Fail-Safe Vehicle Pose Estimation in Lane-Level Maps Using Pose Graph Optimization

Zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
von der KIT-Fakultät für Maschinenbau
des Karlsruher Instituts für Technologie (KIT)

genehmigte
Dissertation
von

M.Sc. Maximilian Harr

Tag der mündlichen Prüfung:	28. Juni 2019
Hauptreferent:	Prof. Dr.-Ing. Christoph Stiller
Korreferent:	Prof. Dr.-Ing. Klaus Dietmayer

Vorwort und Danksagungen

Die vorliegende Arbeit enstand während meiner Tätigkeit als Doktorand im Bereich EE Advanced Technology der Opel Automobile GmbH unter der Betreuung von Prof. Dr. Christoph Stiller, Leiter des Instituts für Mess- und Regelungstechnik (MRT) am Karlsruher Institut für Technologie (KIT).

Zunächst möchte ich mich herzlich bei Prof. Dr. Christoph Stiller für die durchgehend professionelle Betreuung, die anregenden, fachlichen Gespräche und die Übernahme des Hauptreferates bedanken. Prof. Dr. Klaus Dietmayer danke ich für den wissenschaftlichen Austausch, die Übernahme des Korreferats und das damit verbundene Interesse an der vorliegenden Arbeit.
Ein besonderer Dank gilt meinem Betreuer bei Opel, Dr. Nikolas Wagner für die Förderung meiner fachlichen Arbeit bei Opel sowie der durchweg guten Beziehung die über das Berufliche hinausging. Dr. Philipp Heidenreich, Dr. Andreas Andrä undf Dr. Stefan Berger danke ich für die konstruktive Hilfestellung und Korrektur der vorliegenden Arbeit. Für die fachlichen Gespräche, wertvollen Diskussionen auf den Sommerseminaren und der Bereitstellung und den Support eines hervorragenden Software-Frameworks danke ich den Doktoranden des MRT, insbesondere Claudio Bandera. Andre-Marcel Hellmund, Johannes Janosovits und Sascha Wirges danke ich für die durchweg sehr gute Zusammenarbeit in unserem Projekt und die Bereitstellungen der Bildverarbeitungsalgorithmen. David Augustin und Jeremias Schucker, danke ich für die jahrelange Zusammenarbeit, die gemeinsame Inbetriebnahme des Versuchsträgers, die langen Testtage und die gegenseite Hilfe und Stärkung. Insbesondere vor dem Ko-HAF Abschlussevent. Vor allem aber für die schöne und lustige gemeinsame Zeit die wir zusammen mit und bei Opel hatten.
Ein großer Dank geht an die Konzeptwerkstatt der Opel Automobile GmbH für die Hilfestellung bei der Inbetriebnahme und den Aufbau des Versuchsträgers sowie der Umsetzung der unzähligen Änderungswünsche. Ganz besonders möchte ich Michael Dahlke danken.
Unserer Sekräterin Claudia Holschuh-Deja danke ich für die jahrelange Zusammenarbeit und dem ganzen Ko-HAF Team für die interessanten Treffen, Telefonkonferenzen und gemeinsamen Konsortialtests, insbesondere im Arbeitspaket 2 "Lokalisierung und statisches Umfeldmodell".
Der Streckenleitung des Opel Testcenter Dudenhofen danke ich für die Geduld

während der erschöpfend langen Testfahrten und Jens Guder für die Unterstützung bei der Inbetriebnahme der Ground Truth auf dem Opel Testcenter. Zu guter letzt gebührt meinen Eltern für die jahrelange Förderung und Stärkung in meinem Leben, der größte Dank. Ihr seid der Ruhepol in meinem Leben wo ich Kraft für neue Aufgaben finde, dafür danke ich euch. Euch widme ich diese Arbeit.

Karlsruhe, im Februar 2019 *Maximilian Harr*

Kurzfassung

Die hochgenaue Posenschätzung autonomer Fahrzeuge sowohl in HD-Karten als auch spurrelativ ist unerlässlich um eine sichere Fahrzeugführung zu gewährleisten. Für die Serienfertigung wird aus Kosten- und Platzgründen bewusst auf hochgenaue, teure Einzelsensorik verzichtet und stattdessen auf eine Vielzahl von Sensoren, die neben der Posenschätzung auch von anderen Modulen verwendet werden können, zurückgegriffen. Im Fokus dieser Arbeit steht die Unsicherheitsschätzung, Bewertung und Fusion dieser Sensordaten.
Die Optimierung von Posengraphen zur Fusion von Sensordaten zeichnet sich, im Gegensatz zu klassischen Filterverfahren, wie Kalman oder Partikelfilter, durch seine Robustheit gegenüber Fehlmessungen und der Flexibilität in der Modellierung aus. Die Optimierung eines Posengraphen wurde erstmalig auf mobilen Roboterplattformen zur Lösung sogenannter SLAM-Probleme angewendet [83]. Diese Verfahren wurden immer weiter entwickelt und im Speziellen auch zur rein kamerabasierten Lokalisierung autonomer Fahrzeuge in 3D-Punktwolken erfolgreich demonstriert [97]. Für die Entwicklung und Freigabe sicherheitsrelevanter Systeme nach ISO 26262 wird neben der Genauigkeit jedoch auch eine Aussage über die Qualität und Ausfallsicherheit dieser Systeme gefordert.
Diese Arbeit befasst sich, neben der Schätzung der karten- und spurrelativen Pose, auch mit der Schätzung der Posenunsicherheit und der Integrität der Sensordaten zueinander. Auf Grundlage dieser Arbeit wird eine Abschätzung der Ausfallsicherheit des Lokalisierungsmoduls ermöglicht. Motiviert durch das Projekt Ko-HAF werden zur Lokalisierung in HD-Karten lediglich Spurmarkierungen verwendet. Die speichereffiziente Darstellung dieser Karten ermöglicht eine hochfrequente Aktualisierung der Karteninhalte durch eine Fahrzeugflotte.
Der vorgestellte Ansatz wurde prototypisch auf einem Opel Insignia umgesetzt. Der Testträger wurde um eine Front- und Heckkamera sowie einen GNSS-Empfänger erweitert. Zunächst werden die Schätzung der karten- und spurrelativen Fahrzeugpose, der GNSS-Signalauswertung sowie der Bewegungsschätzung des Fahrzeugs vorgestellt. Durch einen Vergleich der Schätzungen zueinander werden die Unsicherheiten der einzelnen Module berechnet. Das Lokalisierungsproblem wird dann durch einen Optimierer gelöst. Mithilfe der berechneten Unsicherheiten wird in einem nachgelagerten Schritt eine Bewertung der einzelnen Module durchge-

führt. Zur Bewertung des Ansatzes wurden sowohl hochdynamische Manöver auf einer Teststrecke als auch Fahrten auf öffentlichen Autobahnen ausgewertet.

Abstract

Obtaining frequent and high-precision pose estimates of autonomous vehicles in HD maps as well as within lanes is essential to ensure reliable vehicle maneuvering. For reasons of cost and space, expensive high-precision sensors are deliberately avoided in series production. Instead, a large number of economic sensors are implemented that can be used by other modules as well. Estimating uncertainties, evaluation, and fusion of sensor measurements to guarantee fail-safe operation will be the focus of this thesis.

The use of pose graph optimization techniques to fuse sensor data is characterized, in contrast to classical filter methods such as Kalman or particle filtering, by its robustness against erroneous measurements and its modeling flexibility. Pose graph optimization techniques were first studied on mobile robot platforms to solve the SLAM problem [83]. These methods have been further developed and successfully demonstrated for purely camera-based localization of autonomous vehicles in 3D point clouds [97]. To develop and release safety-relevant systems according to the ISO 26262 standard, a statement of estimation quality and reliability is indispensable.

In addition of map and lane-relative pose estimation, this thesis further surveys the estimation of pose estimate uncertainties and integrity within sensor data. Based on this work, a conclusive statement on the reliability of the localization module can be derived. Motivated by the Ko-HAF project, the presented localization approach merely relies on lane markers of mapped highway sections. Their memory-efficient, parametric representation enables a high-frequency update of map contents through a vehicle fleet.

The presented approach was investigated using an Opel Insignia as experimental vehicle. Its sensor setup was extended by a front and rear camera and a GNSS receiver. First, the estimation of map and lane-relative vehicle poses, the GNSS signal evaluation as well as the vehicle's odometry estimation are presented. Uncertainties of the estimators are calculated by comparing their estimates. Using these uncertainties, an evaluation of the estimator's integrity is carried out. The localization problem is solved by using optimization. The approach is evaluated on both highly dynamic maneuvers on a test track and test drives on public highways.

Contents

Notation and Symbols

This chapter introduces notations and symbols used throughout this work. If not declared otherwise, all physical quantities are in SI units. If symbols have more than one meaning either the context itself or a distinct statement dissolves the ambiguity.

