline{z}_k \middle| x_k, \underline{a}_k\right)$, respectively, are not given in general as both describe a complex transformation of a potentially high-dimensional feature vector to an abstract object class. 2) Calculating $p_{k|k}$ in (1) cannot be performed in closed form for arbitrary likelihoods and priors $p_{k|k-1}$ [20]. 3) Evaluating mutual information is only possible for some special cases, e.g., if x_k and $\underline{z}_k$ are normally distributed. 4) The optimization problem is non-convex and thus, getting trapped in a sub-optimal solution becomes an issue. A novel active object recognition method addressing these problems is described in the following sections.

3 Gaussian Process Regression

To tackle the issue of not having analytic expressions of the measurement model and the likelihood, a machine learning tool named Gaussian processes (GPs) is employed. GPs allow non-parametric learning of regression functions from noisy training data. They can be considered Gaussian posterior distributions over functions conditioned on the training data [18]. Thus and in contrast to classical regression approaches, GPs provide not only a regression function but also provide uncertainty estimates (error bars) depending on the noise and the variability of the data.

For GP regression, a set of training data $\mathcal{D} = \{(\underline{x}_1, y_1), (\underline{x}_2, y_2), \dots, (\underline{x}_n, y_n)\}$ is assumed to be drawn from the noisy process

$$y_i = h(\underline{x}_i) + \varepsilon \,, \tag{4}$$

where $\underline{x}_i$ are the training inputs, y_i are the training outputs, and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is zero-mean Gaussian noise with variance σ^2. For brevity reasons, $\mathbf{X} = [\underline{x}_1, \dots, \underline{x}_n]$ are all training inputs and $\underline{y} = [y_1, \dots, y_n]^{\mathrm{T}}$ are the corresponding training outputs in the following.

The GP is used to infer the latent function $h(.)$ from the data $\mathcal{D}$ and is completely specified by a mean function $m(.)$ and a positive semi-definite covariance function $k(.,.)$, also called a *kernel*. Throughout this paper, a zero mean function and the squared exponential (SE) kernel

$$k(\underline{x}, \underline{x}') = \alpha^2 \cdot \exp\left(-\tfrac{1}{2}(\underline{x} - \underline{x}')^{\mathrm{T}} \Lambda^{-1} (\underline{x} - \underline{x}')\right)$$

are used, where Λ is a diagonal matrix of the characteristic length-scales for each input dimension and α^2 is the variance of the latent function h. It is worth mentioning that the active object recognition approach proposed in this paper is not restricted to an SE kernel. All derivations presented in the following hold for arbitrary kernels.

The posterior distribution of the function value $h_* = h(\underline{x}_*)$ for an arbitrary test input $\underline{x}_*$ is Gaussian with mean

$$\hat{h}(\underline{x}_*) = \mathrm{E}\{h_*\} = \underline{k}_*^{\mathrm{T}} (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \underline{y} \,, \tag{5}$$

and variance

$$\sigma_h^2(\underline{x}_*) = \mathrm{var}\{h_*\} = k_{**} - \underline{k}_*^{\mathrm{T}} \left(\mathbf{K} + \sigma^2 \mathbf{I}\right)^{-1} \underline{k}_* \, , \tag{6}$$

with E{.} being the expectation value, var{.} being the variance, $\underline{k}_* \triangleq k(\mathbf{X}, \underline{x}_*)$, $k_{**} \triangleq k(\underline{x}_*, \underline{x}_*)$, and $\mathbf{K}$ being the kernel matrix with elements $K_{ij} = k(x_i, x_j)$. Note that the variance depends on the noise ε as well as on the correlation between test input and training data.

The parameters σ, α, Λ of a GP are called the *hyperparameters*, which are learned automatically by maximizing the log-likelihood of the training data using numerical optimization [18]. Learning the hyperparameters corresponds to selecting a GP model describing the training data and thus, the process (4) adequately.

4 Active Object Recognition

The GP regression introduced in the previous section forms the basis of the proposed active object recognition approach. All components necessary for object recognition using GP regression are described in the following. For an overview and an illustration of the interactions between the components see Figure 1 on page 534.

4.1 Learning Object Models

To apply GP regression, it is necessary to map the considered measurement model (2) to the latent process (4). It is obvious that in (4) merely one-dimensional outputs are considered. In object recognition however, multi-dimensional outputs resulting from feature extraction are typical. The straightforward way used in this paper to apply GP regression to the multi-dimensional case is to learn a separate GP for each output dimension $e = 1, \ldots, n_z$. Thus, n_z GPs are learned independently using the same training inputs $\mathbf{X}$ but different training outputs $\underline{z}^e = \left[z_1^e, \ldots, z_n^e\right]^{\mathrm{T}}$ for each output dimension e. In doing so, it is assumed that any two output dimensions are conditionally independent given the input. For a deterministic input—here the deterministic action $\underline{a}$—this results in a posterior Gaussian with diagonal covariance matrix. For a uncertain input however, the covariance matrix is no longer diagonal [7]. An alternative approach resulting in non-diagonal covariance matrices even for deterministic inputs is the recently developed multi-output GP regression (see for example [3]).