Terminology

GNSS - Global Navigation Satellite System

System for pose estimation through satellite signals. GPS is the US implementation of a GNSS and is commonly used as pars pro toto, but GNSS receivers also use Beidou (China), GLONASS (Russia) and Galileo (Europe).

HD-Maps

Navigation maps only contain topological information necessary for routing and basic ADAS. HD or UHD maps contain more detailed geological and geometrical information such as lane markers and reflector poles. If used for fully automated driving these maps are also referred to as FAD maps.

Pose

A pose denotes a position and orientation pair.

Pose Estimation Versus Localization

Robot localization denotes the estimation of the robot's current pose within a reference frame or map. Pose estimation can be used as another term for localization, but is also used when estimating the pose of robot parts, other robots or landmarks as well as estimating a robot's set of poses within a time frame.

Graph, Pose Graph, and Pose Graph Optimization

A graph is a structure, which connect vertices through edges. A pose graph is a specific implementation of a graph, where all vertices are poses. In pose graph optimization, some or all graph vertices are optimized to minimize an objective function, represented by the graph's edges.

Measurement Versus Estimation

Measurements are the result of measuring physical quantities or counting values directly or by applying additional simple arithmetics. Estimation, on the other side, will be used whenever predictions of quantities that may not be observable, are made from single or multiple measurements using refined algorithms.

Odometer, Dead Reckoning and Odometry

Odometers are sensors that measure the distance traveled, usually by counting wheel rotations. Dead Reckoning is a technique to compute the current pose by advancing priorly determined poses using an estimated course and speed from an odometer. Odometry is the estimation of pose deltas, which includes dead reckoning, but also covers other techniques such visual odometry.

Roads and Lanes

See Fig.0.1

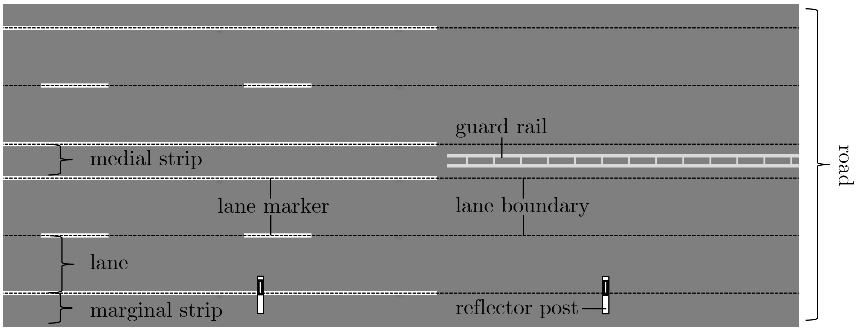


Figure 0.1: **Road Nomenclature**.

Fonts

Scalars	Roman case	a, A, b, B, c, C, ...
Vectors	Bold lower case	$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$, ...
Matrices	Bold upper case	$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$, ...
Sets	Calligraphic upper case	$\mathcal{A}, \mathcal{B}, \mathcal{C}$, ...
Distributions	Calligraphic upper case	$\mathcal{U}, \mathcal{N}$, ...
Set of Numbers	Double-struck upper case	$\mathbb{N}, \mathbb{R}, \mathbb{Z}$, ...

Vectors, arrays and matrices

Vector and array indices

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$$

Matrix indices

$$\boldsymbol{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}$$

Matrix horizontal composition
If $A_1 \in \mathbb{R}^{n\times k}$ and $A_2 \in \mathbb{R}^{n\times l}$ composing A_1 and A_2 horizontally results in a matrix $A \in \mathbb{R}^{n\times(k+l)}$ and is denoted by

$$A = [A_1, A_2] \tag{0.1}$$

Matrix vertical composition
If $A_1 \in \mathbb{R}^{k\times n}$ and $A_2 \in \mathbb{R}^{l\times n}$ composing A_1 and A_2 vertically results in a matrix $A \in \mathbb{R}^{(k+l)\times n}$ and is denoted by

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \tag{0.2}$$

Covariance matrix
The covariance matrix of a multivariate, normally distributed random variable X is represented by

$$\boldsymbol{\Sigma}_x = \boldsymbol{\Sigma}_{xx} = \begin{bmatrix} \sigma_1^2 & \sigma_{1,2} & \cdots & \sigma_{1,n} \\ \sigma_{2,1} & \sigma_2^2 & \cdots & \sigma_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n,1} & \sigma_{n,2} & \cdots & \sigma_n^2 \end{bmatrix}$$

And for the 3D case where x, y, z denote the 3D spatial directions

$$\boldsymbol{\Sigma}_x = \boldsymbol{\Sigma}_{xx} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_z^2 \end{bmatrix}$$

The correlation matrices of two multivariate, normally distributed random variables X and Y is represented by $\mathbf{\Sigma}_{xy}$ and $\mathbf{\Sigma}_{yx}$.

Acronyms

AD	Automatic Differentiation
ADAS	Advanced Driver Assistance Systems
ASIL	Automotive Safety Integrity Level
BF	Bayes Filter
BFGS	Broyden-Fletcher-Goldfarb-Shannon
CAN	Controller Area Network
CI	Covariance Intersection
CLT	Central Limit Theorem
COG	center of gravity
DFP	Davidon-Fletcher-Powell
DGPS	Differential GPS
DOF	Degree Of Freedom
DOP	Dilution Of Precision
ECEF	Earth-Centered, Earth-Fixed
ECI	Earth-Centered Inertial
EKF	Extended Kalman Filter
FDE	Fault Detection and Exclusion
FDI	Fault Detection and Identification
GMM	Gaussian Mixture Model
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HMM	Hidden Markov Model
ICP	Iterative Closest Point
ICR	Instant Center of Rotation
IMM	Interacting Multiple Model
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IPM	Inverse Perspective Mapping
KF	Kalman Filter
KKT	Karush-Kuhn-Tucker

LM	Levenberg-Marquardt
LOS	Line-Of-Sight
LSQ	Least-Squares
MHT	Multi Hypothesis Tracking
MMAE	Multiple Model Adaptive Estimation
MND	Multivariate Normal Distribution
NIS	Normalized Innovation Squared
NLP	Nonlinear Programming
NLSQ	Nonlinear Least Squares
NNS	Nearest Neighbor Search
NTRIP	Networked Transport of RTCM via Internet Protocol
PDF	Probability Density Function
PF	Particle Filter
PRN	Pseudo Random Noise
RAIM	Receiver Autonomous Integrity Monitoring
RANSAC	Random Sample Consensus
RMSE	Root Mean Squared Error
RPM	Robust Point Matching
RTCM	Radio Technical Commission for Maritime Services
SAS	Steering Angle Sensor
SCT	Statistical Consistency Test
SfM	Structure from Motion
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SMC	Sequential Monte Carlo
SPP	Single Point Positioning
SR1	Symmetric-Rank-1
STD	Standard Deviation
SURF	Speeded Up Robust Features
UKF	Unscented Kalman Filter
UTM	Universal Transverse Mercator
WGS84	World Geodetic System 1984
WSS	Wheel Speed Sensor

Accents and Scripts

$\boldsymbol{a}^{cam,f}$	Vector in front camera coordinates
$\boldsymbol{a}^{cam,r}$	Vector in rear camera coordinates

$\boldsymbol{a}^{cam}$	Vector in front or rear camera coordinates
$\boldsymbol{a}^{icr}$	Vector in ICR coordinates
$\boldsymbol{a}^{v}$	Vector in vehicle coordinates
$\boldsymbol{a}^{w}$	Vector in UTM coordinates
$a_{i,j}$	Matrix index iteration
a_i	Vector / array index iteration
$()_v^w$	Rotation / translation from v to w coordinates

Mathematical Notations

$\boldsymbol{a} \times \boldsymbol{b}$	Cross product of vector $\boldsymbol{a}$ and $\boldsymbol{b}$
$det(\boldsymbol{A})$	Determinant of matrix $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	Inverse matrix
$\lVert\boldsymbol{A}\rVert$	Norm of matrix of $\boldsymbol{A}$
$\boldsymbol{a} \otimes \boldsymbol{b}$	Rotating a 3D vector $\boldsymbol{b}$ around a 4D quaternion $\boldsymbol{a}$
$\boldsymbol{a} \cdot \boldsymbol{b}$	Scalar product of vector $\boldsymbol{a}$ and $\boldsymbol{b}$
$\boldsymbol{A}^T, \boldsymbol{a}^T$	Transposed matrix / vector
$\bar{\boldsymbol{A}}, \bar{\boldsymbol{a}}, \bar{a}$	Complex conjugate matrix/vector/scalar
$\dot{\boldsymbol{a}}$	Derivation
$\hat{\boldsymbol{a}}$	Initial value or estimate
$\mathcal{A} \cap \mathcal{B}$	Intersection of set $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{A} \cup \mathcal{B}$	Union of set $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{A} \setminus \mathcal{B}$	Difference between set $\mathcal{A}$ and $\mathcal{B}$
$\mathcal{A} \in \mathcal{B}$	$\mathcal{A}$ is member of $\mathcal{B}$
$\mathcal{A} \subseteq \mathcal{B}$	Set $\mathcal{A}$ is subset of $\mathcal{B}$

Latin Letters

D_r	Doppler frequency shift in Hz
El_i	Elevation angle of GNSS satellite
F	Force in N
F_s	Satellite system error factor
I_i	Ionospheric delay in m
J	Moment of inertia in kg m^2
M_gnss	3D GNSS pose estimate
M_map	3D map matching pose estimate
P	Pose, position $\boldsymbol{p}$ and orientation $\boldsymbol{q}$)

P_r	Pseudorange in m
T_i	Tropospheric delay in m
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{R}(\cdot)$	Converts rotation vector to a rotation matrix
$\boldsymbol{R},\boldsymbol{r}$	Rotation matrix, rotation vector
$\boldsymbol{T},\boldsymbol{t}$	Translation matrix, translation vector
$\boldsymbol{e}_i^w$	Line-Of-Sight-vector between GNSS satellite and receiver
$\boldsymbol{e}_{..}$	Unit vector x,y,z of coordinate system
$\boldsymbol{m}_i$	3D lane marker sample point
$\boldsymbol{p}$	3D-Position
$\boldsymbol{p}_\mathrm{gnss}$	3D GNSS position estimate
$\boldsymbol{p}_\mathrm{map}$	3D map matching position estimate
$\boldsymbol{p}_\mathrm{odo}$	3D odometry position estimate
$\boldsymbol{p}_\mathrm{rs}$	Vector between satellite and receiver position in m
$\boldsymbol{p}_\mathrm{r}$	Receiver position in m
$\boldsymbol{p}_\mathrm{s}$	Satellite position in m
$\boldsymbol{q}$	Quaternion
$\boldsymbol{q}_\mathrm{gnss}$	GNSS quaternion estimate
$\boldsymbol{q}_\mathrm{map}$	Map matching quaternion estimate
$\boldsymbol{q}_\mathrm{odo}$	Odometry quaternion estimate
$\boldsymbol{r}$	Light ray
$\boldsymbol{r}_\mathrm{p}$	Light ray projected on ground plane
a	Acceleration
c	Speed of light in m/s
d	Distance in m
$dT_{\mathrm{c},i}$	GNSS satellite clock delay in s
$d\dot{T}_{\mathrm{c},i}$	GNSS satellite clock drift in s/s
$d\dot{t}_\mathrm{c}$	GNSS receiver clock drift in s/s
$d_{..}$	Rear-right/left (rr/rl) and front-right/left (fr/fl) wheel diameter
dt_c	GNSS receiver clock delay in s
f	Frequency in Hz
$f(\cdot)$	Residual
l_ax	Axle track in m
l_f	Distance of front vehicle axle center to COG in m
l_r	Distance of rear vehicle axle center to COG in m

t	Time in s
t_r	Receiving time stamp in s
t_s	Sending time stamp in s
v	Velocity in m/s
$w_{..}$	Wheel ticks of rear-right/left (rr/rl) and front-right/left (fr/fl) encoder
$\mathbb{N}$	Natural numbers
$\mathbb{R}$	Real numbers
$\mathbb{Z}$	Integer numbers
$\mathcal{I}_\mathrm{gnss}$	Information matrix of GNSS pose estimate uncertainty
$\mathcal{I}_{lane}$	Information matrix of subsequent lane matching uncertainty
$\mathcal{I}_\mathrm{map}$	Information matrix of map matching pose estimate uncertainty
$\mathcal{I}_\mathrm{odo}$	Information matrix of odometry measurement uncertainty
$\mathcal{I}$	Information matrix
$\mathcal{J}_\mathrm{gnss}$	Set of GNSS pose estimate indices
$\mathcal{J}_\mathrm{map}$	Set of map matching pose estimate indices
$\mathcal{J}$	Set of indices
$\mathcal{M}_\mathrm{gnss}$	Set of GNSS pose estimates
$\mathcal{M}_\mathrm{map}$	Set of map matching pose estimates
$\mathcal{M}$	Set of pose estimates
$\mathcal{M}_\mathrm{odo}$	Set of odometry delta pose estimates
$\mathcal{P}_\mathrm{gnss}$	Set of optimized GNSS poses
$\mathcal{P}_\mathrm{map}$	Set of optimized map matching poses
$\mathcal{P}$	Set of optimized poses
$\mathcal{S}_\mathrm{map}$	Set of sampled lane marker points from the digital map
$\mathcal{S}_\mathrm{marker}$	Set of detected lane markers
$\mathcal{S}_{\mathrm{ray},p}$	Set of detected lane markers projected on a ground plane
$\mathcal{S}_\mathrm{ray}$	Set of detected lane markers represented by rays

Greek Letters

β	Slip angle in rad
χ^2	Chi-squared threshold
δ	Steering angle in rad
δ_f	Front wheel steering angle in rad
δ_r	Rear wheel steering angle in rad
ε_pr	Pseudorange measurement error in m

λ	WGS84 Longitude in deg
λ_i	Wavelength in m
$\mathbf{\Sigma}$	Covariance matrix
$\mathbf{\Sigma}_p$	Position covariance matrix
ω_{e}	Earth angular rate in rad/s
ϕ	WGS84 Latitude in deg
ϕ	Roll Angle in rad
ψ	Yaw Angle in rad
$\rho(\cdot)$	Loss function
ρ_i	Geometric range in m
σ	Sample standard deviation
σ_{bias}	PRN code bias error standard deviation in m
σ_{eph}	Ephemeris error standard deviation in m
σ_{ion}	Ionospheric delay error standard deviation in m
σ_{trop}	Tropospheric delay erro standard deviation in m
θ	Pitch Angle in rad
$\boldsymbol{\mu}$	Sample mean

1 Introduction

With the introduction of motorized passenger vehicles in the 19th century, engineers have been seeking to make driving safer, more comfortable, and efficient. From mechanical inventions to decrease mortality in fatal accidents, towards electrical systems that aim to support the driver to prevent accidents during critical maneuvers, modern Advanced Driver Assistance Systems (ADAS) aim to fully assume vehicle control.
Reliable and accurate knowledge of the vehicle's current pose and its static and dynamic environment is indispensable to safely plan trajectories of autonomous vehicles. Prevailing autonomous prototypes monitor vehicle states and environment using high-precision, reliable, and costly sensor setups. Providing these features for series production vehicles strongly confines the variety of feasible sensors. However, empowered by fusing information of large sensing networks in in-series vehicles, autonomous vehicles become readily available [13]. A key challenge herein is to robustly detect and capture faulty sensor performance that possibly lead to severely life-threatening or mortal injuries.
Automotive safety levels, with its highest classification level Automotive Safety Integrity Level (ASIL)-D, are designed to test and validate reliability of ADAS. Without assuring their respective integrity requirements, companies will hesitate to release these features. Nowadays, releasing complex algorithms for vehicle maneuvering according to ASIL-D is avoided by requesting the driver to permanently monitor the system, making the driver responsible of safe maneuvering to preventing judicial charges.

1.1 Robust Pose Estimation Problem Statement

This thesis focuses on obtaining frequent, highly accurate, and reliable vehicle pose estimates within digital highway maps or within highway lanes. These estimates are indispensable to ensure safe vehicle maneuvering. Real-world environments impose complexity such as multipath scattering, pose ambiguities, and sensor outages, depicted in Fig. 1.1 to Fig. 1.3 respectively. Furthermore, the uncertainties of the applied sensors are either unknown or roughly estimated impeding their fusion and validation. The burden of being reliant on a cost-efficient and space-saving sensor setup intensifies these issues. Digital maps, often referred to as additional virtual sensors, can help to alleviate these challenges. However, these maps have to be updated continuously and as availability and performance of wireless interfaces cannot meet the requirements to download huge datasets of point clouds, only sparse HD maps are practical. Lane-level maps represent an applicable approach as they contain an efficient parametric representation. The public funded project *Cooperative highly automated driving* (Ko-HAF)[1] aims to collect, fuse, and redistribute lane-level map content[2] from near-series vehicles on a back-end server.

1.2 Applications of Robust Pose Estimation

Among various applications where precise and robust pose estimates in structured environment are advantageous or indispensable, three major fields are presented below.

Autonomous Vehicles

Beside the gain in comfort when handing over driving task to the system, a major aim of autonomous vehicles is to decrease traffic mortality. Road accidents caused 3180 deaths, 66 513 severely injured and 323 799 slightly injured people in Germany in 2017 [76]. Automated driving features have the potential to significantly decrease the amount of road accidents due to their unconfined learning abilities. Moreover, individual mobility can be provided to a broad range of people, especially for those who are unable to operate vehicles due to legal, financial or physical restrictions. Exchange of static and dynamic map content and vehicle driving strategies enables intelligent route planning through interacting, autonomous vehicles, investigated in UR:BAN[3] and IMAGinE[4]. Therefore, modern traffic can be converted to be far more efficient and environmentally compatible, which helps to decrease air pollutions, especially in cities. Furthermore, automated driving features can be used for freight traffic and reallocation of car sharing vehicles.