Furthermore, learning the GPs for each output dimension has to be performed independently for each object class x_l, $l = 1,\dots,N$. This results in N multi-variate GPs $\underline{h}_l(.) \sim \mathcal{GP}$ of dimension n_z named *object models* in the following. To learn an object model $\underline{h}_l$, samples $\underline{a}_i$, $i = 1,\dots,n$ of the action space $\mathcal{A}$ are used as training inputs $\mathbf{X}$. For each input sample $\underline{a}_i$, an object of the class $x_l \in \mathcal{X}$ is observed by the camera resulting in the feature vector $\underline{z}_i = \left[z_i^1, z_i^2, \dots, z_i^{n_z}\right]^{\mathrm{T}}$ acting as training output. In total, for n_z output dimensions and N object classes, $n_z \times N$ GPs are learned. Since learning these measurement models is an off-line task (see Figure 1), the required computation time is independent of the computation time for object recognition. Furthermore, for high-dimensional features, which may be obtained for instance by means of the scale-invariant feature transform (SIFT, [14]), dimensionality reduction techniques like principal component analysis [1] or GP latent variable models [24] can be employed in order to reduce the number of GPs to be learned.

4.2 Bayesian Estimation

Given the learned object models, the next component towards an active object recognition is the estimation of the object class given an arbitrary but fixed action $\underline{a}_k \in \mathcal{A}$. Determining the next-best action is content of Section 4.3.

To solve Bayes' equation (1), it is at first necessary to provide the representations of all involved distributions.

Prior Distribution

As the latent object class x is a discrete random variable, the prior distribution $p_{k|k-1}$ at stage k can be characterized by means of

$$p_{k|k-1} = \sum_{i=1}^{N} \omega_{k-1,i} \cdot \delta_{x_k,i} \ , \tag{7}$$

where the weight $\omega_{k-1,i}$ represents the probability that object x belongs to class i. The weights are non-negative and sum up to one. Further, $\delta_{x_k,i}$ is defined as

$$\delta_{x_k,i} = \begin{cases} 1, & \text{if } x_k = i \\ 0, & \text{otherwise} \end{cases} \ . \tag{8}$$

and known as the *Kronecker delta*.

Likelihood

In case of a given object class $x_k = i$, the likelihood $p(\underline{z}_k | x_k = i, \underline{a}_k)$ corresponds to the GP $\underline{h}_i(.)$. If in addition the action $\underline{a}_k$ is given, the likelihood becomes a Gaussian density $\mathcal{N}(\underline{z}_k; \hat{\underline{z}}_{k,i}, \mathbf{C}^z_{k,i})$ with mean vector and covariance matrix according to

$$\begin{aligned}
\hat{\underline{z}}_{k,i} &= \left[\hat{z}^1_{k,i}, \hat{z}^2_{k,i}, \ldots, \hat{z}^{n_z}_{k,i}\right]^{\mathrm{T}} , \\
\mathbf{C}^z_{k,i} &= \operatorname{diag}\Big(\big(\sigma^1_{k,i}\big)^2, \big(\sigma^2_{k,i}\big)^2, \ldots, \big(\sigma^{n_z}_{k,i}\big)^2\Big) ,
\end{aligned} \tag{9}$$

respectively. The elements in (9) corresponding to dimension $e = 1, \ldots, n_z$ are calculated according to (5) and (6), respectively, with the given action $\underline{a}_k$ being the test input and $\underline{z}^e$ being the training output vector. Overall, the likelihood for a fixed action $\underline{a}_k$ can be characterized by means of the hybrid conditional distribution

$$p(\underline{z}_k | x_k, \underline{a}_k) = \sum_{i=1}^{N} \delta_{x_k, i} \cdot \mathcal{N}\left(\underline{z}_k; \hat{\underline{z}}_{k,i}, \mathbf{C}^z_{k,i}\right) . \tag{10}$$

It is important to note that for a fixed observation $\underline{z}_k$—as required for solving Bayes' equation—the conditional distribution in (10) becomes a weighted sum of Kronecker deltas as in (7), because all Gaussian components are evaluated at $\underline{z}_k$ and thus, become scalar weighting coefficients.

Normalization Constant

Finally, the normalization constant c in (1) can be calculated by marginalizing the product of prior and likelihood over x_k, which results in

$$\begin{aligned}
c = p(\underline{z}_k | \underline{z}_{0:k-1}, \underline{a}_{0:k}) &= \sum_{x_k} \underbrace{p(x_k, \underline{z}_k | \underline{z}_{0:k-1}, \underline{a}_{0:k})}_{= p(\underline{z}_k | x_k, \underline{a}_k) \cdot p_{k|k-1}} \\
&= \sum_{x_k} \left(\sum_{i=1}^{N} \omega_{k-1,i} \cdot \delta_{x_k,i} \cdot \mathcal{N}\left(\underline{z}_k; \hat{\underline{z}}_{k,i}, \mathbf{C}^z_{k,i}\right) \right) \\
&= \sum_{i=1}^{N} \omega_{k-1,i} \cdot \mathcal{N}\left(\underline{z}_k; \hat{\underline{z}}_{k,i}, \mathbf{C}^z_{k,i}\right) .
\end{aligned} \tag{11}$$

Thus, the normalization constant is a *Gaussian mixture* evaluated at the given observation $\underline{z}_k$.

Posterior Distribution

With the closed-form representations of all required distributions at hand, it is now possible to solve Bayes' equation resulting in the posterior distribution of x_k

$$\begin{aligned} p_{k|k} &= \frac{1}{c} \cdot \left(\sum_{i=1}^{N} \delta_{x_k,i} \cdot \mathcal{N}\left(\underline{z}_k; \underline{\hat{z}}_{k,i}, \mathbf{C}_{k,i}^{z}\right) \right) \cdot \left(\sum_{i=1}^{N} \omega_{k-1,i} \cdot \delta_{x_k,i} \right) \\ &= \frac{1}{c} \cdot \sum_{i=1}^{N} \omega_{k-1,i} \cdot \delta_{x_k,i} \cdot \mathcal{N}\left(\underline{z}_k; \underline{\hat{z}}_{k,i}, \mathbf{C}_{k,i}^{z}\right) = \sum_{i=1}^{N} \omega_{k,i} \cdot \delta_{x_k,i} \end{aligned}$$

with weights $\omega_{k,i} \triangleq \frac{1}{c} \cdot \omega_{k-1,i} \cdot \mathcal{N}\left(\underline{z}_k; \underline{\hat{z}}_{k,i}, \mathbf{C}_{k,i}^{z}\right)$. As expected, the incorporation of a new observation $\underline{z}_k$ leads to an adaption of the prior probability $\omega_{k-1,i}$ of each object class i depending on the individual likelihood $\mathcal{N}\left(\underline{z}_k; \underline{\hat{z}}_{k,i}, \mathbf{C}_{k,i}^{z}\right)$ of the object class.

4.3 Next-Best-View Planning

The final component in Figure 1 is the planning of the next-best-view and optimal action $\underline{a}_k^* \in \mathcal{A}$, respectively, allowing for fast and accurate object recognition. As discussed in Section 2, the optimal action results from solving the optimization problem (3), where the mutual information

$$\begin{aligned} \mathrm{I}\left(x_k, \underline{z}_k | \underline{a}_k\right) &= \mathrm{H}\left(x_k\right) - \mathrm{H}\left(x_k | \underline{z}_k, \underline{a}_k\right) && (12) \\ &= \mathrm{H}\left(\underline{z}_k | \underline{a}_k\right) - \mathrm{H}\left(\underline{z}_k | x_k, \underline{a}_k\right) && (13) \end{aligned}$$

is employed for quantifying the utility of a particular action $\underline{a}_k \in \mathcal{A}$. In (12) and (13), the first term H(.) denotes Shannon's entropy

$$\mathrm{H}(x) = -\sum_{x} p(x) \cdot \log p(x) \tag{14}$$

for discrete random variables and the differential entropy

$$\mathrm{H}(x) = -\int_{\mathcal{X}} p(x) \cdot \log p(x) \, \mathrm{d}x$$

for continuous random variables, respectively (see [4]). The second term H(.|.) denotes the *conditional entropy* given by

$$\mathrm{H}(z|x) = -\int_{\mathcal{X}} p(x) \int_{\mathcal{Z}} p(z|x) \cdot \log p(z|x) \, \mathrm{d}z \, \mathrm{d}x \tag{15}$$

for continuous random variables x and z. By replacing the integrals with sums, a similar expression for the conditional entropy can be found for discrete random variables.

Evaluation of Mutual Information

Unfortunately, neither (12) nor (13) allow an analytical calculation of the mutual information value. An approximate evaluation of the mutual information based on (12), however, is inappropriate for many reasons. While the first term $\mathrm{H}\left(x_k\right)$ is straightforward to evaluate as it is Shannon's entropy (14) of the discrete prior distribution $p_{k|k-1}$, the second conditional entropy term can only be evaluated approximately by discretizing the Gaussian mixture distribution $p\left(\underline{z}_k \middle| \underline{z}_{0:k-1}, \underline{a}_{0:k}\right)$, e.g., by means of random sampling or the unscented transform [10]. Depending on the number of samples used, this approach of approximating mutual information becomes computationally demanding. For each sample, Bayes' equation has to be evaluated completely in order to provide the posterior distribution $p_{k|k}$ required for the inner integral in (15). Furthermore, random sampling precludes classical optimization techniques like gradient descent for solving the optimization problem (3).

Directly approximating mutual information via (13) is also critical, but (13) allows calculating a lower bound, which is very convenient for the maximization in (3). Here, the first term needs special treatment as it requires the calculation of the entropy of the Gaussian mixture (11), which is not possible in closed form in general due to the logarithm of a sum of exponential functions. Fortunately, the entropy of a Gaussian mixture can be bounded from below according to [12]

$$\begin{aligned}
\mathrm{H}\left(\underline{z}_k \middle| \underline{a}_k\right) &= -\int_{\mathcal{Z}} p\left(\underline{z}_k \middle| \underline{a}_k\right) \cdot \log p\left(\underline{z}_k \middle| \underline{a}_k\right) \, \mathrm{d}\underline{z}_k \\
&= -\sum_{i=1}^{N} \omega_{k-1,i} \int_{\mathcal{Z}} \mathcal{N}\left(\underline{z}_k; \hat{\underline{z}}_{k,i}, \mathbf{C}_{k,i}^z\right) \log\left(\sum_{j=1}^{N} \omega_{k-1,j} \cdot \mathcal{N}\left(\underline{z}_k; \hat{\underline{z}}_{k,j}, \mathbf{C}_{k,j}^z\right)\right) \mathrm{d}\underline{z}_k \\
&\geq -\sum_{i=1}^{N} \omega_{k-1,i} \cdot \log\left(\sum_{j=1}^{N} \omega_{k-1,j} \cdot c_{ij}\right)
\end{aligned} \tag{16}$$