Mobile Robots

Another, yet similar field of application is mobile robots, independently whether indoor, outdoor, aerial, ground or underwater. All share the necessity to obtain exact pose estimates either of the robot itself, other robots, or parts of the robot.

[1] https://www.ko-haf.de/
[2] With a mean data size of approximately 33 kB/km
[3] http://urban-online.org/en/urban.html
[4] https://imagine-online.de/en/home/

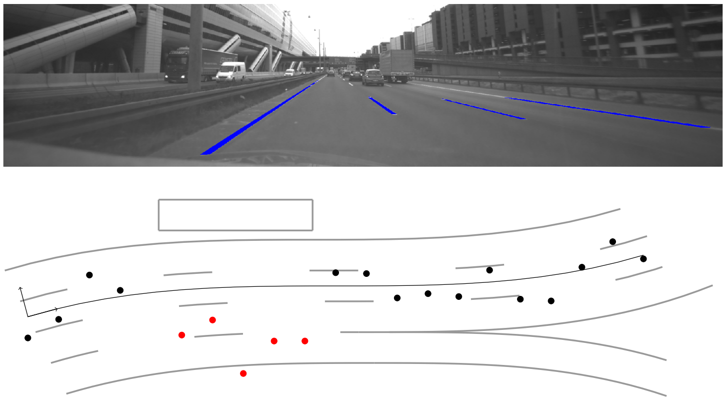


Figure 1.1: **Outlier - Multipath Scattering**. The black line displays the true vehicle movement and the black dots its corresponding Global Navigation Satellite System (GNSS) pose estimates. Buildings (gray box) with tall, flat surfaces reflect GNSS signals, which can lead to systematic offsets in GNSS pose estimates (red dots). The image shows the highway A3 through the Frankfurt Airport.

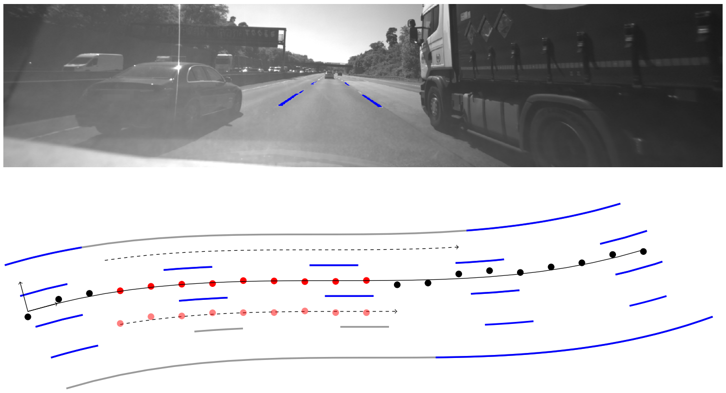


Figure 1.2: **Ambiguities - Map Matching**. Detected lane markers (blue lines) are aligned with the map (gray lines) to estimate map-relative poses (black dots). Due to missing detections, insufficient field of view or concealed by other vehicles (black, dashed lines), as displayed in the image, only a subset of lane markers can be detected. Thus the true pose (red dots) may be ambiguous to the matcher (light red dots).

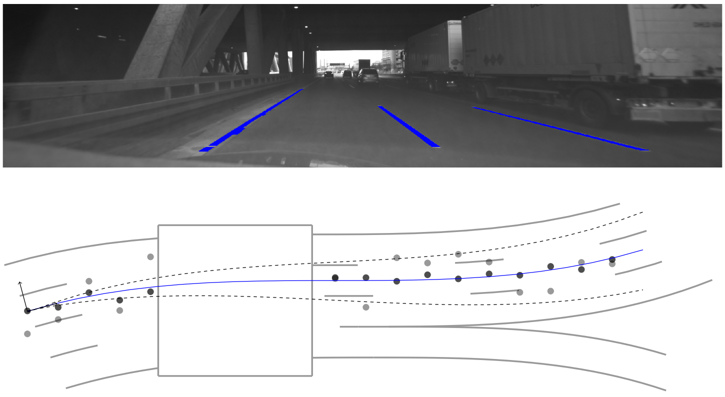


Figure 1.3: **Sensor Outage - Shadowing and Illumination**. Tunnels (white box) disturb or preclude GNSS (gray dots) and camera (black dots) pose estimates due to shadowing of GNSS signals and volatile illumination. However, the true vehicle movement (blue line) may still be estimated by integrating odometry estimates (dashed black lines) which drift over time. The image shows a tunnel on the highway A3.

The latter is of special interest if parts of the robot are not mechanically fixed, e.g. flexible robot arms or in distributed systems.

Indoor Navigation

A key example of localization in structured environment is indoor navigation, which is important whenever rescuing robots or firemen enter buildings with damaged infrastructure or decreased visibility due to smog and fire. Indoor localization and navigation can help to reach a desired point of interest on the shortest way possible, reducing the required amount of time spent in the building.

1.3 Contribution

The objective of this thesis is to estimate the vehicle's pose in lane-level highway maps using a front and rear gray-scale camera, a low-cost GNSS receiver, and the 2D vehicle dynamic sensor data provided over the Controller Area Network (CAN) interface.
Major limitations are the use of single camera setups with non-overlapping views, where the distance of detected lane markers cannot be directly observed resulting in a ray of feasible positions, a low-cost GNSS receiver with systematic offsets and

susceptibility to multipath scattering and 2D dynamic sensors, which make it impractical to estimate roll and pitch angles directly. Furthermore, digital maps have to be memory efficient, as they are updated constantly from a server. The utilized maps only contain 2D lane markers, reflector posts and traffic signs.
Promising results were shown to estimate poses within driving lanes or digital maps using Kalman Filters (KFs), Particle Filters (PFs) or pose graph optimization techniques. To enable higher-level ADAS applications, both a precise lane- and map-relative pose estimate must be provided at high frequencies using near-series sensors and sparse HD maps. While most approaches suffer from limiting applicability due to accuracy, computational effort or robustness, this thesis presents a localization approach that meets the depicted requirements. As visual pose estimation techniques are prone to viewing conditions, a statement on the estimations accuracy and integrity must be permanently provided.

The contributions of this thesis are

- An approach to estimate sensor uncertainties, with heterogeneous measuring principles, without the necessity of costly, high-precision validation sensors or extensive measurement procedures.
- An elaborate approach for robust map matching of lane marker light rays on 2D point clouds.
- Consolidation of sensor measurements through pose graph optimization to accurately estimate vehicle poses within lane-level highway maps.
- Detection of temporary and permanent sensor corruption and resolving of measurement ambiguities.
- Estimation of the current pose accuracy and integrity by investigating the optimization residuals.
- An extensive evaluation of the algorithms across the vehicle's full dynamic range and in demanding scenarios with perturbed viewing conditions.

1.4 Thesis Outline

This thesis is structured as follows.

Ch. 2: Fundamentals
Chapter 2 introduces essential fundamentals and deepening literature for the reader to understand and retrace this thesis. A short introduction into quaternions and dual numbers is followed by a subsection on multi-sensor data fusion techniques with the focus on Bayes Filter (BF). The chapter is concluded by an overview on numerical optimization.

Ch. 3: Related Work
Chapter 3 presents pioneer work regarding autonomous driving in urban and highway areas with a deepening survey and comparison of localization techniques using KFs, PFs and pose graph optimization in particular. This chapter is devoted to present the scope and contribution of this work.

Ch. 4: Robust Pose Estimation
Chapter 4 presents a novel approach for robust vehicle pose estimation in lane-level highway maps by solving an optimization problem represented by a pose graph. Sensor uncertainties are estimated from variations within pose and odometry estimators. These uncertainties are used to weight the optimization residuals. It's key elements are the rejection of outliers and resolving of map matching ambiguities. Furthermore, an approach to compute the integrity of optimized pose estimates is provided.

Ch. 5: Experimental Evaluation
Chapter 5 describes the test vehicle setup, software framework, test field, ground truth, and datasets. Both performance and feasibility of the presented approach will be investigated using a variety of datasets across the vehicle's full dynamic range. Additionally, the approach will be applied on datasets with demanding viewing conditions, such as rain and undergrade crossings in cloverleaf intersections. These datasets will be used to investigate whether the integrity module can detect permanent sensor corruption. The chapter is concluded by examining the computational burden of the presented approach.

Ch. 6: Summary, Conclusion and Outlook
Chapter 6 discusses the results and applicability of the presented approach. Both the approach and sensor setup are examined and future research in this field is motivated.