with shorthand term $p\left(\underline{z}_k \middle| \underline{a}_k\right) \triangleq p\left(\underline{z}_k \middle| \underline{z}_{0:k-1}, \underline{a}_{0:k}\right)$ and $c_{ij} \triangleq \mathcal{N}\left(\hat{\underline{z}}_{k,i}; \hat{\underline{z}}_{k,j}, \mathbf{C}_{k,i}^z + \mathbf{C}_{k,j}^z\right)$ being the value resulting from integrating over the product of the two Gaussians $\mathcal{N}\left(\underline{z}_k; \hat{\underline{z}}_{k,i}, \mathbf{C}_{k,i}^z\right)$ and $\mathcal{N}\left(\underline{z}_k; \hat{\underline{z}}_{k,j}, \mathbf{C}_{k,j}^z\right)$. The lower bound follows directly from applying Jensen's inequality [4], which allows pulling the logarithm out of the integral. With regard to complexity, the lower bound scales quadratically with the number of object classes N and thus is computationally very efficient as the number of classes is expected to be a few tens.

Utilizing the sifting property of the Kronecker delta (8) and the analytical evaluation of the entropy of a Gaussian distribution, the second conditional entropy term in (13) can written as

$$\begin{aligned}\mathrm{H}\left(\underline{z}_k|x_k,\underline{a}_k\right) &= -\sum_{x_k} p_{k|k-1}\int_{\mathcal{Z}} p\left(\underline{z}_k|x_k,\underline{a}_k\right)\cdot\log p\left(\underline{z}_k|x_k,\underline{a}_k\right)\,\mathrm{d}\underline{z}_k \\ &= -\sum_{i=1}^{N}\omega_{k-1,i}\cdot\underbrace{\int_{\mathcal{Z}}\mathcal{N}\left(\underline{z}_k;\hat{\underline{z}}_{k,i},\mathbf{C}^z_{k,i}\right)\cdot\log\mathcal{N}\left(\underline{z}_k;\hat{\underline{z}}_{k,i},\mathbf{C}^z_{k,i}\right)\,\mathrm{d}\underline{z}_k}_{=-\frac{1}{2}\log\left|2\pi\mathrm{e}\mathbf{C}^z_{k,i}\right|}\,,\end{aligned} \quad (17)$$

where $|.|$ is the determinant of a matrix. Putting (16) and (17) together, the lower bound

$$\bar{\mathrm{I}} \triangleq -\sum_{i=1}^{N}\omega_{k-1,i}\cdot\log\left(\left|2\pi\mathrm{e}\mathbf{C}^z_{k,i}\right|^{\frac{1}{2}}\cdot\sum_{j=1}^{N}\omega_{k-1,j}\cdot c_{ij}\right) \quad (18)$$

of (13) is used in (3) to approximate the mutual information value.

Solving the Optimization Problem

Solving the optimization problem (3) for finding the optimal action or next-best-view $\underline{a}_k^* \in \mathcal{A}$ for the current stage k requires to calculate the maximum of the mutual information and its lower bound (18), respectively. Unfortunately, the optimal action cannot be calculated in closed form. Additionally, the maximum of the mutual information with respect to the actions $\underline{a}_k$ is not unique and thus, the optimization problem is non-convex, which further complicates numerical optimization.

To increase the probability of finding the optimal action or at least to ensure finding an action that is very close to the optimal one, so-called *multi-start optimization* is performed (see e.g. [21]). Here, optimization is repeated from varying initial points. To cover the action space $\mathcal{A}$ uniformly, the initial points form a regular grid on $\mathcal{A}$. For each initial point, the lower bound (18) is maximized by means of the BFGS method [9]. This well known quasi-Newton numerical optimization technique utilizes—in contrast to a classical gradient ascent—an estimate of the Hessian matrix, which results in an increased converge speed towards the sub-optimal solution. The derivation of the required gradient with respect to $\underline{a}_k$ can be found in Appendix A.

5 Simulation Results

The effectiveness of the proposed active object recognition approach is now demonstrated by means of numerical simulations. At first, the setup of all simulations is described. Then, two different object sets are considered for comparison.

5.1 General Simulation Setup

The considered objects are synthetic 3D models rendered by means of the Visualization Toolkit (VTK)[2]. In Figure 2, for each set some of the objects are depicted. For learning and recognition, 100×100 pixel normalized grayscale images are generated from these objects, where zero-mean Gaussian noise with variance 14.7 is added.

1D and 2D features are extracted from the images. In the 1D case, the mean gray value is considered. The eigenspace or principal component decomposition

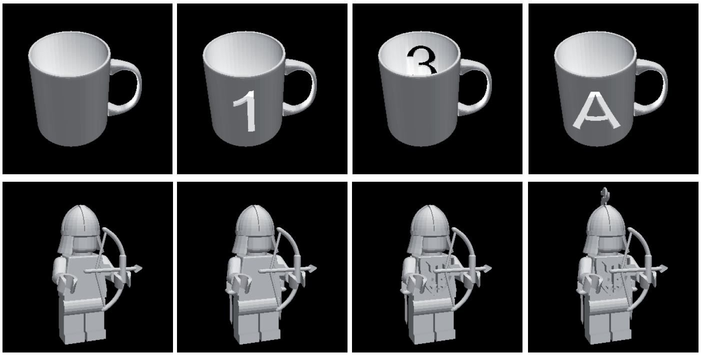


Figure 2: Upper row: cups with different labels. Lower row: toy manikins with different equipment (bow [left] + sword at each hip [second left] + emblem [second right] + crest on the helmet [right]).