2 Fundamentals

This chapter is devoted to introduce key concepts and some advanced background information for the reader to understand the presented work herein. At this point, the author wishes to stay with the basics, without the claim for completeness, and references to the scientific literature in the corresponding sections.

2.1 Numbers

2.1.1 Quaternions

Quaternions were introduced by Hamilton in 1866 [43]. Although frequently used as another way to express rotations in a 3D space, quaternions originally are an extension to complex numbers with a broad range of applications. Quaternions are commonly represented by

$$q = q_w + q_x i + q_y j + q_z k \tag{2.1}$$

where q_w, q_x, q_y, q_z are real numbers and i, j, k are the quaternion units with the property

$$i^2 = j^2 = k^2 = ijk = -1 \tag{2.2}$$

known as the *Hamilton convention* [43], utilized by most programming libraries and throughout this work. Another common convention is the JPL convention, where $ijk = 1$. A unit quaternion fulfills the condition

$$||q|| = \sqrt{q\overline{q}} = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2} = 1 \tag{2.3}$$

where $\overline{q}$ is the conjugate of q

$$\overline{\boldsymbol{q}} = q_w - q_x i - q_y j - q_z k \tag{2.4}$$

A pure quaternions fulfills the condition $q_w = 0$. Popularity of quaternions in 3D space arises from *Euler's rotation theorem*, which states that manifold subsequent rotations about a fixed point can be reproduced by a single rotation around a fixed axis. This axis is called *Euler axis* and will be denoted by $\boldsymbol{u} = (u_x, u_y, u_z)^T$. The total rotation will be denoted by α. Extending *Euler's formula*

$$e^{ix} = \cos(x) + i\sin(x) \tag{2.5}$$

from complex to quaternion numbers yields

$$q = \cos\left(\frac{\alpha}{2}\right) + (u_x i + u_y j + u_z k)\sin\left(\frac{\alpha}{2}\right) \tag{2.6}$$

For simplicity, in the following chapters, a quaternion will be represented by a 4D vector

$$\boldsymbol{q} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} cos(\alpha/2) \\ sin(\alpha/2)u_x \\ sin(\alpha/2)u_y \\ sin(\alpha/2)u_z \end{bmatrix} \tag{2.7}$$

When representing a 3D point $[p_x, p_y, p_z]$ by a pure quaternion $\boldsymbol{p} = [0, p_x, p_y, p_z]$ the rotation around a unit quaternion q can be expressed by

$$\boldsymbol{p}' = H(H(\boldsymbol{q}, \boldsymbol{p}), \overline{\boldsymbol{q}}) \tag{2.8}$$

using the *Hamilton product* between two quaternions

$$H(\boldsymbol{q}, \boldsymbol{p}) = \boldsymbol{q} \cdot \boldsymbol{p} = \begin{bmatrix} q_w p_w - \boldsymbol{p}_{xyz} \cdot \boldsymbol{q}_{xyz} \\ p_w \boldsymbol{q}_{xyz} + q_w \boldsymbol{p}_{xyz} + \boldsymbol{p}_{xyz} \times \boldsymbol{q}_{xyz} \end{bmatrix} \tag{2.9}$$

$\boldsymbol{q}_{xyz}$ and $\boldsymbol{p}_{xyz}$ denote the 3D quaternion units vector and q_w and p_w the scalar real part. $\boldsymbol{p}'$ can be interpreted as $\boldsymbol{p}$ rotated clockwise around the *Euler axis* $\boldsymbol{u}$ of $\boldsymbol{q}$ by the angle α. Again, for simplicity, the rotation of a 3D vector $\boldsymbol{p}$ around a unit quaternion $\boldsymbol{q}$ will be denoted by

$$\boldsymbol{p}' = \boldsymbol{q} \cdot \boldsymbol{p} \tag{2.10}$$

An inverse or counter-clockwise rotation of $\boldsymbol{q}$ can be computed by using a negative angle α. Due to *Euler's formula*, this is equivalent to conjugation of $\boldsymbol{q}$

$$q^{-1} = \cos\left(\frac{-\alpha}{2}\right) + (u_x i + u_y j + u_z k)\sin\left(\frac{-\alpha}{2}\right)$$
$$= \cos\left(\frac{\alpha}{2}\right) - (u_x i + u_y j + u_z k)\sin\left(\frac{\alpha}{2}\right) \tag{2.11}$$
$$= \overline{q}$$

Quaternion rotations can be combined by

$$\boldsymbol{q}' = \boldsymbol{q}_2 \cdot \boldsymbol{q}_1 \tag{2.12}$$

where the rotation around $\boldsymbol{q}'$ resembles a rotation around $\boldsymbol{q}_1$ followed by a rotation around $\boldsymbol{q}_2$. This operation is noncommutative. Eq. 2.8 is not a combination of a clockwise and counter-clockwise rotation. Subsequent quaternion rotations in the *Hamilton product* are

$$\boldsymbol{p}' = \boldsymbol{q}_2\boldsymbol{q}_1\boldsymbol{p}\overline{\boldsymbol{q}}_1\overline{\boldsymbol{q}}_2 \tag{2.13}$$

Partial or weighted quaternion rotations, which are essential in recursive state estimation, can be performed by averaging weighted quaternions

$$\overline{\boldsymbol{q}} = \left(\sum_{i=1}^{n} w_i\right)^{-1} \sum_{i=1}^{n} w_i \boldsymbol{q}_i \tag{2.14}$$

where w_i is the scalar quaternions weight. However, two flaws arise. Firstly, the averaged quaternion is not a unit quaternion. Secondly, although $-\boldsymbol{q}$ and $\boldsymbol{q}$ represent the same rotation, the sign inversion of single quaternions changes the overall average. Markley et al. [62] present an approach to overcome these issue by minimizing the weighted sum of the squared Frobenius norm of the corresponding rotation matrices, which results in

$$\overline{\boldsymbol{q}} = \frac{[w_1 - w_2 + z]\boldsymbol{q}_1 + 2w_2(\boldsymbol{q}_1^T \cdot \boldsymbol{q}_2) \cdot \boldsymbol{q}_2}{||(w_1 - w_2 + z)\boldsymbol{q}_1 + (w_2 - w_1 + z)\boldsymbol{q}_2||} \tag{2.15}$$

$$z = \sqrt{(w_1 - w_2)^2 + 4w_1 w_2 (\boldsymbol{q}_1^T \cdot \boldsymbol{q}_2)^2} \tag{2.16}$$

Compared to other common rotations, such as *Euler angles* or *Rodrigues' rotation formula*, rotating with quaternions has the following advantages

- compact storage

- efficient composition
- stable spherical interpolation[1]

For completeness, rotation matrices can be derived from quaternions by

$$\boldsymbol{R}(\boldsymbol{q}) = \begin{bmatrix} 1-2q_y^2-2q_z^2 & 2q_xq_y-2q_wq_z & 2q_xq_z+2q_wq_y \\ 2q_xq_y+2q_wq_z & 1-2q_x^2-2q_z^2 & 2q_yq_z-2q_wq_x \\ 2q_xq_z-2q_wq_y & 2q_yq_z+2q_wq_x & 1-2q_x^2-2q_y^2 \end{bmatrix} \tag{2.17}$$

Example: A 90 degree rotation of the unit-x vector, represented by a pure quaternion $\boldsymbol{p}$, around the z-axis results in the unit-y vector. The rotation will be performed by a quaternion $\boldsymbol{q}$ using the rotation matrix representation and the *Hamilton product*.

$$\boldsymbol{q} = \begin{bmatrix} cos(\pi/2/2) \\ sin(\pi/2/2)\cdot 0 \\ sin(\pi/2/2)\cdot 0 \\ sin(\pi/2/2)\cdot 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ 0 \\ 0 \\ 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} q_w \\ 0 \\ 0 \\ q_z \end{bmatrix} \quad \boldsymbol{p} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ p_x \\ 0 \\ 0 \end{bmatrix}$$

For the rotation matrix $\boldsymbol{R}(\boldsymbol{q})$, according to Eq. 2.17, follows

$$\boldsymbol{R}(\boldsymbol{q})\boldsymbol{p} = \begin{bmatrix} 1-2q_z^2 & -2q_wq_z & 0 \\ 2q_wq_z & 1-2q_z^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1-2q_z^2 \\ 2q_wq_z \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

And for the *Hamilton product*

$$\boldsymbol{r} = H(\boldsymbol{q},\boldsymbol{p}) = \begin{bmatrix} 0q_w + \begin{bmatrix} 0 \\ 0 \\ q_z \end{bmatrix} \cdot \begin{bmatrix} p_x \\ 0 \\ 0 \end{bmatrix} \\ 0\begin{bmatrix} 0 \\ 0 \\ q_z \end{bmatrix} + q_w \begin{bmatrix} p_x \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ q_z \end{bmatrix} \times \begin{bmatrix} p_x \\ 0 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 \\ p_xq_w \\ p_xq_z \\ 0 \end{bmatrix}$$

[1] See Gimbal Lock

And then

$$\boldsymbol{q}\cdot\boldsymbol{p} = H(\boldsymbol{r},\overline{\boldsymbol{q}}) = \begin{bmatrix} 0 + \begin{bmatrix} p_x q_w \\ p_x q_z \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ -q_z \end{bmatrix} \\ q_w \begin{bmatrix} p_x q_w \\ p_x q_z \\ 0 \end{bmatrix} + 0 \begin{bmatrix} 0 \\ 0 \\ -q_z \end{bmatrix} + \begin{bmatrix} p_x q_w \\ p_x q_z \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ -q_z \end{bmatrix} \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ p_x q_w^2 - p_x q_z^2 \\ p_x q_w q_z + p_x q_w q_z \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

with

$$(\boldsymbol{q}\cdot\boldsymbol{p})_{xyz} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{2.18}$$

2.1.2 Dual Numbers

Dual numbers were introduced in the 19th century by Clifford [26]. Similarly to complex and quaternion numbers, dual numbers extend real numbers by a nilpotent dual operator ε

$$d = a + b\varepsilon \tag{2.19}$$

where a and b are real numbers and d is a dual number. The dual operator ε has the property

$$\varepsilon^2 = 0 \tag{2.20}$$

Different from complex and quaternion numbers, the nonreal part of a dual number does not feedback to the real part. Addition and subtraction is performed similarly to complex numbers, where real and dual part are added/subtracted separately. However, the multiplication differs, due to the dual operator's property

$$(a+b\varepsilon)(c+d\varepsilon) = ac+(ad+bc)\varepsilon+bd\varepsilon^2 = ac+(ad+bc)\varepsilon \tag{2.21}$$

An elaborate presentation on dual numbers and their arithmetic properties is presented by Veldkamp [88]. One application for dual numbers is Automatic Differentiation (AD), where derivatives of a function can be computed exactly, by simply evaluating the function itself using dual numbers instead of ordinary scalars. This can be particularly useful for optimization algorithms, where derivatives of a function are needed, for instance for a parameter update step.

2.1.3 Dual Quaternions

Expressing a quaternion's real numbers q_w, q_x, q_y, q_z with dual numbers, is called a dual quaternion. The dual quaternion is composed of a real and a dual part

$$\boldsymbol{q} = \boldsymbol{q}_r + \boldsymbol{q}_d\varepsilon \tag{2.22}$$

where $\boldsymbol{q}_r$ and $\boldsymbol{q}_d$ are quaternions in a vector representation. Dual quaternions are used to obtain closed-form algebraic expressions for spatial displacements in sensor registration [74]. Another application, is the generation of derivatives in the field of optimization presented in Sec.2.3.3.

2.2 Multisensor Data Fusion

One key challenge of robotics is the fusion of sensor data among a variety of physical domains to estimate system states that are not be directly observable. One example is the localization and mapping problem for autonomous vehicles with a variety of sensors such as its dynamic sensors, cameras, lidars, radars, GNSS receivers. The aim of filtering techniques is to both recover and increase the accuracy of system state estimates.
The following section will give a short introduction on temporal generative models, where the state at time t is only dependent on the state at time $t-1$ and the control u at time t. These models are referred to as Hidden Markov Models. A more elaborate introduction is presented by Thrun et al. [82] and Bar-Shalom et al. [11].

2.2.1 Bayes Filter

BFs are filters used to estimate system states from measurement and control data. In the following, $bel(x_t)$ will denote the *belief* of a true system state x_t and $p(\cdot|\cdot)$ is used to describe uncertainties and noise in measurements and system models. Both are conditional Probability Density Functions (PDFs).
BFs consist of two fundamental steps, outlined in Alg. 1. In the *prediction* or *transition* step a *belief* $\overline{bel}(x_t)$ is computed from a prior *belief* $bel(x_{t-1})$ using the most recent control variable u_t. For localization problems this corresponds to the numerical integration of a system model's differential equation to estimate the movement of a robot. In the *estimation*, *update* or *innovation* step a *belief* $bel(x_t)$ is computed from another *belief* $\overline{bel}(x_t)$. Again, for localization problems, this step corresponds to the fusion of a pose measurement with a pose estimate from a prior pose updated with the vehicle odometry.

Algorithm 1 Bayes Filter Step [82]

```
   Input: $bel(x_{t-1}, u_t, z_t)$
   Output: $bel(x_t)$
   Prediction Step:
1: $\overline{bel}(x_t) = \int p(x_t|u_t,x_{t-1})bel(x_{t-1})dx_{t-1}$
   Update Step:
2: $bel(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t)$
```

A central issue of BFs is the implementation of the measurement probability $p(z_t|x_t)$ and state transition probability $p(x_t|u_t,x_{t-1})$. Gaussian filters were among the earliest implementations of BF. *Beliefs* are represented by Multivariate Normal Distributions (MNDs). Its most prominent representative is the KF [52] described in Sec. 2.2.3. An alternative to *Gaussian Filters* are nonparametric filters that approximate the PDFs by a finite number of samples. Key advantages, towards KFs, are that highly nonlinear systems and other distributions, including multimodal distributions, can be modeled. However, there is a trade-off between computation time and accuracy when choosing the number of samples. Its most prominent representatives are Sequential Monte Carlo (SMC) methods or PF described in Sec. 2.2.4.

2.2.2 Multivariate Normal Distributions as Beliefs

To compute the integration in line 1 and composition in line 2 of Alg. 1 either a closed-form solution or a numerical integration of the *belief* function or PDF is required. However, most PDFs do not have definite or finite integrals and their numerical integration is computationally expensive. This motivates the application of MNDs in *Gaussian Filters*. MND are represented by a mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$

$$\boldsymbol{f}(\boldsymbol{x}) = \frac{1}{(\sqrt{2\pi})^m \det(\boldsymbol{\Sigma})} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{2.23}$$

where m is the vector and matrix dimension. MNDs arise as limits of sums of independent multidimensional random variables, which is known as the Central Limit Theorem (CLT). The occurrence of MNDs where multidimensional random variables can be seen as a superposition of many independent individual effects, motivates their wide applicability. Advantages of using MNDs are

- MNDs can be described with a few parameters, its mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
- Addition and composition of uncorrelated MNDs have simple arithmetics and their result is normally distributed.
- MNDs are widely applicable due to the CLT
- Although MNDs have no definite integral, the definite integral from $-\infty$ to ∞ can be evaluated using the *Gaussian integral*

Basic Arithmetics

A linear combination of two uncorrelated MNDs $\mathcal{X}$ and $\mathcal{Y}$ with mean $\boldsymbol{\mu}_x$, $\boldsymbol{\mu}_y$, covariance matrices $\boldsymbol{\Sigma}_{xx}$, $\boldsymbol{\Sigma}_{yy}$ and correlation matrices $\boldsymbol{\Sigma}_{xy}$, $\boldsymbol{\Sigma}_{yx}$ is

$$Z = \boldsymbol{A}\mathcal{X} + \boldsymbol{B}\mathcal{Y} + \boldsymbol{c} \tag{2.24}$$

$$\boldsymbol{\mu}_z = \boldsymbol{A}\boldsymbol{\mu}_x + \boldsymbol{B}\boldsymbol{\mu}_y + \boldsymbol{c} \tag{2.25}$$

$$\boldsymbol{\Sigma}_{zz} = \boldsymbol{A}\boldsymbol{\Sigma}_{xx}\boldsymbol{A}^T + \boldsymbol{A}\boldsymbol{\Sigma}_{xy}\boldsymbol{B}^T + \boldsymbol{B}\boldsymbol{\Sigma}_{yx}\boldsymbol{A}^T + \boldsymbol{B}\boldsymbol{\Sigma}_{yy}\boldsymbol{B}^T \tag{2.26}$$

Further arithmetic properties on MNDs can be found in [5].The addition of MNDs motivates the *prediction* step of the KF.