2 `http://www.vtk.org/`

approach proposed in [15] is used for extracting 2D features, where the two largest eigenvalues are taken into account. It is important to note that although low-dimensional features are considered here for simplicity, the proposed approach has been derived without any restrictions on the features. Thus, even very complex and high-dimensional features like SIFT can be employed as well.

The simulations focus on actions that change the camera position in one or two dimensions. In the 1D case, the camera moves on a circle that is parallel to the horizontal plane and centered at the object. In the 2D case, the camera position can be varied on a sphere centered at the object. Here, the actions correspond to the azimuth and elevation angles.

To learn the GPs, each dimension of the action space is sampled regularly in 10 decimal degree steps, i.e., for the one-dimensional circular action space, this leads to 36 sample images.

For comparison, the following active object recognition approaches are considered:

Planner	The proposed approach, where 5 and 15 initial points for optimization are exploited for the 1D and 2D action space, respectively.
Grid	An approach similar to [8], where at each stage the action maximizing the mutual information is taken from a finite set. Here, this finite set coincides with the set of initial points of the Planner.
Random	Actions are selected uniformly at random.

All approaches merely differ in the way the next action is selected, while for instance the same GP object models are used and the Bayesian update step is performed identically. Furthermore, Planner and Grid utilize the lower bound (18) of the mutual information.

For each set of objects and each combination of feature and action space, 50 Monte Carlo simulation runs are performed, where the true object is selected uniformly at random. The initial distribution p_0 is uniform. A decision about the object type is made if either the probability of one object estimate exceeds 0.95 or after eight stages.

5.2 Example I: Cups

The first set of objects consists of eight cups that are identical except for the label that is cut through the surface (see Figure 2). The labels of six cups are visible

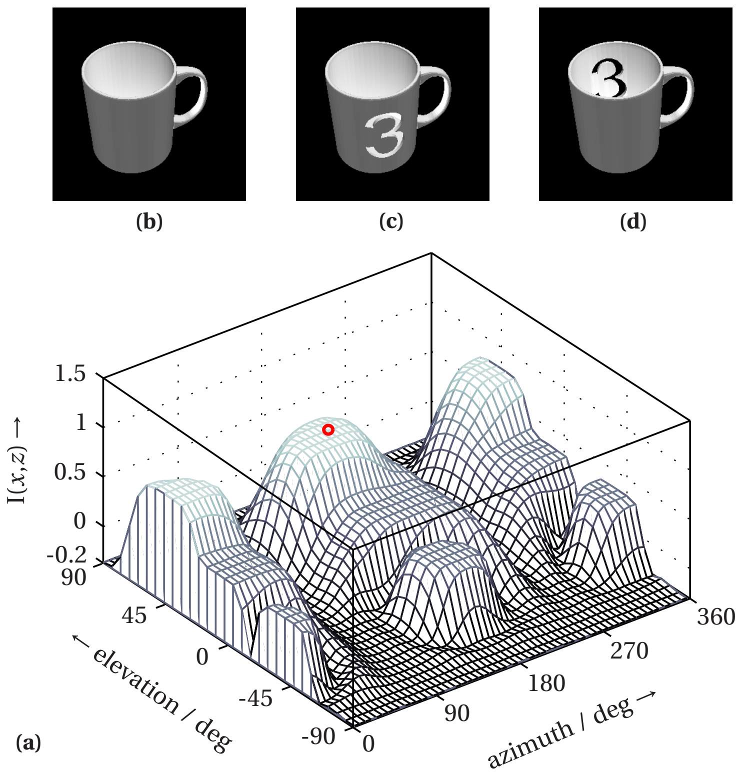


Figure 3: (a) Lower bound of mutual information with optimal view/action (red circle). (b)–(d) View of three of the cups corresponding to the optimal action.

from the same perspective, one is visible from the opposite point of view and one cup is not labeled at all.

For the 2D action space, the mutual information surface for three cups is plotted in Figure 3(a). Here, the optimal action is indicated by the red circle, which corresponds to an elevation angle of approximately 45^o. For this action, the corresponding views on the three cups are depicted in Figure 3(b)–(d). It can be seen that this view facilitates to look inside the cups and thus, allows an easy discrimination of all three cups.

Table 1: Cup recognition. (a) recognition rate in percent, (b) average number of views, (c) average maximum object probability.