Composite Probability

Fusing measurements of multiple sensors or multiple measurements of single sensors over time can significantly increase the estimation accuracy. Thus, it is elementary to *Gaussian Filters* to compose estimates of multiple MNDs with no, known or unknown correlations. Fig. 2.1 shows the resulting composed covariance matrices for these cases in blue. The simplest case is when X and Y are uncorrelated and results in

$$\boldsymbol{K} = \boldsymbol{\Sigma}_{xx}\left(\boldsymbol{\Sigma}_{xx} + \boldsymbol{\Sigma}_{yy}\right)^{-1} \tag{2.27}$$
$$\boldsymbol{\mu}_{zz} = \boldsymbol{\mu}_x + \boldsymbol{K}\left(\boldsymbol{\mu}_y - \boldsymbol{\mu}_x\right) \tag{2.28}$$
$$\boldsymbol{\Sigma}_{zz} = \boldsymbol{\Sigma}_{xx}\boldsymbol{\Sigma}_{yy}\left(\boldsymbol{\Sigma}_{xx} + \boldsymbol{\Sigma}_{yy}\right)^{-1} = (\boldsymbol{I} - \boldsymbol{K})\boldsymbol{\Sigma}_{xx} \tag{2.29}$$

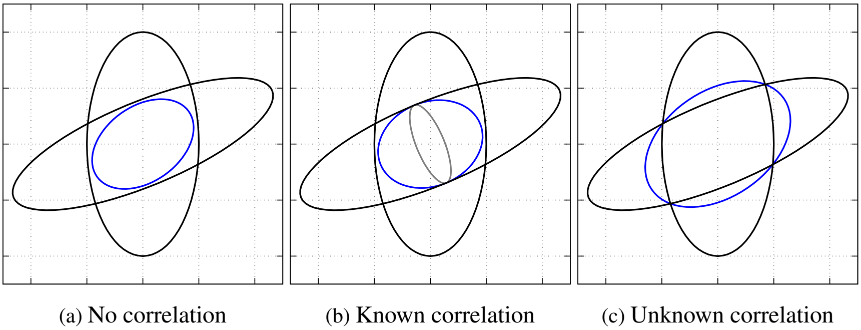
$\boldsymbol{K}$ can be interpreted as a *gain*, which describes how much of the update $(\boldsymbol{\mu}_y - \boldsymbol{\mu}_x)$ is performed. It is evident that increasing $\boldsymbol{\Sigma}_{yy}$ the composite mean tends to $\boldsymbol{\mu}_z \rightarrow \boldsymbol{\mu}_x$ and decreasing $\boldsymbol{\Sigma}_{yy}$ the composite mean tends to $\boldsymbol{\mu}_z \rightarrow \boldsymbol{\mu}_y$. The introduction of $\boldsymbol{K}$ and its interpretation as *gain* is done intentionally as the *update* step of KFs uses a similar notation. If X and Y are correlated and their correlation $\boldsymbol{\Sigma}_{xy}$ is known, their estimates can be fused

$$\boldsymbol{K} = \left(\boldsymbol{\Sigma}_{xx} + \boldsymbol{\Sigma}_{xy}\right)\left(\boldsymbol{\Sigma}_{xx} - 2\boldsymbol{\Sigma}_{xy} + \boldsymbol{\Sigma}_{yy}\right)^{-1} \tag{2.30}$$
$$\boldsymbol{\mu}_{zz} = \boldsymbol{\mu}_x + \boldsymbol{K}\left(\boldsymbol{\mu}_y - \boldsymbol{\mu}_x\right) \tag{2.31}$$
$$\boldsymbol{\Sigma}_{zz} = \boldsymbol{\Sigma}_{xx} + \boldsymbol{\Sigma}_{xy} - \boldsymbol{K}(\boldsymbol{\Sigma}_{xx} + \boldsymbol{\Sigma}_{xy}) = (\boldsymbol{I} - \boldsymbol{K})(\boldsymbol{\Sigma}_{xx} + \boldsymbol{\Sigma}_{xy}) \tag{2.32}$$

As motivated in Fig. 2.2a, the covariance matrix computed from Eq. 2.32 stays within the intersection of the corresponding covariance matrices $\boldsymbol{\Sigma}_{xx}$ and $\boldsymbol{\Sigma}_{yy}$ for changing correlation matrices. This only holds for correlation matrices within the intersection. This is a nonrestricting assumption as the noise of a signal cannot be smaller than its correlation with another signal in general. The idea of Covariance Intersection (CI) [51] is to compute a covariance matrix, which inherits the intersection. Therefore a linear interpolation between two input matrices is computed by

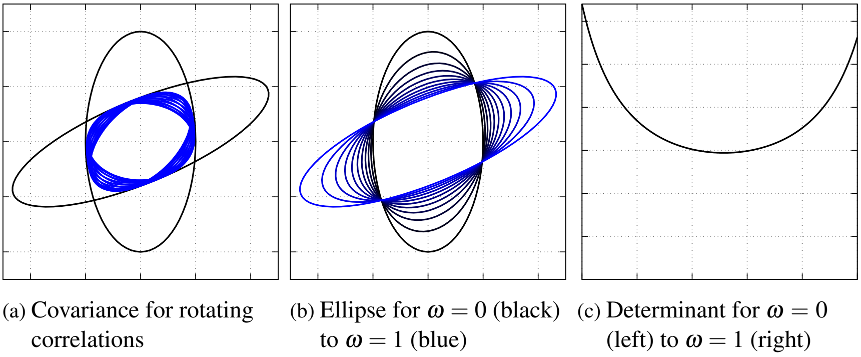
$$\boldsymbol{\Sigma}_{zz}^{-1} = \omega\boldsymbol{\Sigma}_{xx}^{-1} + (1-\omega)\boldsymbol{\Sigma}_{yy}^{-1} \tag{2.33}$$
$$\boldsymbol{\mu}_z = \boldsymbol{\Sigma}_{zz}\left(\omega\boldsymbol{\Sigma}_{xx}^{-1}\boldsymbol{\mu}_x + (1-\omega)\boldsymbol{\Sigma}_{yy}^{-1}\boldsymbol{\mu}_y\right) \tag{2.34}$$

where the scalar ω is typically chosen to minimize either the sum of eigenvalues (trace) or product of eigenvalues (determinant) of $\boldsymbol{\Sigma}_{zz}$ as depicted in Fig. 2.2c. Instead of optimizing ω numerically Reinhard et al. [71] presented a closed-form solution to minimize the determinant of $\boldsymbol{\Sigma}_{zz}$, for matrix dimensions below five, and trace, for matrix dimensions below four. While Eqs. 2.27 - 2.29 and Eqs. 2.30 - 2.32 are optimal if their assumptions of no or known correlation hold, CI is a nonoptimal algorithm and a conservative upper bound of the uncertainty or covariance matrix. Optimal data fusion techniques in presence of unknown correlations are discussed in [96].



(a) No correlation (b) Known correlation (c) Unknown correlation

Figure 2.1: **Composition of Multivariate Normal Distributions**. Two MNDs with $\sigma_x = 5$ / $\sigma_y = 1.5$ rotated counterclockwise by $\pi/8$ and $\sigma_x = 1$ / $\sigma_y = 4$ are displayed as black ellipses. Eqs. 2.27 - 2.32 (a), Eqs. 2.30 - 2.32 (b), Eqs. 2.33 - 2.34 (c) are applied to compute the resulting ellipse (blue). An MND with $\sigma_x = 0.5$ / $\sigma_y = 1.5$ rotated counterclockwise by $\pi/8$ (gray ellipse) is used as correlation for b).



(a) Covariance for rotating correlations (b) Ellipse for $\omega = 0$ (black) to $\omega = 1$ (blue) (c) Determinant for $\omega = 0$ (left) to $\omega = 1$ (right)

Figure 2.2: **Covariance Intersection**. The black ellipses are similar to Fig. 2.1. However, the correlation is iteratively rotated counterclockwise by $\pi/16$. a) Eq. 2.32 is used to compute the resulting covariance (blue) which stays within the intersection of the black ellipses. b) shows the resulting ellipses for CI for increasing $\omega \in [0,1]$ and c) its determinant.

2.2.3 Kalman Filter

The KF was derived by Kalman in 1960 [52]. KFs are an implementation of BFs for linear systems, where the *beliefs* are represented by MNDs. Its *prediction* and *update* steps are outlined in Alg. 2.

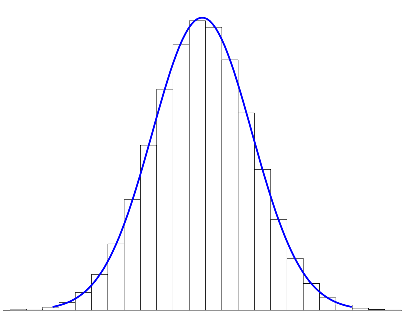
Algorithm 2 Kalman Filter Step [82]

```
   Input: $\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}, \boldsymbol{u}_t, \boldsymbol{z}_t$
   Output: $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$
   Prediction Step:
1: $\overline{\boldsymbol{\mu}}_t = \boldsymbol{A}_t\boldsymbol{\mu}_{t-1} + \boldsymbol{B}_t\boldsymbol{u}_t$
2: $\overline{\boldsymbol{\Sigma}}_{t-1} = \boldsymbol{A}_t\boldsymbol{\Sigma}_{t-1}\boldsymbol{A}_t^T + \boldsymbol{R}_t$
   Update Step:
3: $\boldsymbol{K}_t = \overline{\boldsymbol{\Sigma}}_t\boldsymbol{C}_t^T(C_t\overline{\boldsymbol{\Sigma}}_{t-1}\boldsymbol{C}_t^T + \boldsymbol{Q}_t)^{-1}$
4: $\boldsymbol{\mu}_t = \overline{\boldsymbol{\mu}}_t + \boldsymbol{K}_t(\boldsymbol{z}_t - \boldsymbol{C}_t\overline{\boldsymbol{\mu}}_t)$
5: $\boldsymbol{\Sigma}_t = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{C}_t)\overline{\boldsymbol{\Sigma}_t}$
```

Besides linearity, KFs furthermore assume that system states $\boldsymbol{\mu}$, control variables $\boldsymbol{u}_t$ and measurements $\boldsymbol{z}_t$ are uncorrelated. The *prediction* step is motivated by the addition of MNDs, see Eqs. 2.24 - 2.26. The *update* step is motivated by the composition of MNDs, see Eqs. 2.27 - 2.29. Over the years, a plurality of versatile extensions was investigated by researchers to extend the applicability of KF to nonlinear systems. The Extended Kalman Filter (EKF) [9] and Unscented Kalman Filter (UKF) [50] are probably its most prominent representatives. A detailed derivation and introduction on KFs and its derivatives can be found in [91] [82] [44].