Dim.	Planner			Grid			Random		
$\mathcal{A}$ / $\mathcal{Z}$	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
1 / 1	**66**	**6.06**	**0.74**	62	6.1	0.71	50	7.32	0.53
1 / 2	88	**3.08**	**0.97**	74	4.96	0.89	**94**	6.88	0.81
2 / 1	**92**	**2.5**	**0.99**	62	4.1	0.95	76	6.34	0.70
2 / 2	**100**	**1.88**	**0.99**	88	2.5	0.97	68	6.92	0.74

The average values over the 50 simulation runs in terms of recognition rate, number of views, and maximum object probability are listed in Table 1. It can be seen that the Planner performs best with respect to almost any performance indicator. In comparison to Random, the number of stages after which a recognition decision is made is significantly lower. Simultaneously, the certainty in this decision is much higher as the average maximum object probability indicates. The performance of the Grid approach is often close to the proposed approach. But the significantly lower number of views of the Planner shows the benefits of performing a continuous optimization for next-best-view planning. In contrast to both Grid and Random, the proposed Planner can take advantage of an increasing feature and action dimension, i.e., with an increasing dimension the recognition rate increases as well and the number of views decreases.

A high object probability not necessarily coincides with the best recognition rate as seen in the case of the 1D action space and 2D feature space. While Random merely relies on the GP object models for inference, Grid and Planner additionally use the models for decision making. Thus, a bootstrapping effect can cause the decision maker to get stuck in a repetitive pattern. The quality of the GP models is essential for the recognition process and thus, under- and over-fitting require special attention.

5.3 Example II: Toy Manikins

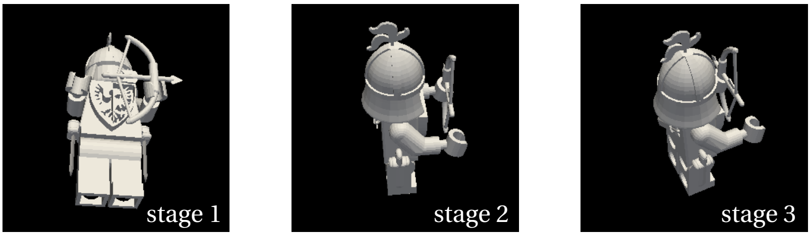
The second set of objects used for simulation consists of nine toy manikins that carry different pieces of equipment (bow, quiver, sword, emblem, helmet, and crest—see Figure 2). Compared to the cups, the toy manikins have much more details and the differences between each object are more subtle. In Figure 4, the decision making of the Planner is shown for the 2D action space and 1D feature

Table 2: Toy manikin recognition. (a) recognition rate in percent, (b) average number of views, (c) average maximum object probability.

Dim.	Planner			Grid			Random		
$\mathcal{A}$ / $\mathcal{Z}$	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
1 / 1	**68**	**2.58**	**0.98**	64	5.32	0.93	**68**	2.7	**0.98**
1 / 2	**90**	**4.9**	**0.95**	72	5.78	0.87	82	7.3	0.83
2 / 1	**100**	**2.34**	**0.99**	90	3.12	0.97	92	5.66	0.92
2 / 2	**100**	**1.56**	**0.99**	88	2.96	0.96	90	5.56	0.91

space. The first view reveals most of the equipment items in such a way that the differences to other manikins are significant regarding the rather simple mean gray value feature. The next two views highlight the sword as well as the crest and thus, help to distinguish the manikin from those without these items.

In Table 2, the same performance indicators as in the cup scenario are listed. While Planner and Random perform nearly identical for the 1D action and feature space, for higher dimensions, the Planner clearly is the best object recognition algorithm. Interestingly, all algorithms perform better than in the cup scenario. This is mainly due to the more details of the manikins and thus, much more views exist that allow discriminating different manikins from each other.



stage	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	H(x)
0	.111	.111	.111	.111	.111	.111	.111	.111	.111	1.0
1	.149	**.383**	.000	.000	.013	.000	.259	.188	.009	.643
2	.000	.000	.000	.000	.000	.000	.322	**.674**	.004	.298
3	.000	.000	.000	.000	.000	.000	.000	**.998**	.001	.006

Figure 4: Recognition of object x_8 via proposed approach: selected views (top) and corresponding distributions $p_{k|k}$ with entropy (bottom).

6 Conclusion and Future Work

The proposed approach exploits Gaussian process regression for object recognition. Thanks to the probabilistic nature of the GPs, the variability in image acquisition—resulting for instance from changing light conditions, occlusion, or changing background—is incorporated and robust object models over continuous action spaces are generated from few training samples. In combination with recursive Bayesian estimation and optimizing the next view, this approach allows a reliable recognition even with low dimensional and thus, rather simple image features. The proposed approach can be applied in various recognition scenarios as it is not restricted to specific features, action spaces, or kernel functions.

Future work is devoted to applying the proposed approach in a real-world experiment, where a camera is mounted on a six degree-of-freedom robotic arm. By this means, the camera can be moved either in 2D or 3D space, as it is done in the simulations. So far, a recognition or classification problem has been considered. It is also intended to combine classification with pose estimation, i.e., to simultaneously identify the object class as well as its orientation and location in space. An improved recognition rate is expected—especially in situations with for instance time or kinematic constraints [11]—if actions are planned is a non-myopic fashion, i.e., for more than one stage ahead. Furthermore, learning and planning are currently decoupled. By means of reinforcement learning techniques [22], both steps could be performed simultaneously.

A Gradient

Next-best-view planning requires the calculation of the gradient of the lower bound (18) of the mutual information with respect to the action $\underline{a} \in \mathcal{A}$. An analytical expression of the gradient is derived in the following. The stage index k is omitted for improved readability. By rewriting the lower bound $\bar{\mathrm{I}} = -\sum_{i=1}^{N} \omega_i \cdot \log f_i$ with $f_i = |2\pi \mathrm{e} \mathbf{C}_i^z|^{\frac{1}{2}} \cdot \sum_{j=1}^{N} \omega_j \cdot c_{ij}$, where $c_{ij} = \mathcal{N}\left(\underline{\hat{z}}_i; \underline{\hat{z}}_j, \mathbf{C}_{ij}\right)$ and $\mathbf{C}_{ij} \triangleq \mathbf{C}_i^z + \mathbf{C}_j^z$, its partial derivative with respect to action $\underline{a}$ can be written as

$$\frac{\partial \bar{\mathrm{I}}}{\partial \underline{a}} = -\sum_{i=1}^{M} \frac{\omega_i}{f_i} \cdot \frac{\partial f_i}{\partial \underline{a}} \, . \tag{19}$$

To solve (19), the differential identities

$$\partial|\mathbf{X}| = |\mathbf{X}| \cdot \mathrm{Tr}\left(\mathbf{X}^{-1} \cdot \partial\mathbf{X}\right) , \tag{20}$$

$$\partial\mathbf{X}^{-1} = -\mathbf{X}^{-1} \cdot \partial\mathbf{X} \cdot \mathbf{X}^{-1} \tag{21}$$

are required (see [17]), with Tr(.) being the matrix trace. Applying the chain rule and (20), the derivative of f_i is

$$\frac{\partial f_i}{\partial \underline{a}} = \left|2\pi \mathrm{e} \mathbf{C}_i^z\right|^{\frac{1}{2}} \cdot \sum_{j=1}^{N} \omega_j \cdot \left[\frac{c_{ij}}{2} \mathrm{Tr}\left(\left(\mathbf{C}_i^z\right)^{-1} \frac{\partial \mathbf{C}_i^z}{\partial \underline{a}} \right) + \frac{\partial c_{ij}}{\partial \underline{a}} \right]$$

with

$$\frac{\partial \mathbf{C}_i^z}{\partial a_l} = \mathrm{diag}\left(\frac{\partial (\sigma_i^1)^2}{\partial a_l}, \ldots, \frac{\partial \left(\sigma_i^{n_z}\right)^2}{\partial a_l} \right)$$

for each dimension $l = 1, \ldots, n_a$ of action $\underline{a}$, where the variances $\left(\sigma_i^e\right)^2$, $e = 1, \ldots, n_z$ correspond to (6), and

$$\frac{\partial c_{ij}}{\partial \underline{a}} = \frac{\partial}{\partial \underline{a}} \left(|2\pi \mathbf{C}_{ij}|^{-\frac{1}{2}} \cdot g_{ij} \right) \tag{22}$$

with $g_{ij} \triangleq \exp\left(-\frac{1}{2} \cdot \hat{\underline{z}}_{ij}^{\mathrm{T}} \cdot \mathbf{C}_{ij}^{-1} \cdot \hat{\underline{z}}_{ij}\right)$ and $\hat{\underline{z}}_{ij} \triangleq \hat{\underline{z}}_i - \hat{\underline{z}}_j$. Applying (20) and (21) on (22) yields

$$\begin{aligned} \frac{\partial c_{ij}}{\partial \underline{a}} &= \frac{\partial |2\pi \mathbf{C}_{ij}|^{-\frac{1}{2}}}{\partial \underline{a}} \cdot g_{ij} + |2\pi \mathbf{C}_{ij}|^{-\frac{1}{2}} \cdot \frac{\partial g_{ij}}{\partial \underline{a}} \\ &= -\frac{g_{ij}}{2} |2\pi \mathbf{C}_{ij}|^{-\frac{1}{2}} \mathrm{Tr}\left(\mathbf{C}_{ij}^{-1} \cdot \frac{\partial \mathbf{C}_{ij}}{\partial \underline{a}} \right) - \frac{c_{ij}}{2} \cdot \\ &\qquad \left(2 \left(\frac{\partial \hat{\underline{z}}_{ij}}{\partial \underline{a}} \right)^{\mathrm{T}} \cdot \mathbf{C}_{ij}^{-1} \cdot \hat{\underline{z}}_{ij} - \hat{\underline{z}}_{ij}^{\mathrm{T}} \cdot \mathbf{C}_{ij}^{-1} \cdot \frac{\partial \mathbf{C}_{ij}}{\partial \underline{a}} \cdot \mathbf{C}_{ij}^{-1} \cdot \hat{\underline{z}}_{ij} \right) \end{aligned} \tag{23}$$

The remaining derivatives $\partial \hat{\underline{z}}_{ij} / \partial \underline{a}$ and $\partial \mathbf{C}_{ij} / \partial \underline{a}$ can easily be decomposed into the derivatives of the respective summands. Furthermore, calculating the derivatives can be performed dimension-wise. Thus, the remaining partial derivatives $\partial / \partial \underline{a} \hat{z}_i^e$

and $\partial/\partial\underline{a}\left(\sigma_i^e\right)^2$ for each dimension $e = 1,\ldots,n_z$, correspond to the derivatives

$$\begin{aligned}
\frac{\partial\hat{h}}{\partial\underline{a}} &= \left(\frac{\partial}{\partial\underline{a}}\underline{k}_*\right)^{\mathrm{T}} \left(\mathbf{K}+\sigma^2\mathbf{I}\right)^{-1} \underline{y}\,, \\
\frac{\partial\sigma_h^2}{\partial\underline{a}} &= \underbrace{\frac{\partial}{\partial\underline{a}}k_{**}}_{=0} - 2\left(\frac{\partial}{\partial\underline{a}}\underline{k}_*\right)^{\mathrm{T}} \left(\mathbf{K}+\sigma^2\mathbf{I}\right)^{-1} \underline{k}_*
\end{aligned}$$

of (5) and (6) with respect to $\underline{a}$, respectively, with the matrix

$$\frac{\partial\underline{k}_*}{\partial\underline{a}} = \left[\frac{\partial}{\partial\underline{a}}k(\underline{a}_1,\underline{a}),\ldots,\frac{\partial}{\partial\underline{a}}k(\underline{a}_n,\underline{a})\right]\,. \tag{24}$$

Here, $\underline{a}_1,\ldots,\underline{a}_n$ are the training inputs. The derivative of the SE kernel in (24) for $i = 1,\ldots,n$ is given by

$$\frac{\partial k(\underline{a}_i,\underline{a})}{\partial\underline{a}} = \alpha^2 \cdot \Lambda^{-1} \cdot \left(\underline{a}_i - \underline{a}\right) \cdot \exp\left(-\tfrac{1}{2}\left(\underline{a}_i - \underline{a}\right)^{\mathrm{T}} \Lambda^{-1} \left(\underline{a}_i - \underline{a}\right)\right)\,.$$

References

[1] Hervé Abdi and Lynne J. Williams. Principal Component Analysis. In *Wiley Interdisciplinary Reviews: Computational Statistics*, volume 2, pages 433–459. Wiley, New York, July 2010.

[2] Hermann Borotschnig, Lucas Paletta, Manfred Prantl, and Axel Pinz. Appearance-Based Active Object Recognition. *Image and Vision Computing*, 18:715–727, 2000.

[3] Philipp Boyle and Marcus Frean. Dependent Gaussian Processes. In Lawrence K. Saul, Yair Weiss, and Leon Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 217–224. MIT Press, 2005.

[4] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.

[5] Guido de Croon, Ida G. Sprinkhuizen-Kuyper, and Eric O. Postma. Comparing Active Vision Models. *Image and Vision Computing*, 27:374–384, March 2009.

[6] Frank Deinzer, Joachim Denzler, and Heinrich Niemann. Viewpoint Selection - Planning Optimal Sequences of Views for Object Recognition. In *In International Conference on Computer Vision*, pages 65–73. Springer, 2003.

[7] Marc P. Deisenroth, Marco F. Huber, and Uwe D. Hanebeck. Analytic Moment-based Gaussian Process Filtering. In *26th International Conference on Machine Learning (ICML)*, pages 225–232, Montreal, Canada, June 2009.

[8] Joachim Denzler and Christopher M. Brown. Information Theoretic Sensor Data Selection for Active Object Recognition and State Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):145–157, February 2002.

[9] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2nd edition, May 2000.

[10] Jacob Goldberger, Shiri Gordon, and Hayit Greenspan. An Efficient Image Similarity Measure based on Approximations of KL-Divergence Between Two Gaussian Mixtures. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, volume 1, pages 487–493, October 2003.

[11] Marco Huber. *Probabilistic Framework for Sensor Management*. PhD thesis, Universität Karlsruhe (TH), April 2009.

[12] Marco F. Huber, Tim Bailey, Hugh Durrant-Whyte, and Uwe D. Hanebeck. On Entropy Approximation for Gaussian Mixture Random Vectors. In *Proceedings of the 2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 181–188, Seoul, Republic of Korea, August 2008.

[13] Catherine Laporte and Tal Arbel. Efficient Discriminant Viewpoint Selection for Active Bayesian Recognition. *International Journal of Computer Vision*, 68:267–287, July 2006.

[14] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the 7th International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157, Kerkyra, Greece, September 1999.

[15] Hiroshi Murase and Shree K. Nayar. Visual learning and recognition of 3-D objects from appearance. *International Journal Computer Vision*, 14:5–24, January 1995.

[16] Lucas Paletta and Axel Pinz. Active Object Recognition By View Integration and Reinforcement Learning. *Robotics and Autonomous Systems*, 31:71–86, 2000.

[17] Kaare Brandt Petersen and Michael Syskind Pedersen. The Matrix Cookbook. Online: http://www2.imm.dtu.dk/pubdb/p.php?3274, November 2008.

[18] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[19] Sumantra Dutta Roy, Santanu Chaudhury, and Subhashis Banerjee. Active recognition through next view planning: a survey. *Pattern Recognition*, 37(3):429–446, March 2004.

[20] Dan Simon. *Optimal State Estimation: Kalman, H_∞, and Nonlinear Approaches*. Wiley & Sons, 1st edition, 2006.

[21] Francisco J. Solis and Roger J-B. Wets. Minimization by Random Search Techniques. *Mathematics of Operations Research*, 6(1):19–30, February 1981.

[22] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[23] Richard Szeliski. *Computer Vision: Algorithms and Applications*, chapter 14 – Recognition. Springer London, 2010.

[24] Raquel Urtasun and Trevor Darrell. Discriminative Gaussian Process Latent Variable Model for Classification. In *Proceedings ot the 24th International Conference on Machine Learning (ICML)*, Corvallis, OR, 2007.