2.2.4 Particle Filter

Contrary to parametric KFs, PFs [59] or SMCs are a nonparametric filters. KFs avoid the computationally expensive numerical integration in Alg. 1 line 1 and multiplication in Alg. 1 line 2 of PDFs by exploiting the simple arithmetics of MNDs. PFs represent the *belief* by samples or particles instead of PDFs. This is useful, as sampling from PDFs is easier and computationally less expensive than numerical integration. Fig. 2.3a and Fig. 2.3b show the approximation of a 1D and 2D MND by a histogram and a grid map, where each bar or cell represents a sample m at location x_t^m with weight w_t^m.
It is apparent that there is a trade-off between number of samples, and thus the PDFs approximation accuracy, and calculation effort. PFs are especially useful when the usage of non- Gaussian PDFs is required. For example, in multi-lane environments,


(a) Gaussian distribution represented by histogram with weights.

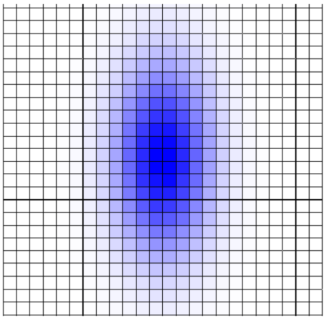

(b) 2D multivariate normal distribution approximated by grid with weights

Figure 2.3: **Representing Gaussians with Samples**. Non-parametric BF approximate continuous PDFs with discrete samples.

the lane assignment of detected lane markers can be ambiguous and must be modeled by multimodal PDFs. Alg. 3 delineates the fundamental computation tasks of a PF step. The PF *prediction* is computed in line 2, where the prior particles x_{t-1} are propagated using the control input u_t to x_t. The PF *update* is computed in line 3 where the weights of the particles w_t are recalculated using the conditional probability of measurement z_t given the weights particle x_t. Lines 5 - 7 represent the resampling step. Resampling is necessary to avoid degeneration of particles, where only a few particles with high weights are left.

Algorithm 3 Particle Filter Step [82]

```
    Input: $\mathcal{X}_{t-1}$, $u_t$, $z_t$
    Output: $\mathcal{X}_t$
 1: for $m = 1$ to $M$ do
 2:     sample $x_t^m \sim p(x_t | u_t, x_{t-1}^m)$
 3:     $w_t^m = p(z_t | x_t^m)$
 4:     $\overline{\mathcal{X}_t} = \overline{\mathcal{X}_t} + \langle x_t^m, w_t^m \rangle$
 5: for $m = 1$ to $M$ do
 6:     draw $i$ with probability $\propto w_t^m$
 7:     add $x_t^{[i]}$ to $\mathcal{X}_t$
```

This concludes the chapter on multisensor fusion. Wendel [91] or Bar-Shalom et al. [10] present a good overview on both parametric and nonparametric BFs with applications. A more extensive and versatile examination on *Probabilistic Robotics* is given by Thrun et al. [82]. Multisensor data fusion is not limited to probabilistic fusion, a brief survey on additional fusion technigues such as evidential belief reasoning, fuzzy reasoning, hybrid fusion and random set theoretic fusion is outlined in the survey paper of Khaleghi et al. [53].

2.3 Optimization

This section introduces the basics and nomenclature concerning convex optimization used throughout this work. For further details on optimization the author recommends to consult the respective literature. Boyd and Vandenberghe [16] provide a comprehensive survey on convex optimization. Wright and Nocedal [94] and Burkards [18] work on optimization further covers nonconvex optimization. An elaborate summary on large-scale nonlinear optimization is presented by Gould et al. [41].
Optimization is the task of finding optimal values of a set of parameters or variables with respect to some objective function, a quantitative measure of the system's performance. Parameters or variables can be subject to constraints. Identification of the objective, parameters, and constraints is called modeling. Beside choosing the right model for the objective, it is of utmost importance to further select an appropriate optimization algorithm. In the following, the problem types, solver types, constraints, and differentiation techniques will be introduced.

2.3.1 Problem Types

There are various problem types in optimization theory, which can be characterized by the properties described below.

- Linear or nonlinear
- Differentiable or nonsmooth
- Convex or nonconvex
- Constrained or unconstrained
- Continuous, integer or mixed integer
- Finite or infinite dimensional
- Global or local optimization

There are two types of constraints, equality and inequality constraints. While equality constraints require the current iterate $\boldsymbol{x}_k$ to be equal to some constant $\boldsymbol{c}_i$

$$\boldsymbol{x}_k = \boldsymbol{c} \tag{2.35}$$

inequality constraints keep the current iterate below or above a lower or upper bound $\boldsymbol{c}$

$$\boldsymbol{x}_k \leq \boldsymbol{c} \tag{2.36}$$

LP and convex quadratic programming with equality constraints can be solved directly or iteratively by solving the Karush-Kuhn-Tucker (KKT) system, which combines and rearranges first-order optimality conditions with equality constraints. In the special case of equality constraints in KKT systems the method of Lagrangian multipliers can be applied. Although iterative solving introduces the burden of balancing between solution precision and solver iterations, it can be beneficial for solving large systems. Active-set and interior-point methods are solver classes that are particularly designed to handle inequality constraints and are outlined in Sec. 2.3.2. A set where all points satisfy all constraints is called a feasible set.
There is a distinction between constraints and conditions. While constraints are used to bind parameter values, conditions are a useful tool to check the optimizer's state. Optimality conditions are a mathematical method to check whether the values of a parameter set are indeed a solution. For the first-order necessary optimality condition, the derivate or Jacobian must be equal to zero. For the second-order necessary condition, the second-order derivative or Hessian must be zero.
A problem is considered convex if both its parameter set and objective are convex as displayed in Fig. 2.4. In a convex feasible set $\mathcal{P}$ all connecting lines between set points are within the set. A function is considered convex if all its connecting lines are above the function graph. For convex problems the local minimum is also a global minimum. It is thus sufficient to find the local minimum.

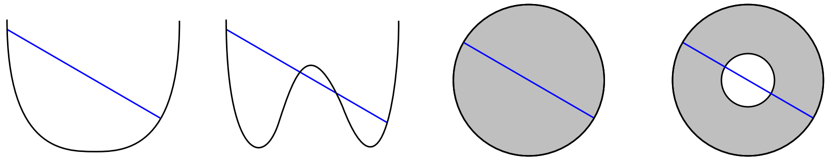

(a) Convex and nonconvex function (b) Convex and nonconvex set

Figure 2.4: **Convex Problem**. A problem is convex if both the objective function and feasible set are convex

Using these properties, optimization problems can be divided into six categories.

- Linear Programming
  linear objective, linear constraints, continuous parameter space, finite dimensional
- Quadratic Programming
  quadratic objective, linear constraints, continuous parameter space, finite dimensional
- Nonlinear Programming
  nonlinear objective, nonlinear constraints, continuous parameter space, finite dimensional
- Linear Integer Programming, integer programming
  linear objective, linear constraints, integer or mixed integer parameter space
- Non-smooth optimization
  nondifferentiable objective function or constraints
- Optimal Control
  Optimization problem includes dynamics in form of differential equations (infinite dimensional)

For the following sections the parameter space, constraints and objective function will be assumed to be continuous and finite dimensional. Thus, the remaining problem types are Nonlinear Programming with its special cases quadratic programming and LP. As all quadratic programming problems where the matrix of the quadratic part is positive definite and all LP problems are convex, a particular emphasis will be put on convex optimization.

2.3.2 Solver Types

Given an objective function and parameter set with an initial guess for its values, it is the solver's task to perform an intelligent sequence of improved estimates or iterates to minimize or maximize the objective function. The strategy how to move from one iteration to another distinguishes the solver algorithms. While some solvers use the values of the objective and constraint functions and, optionally, their first- and/or second-order derivatives, others accumulate information from prior iterates.
This section introduces the solver types used for convex optimization, while some solvers are applicable to nonconvex problems with no or bearable adjustments, others require more modifications. An overview on application of these solvers and required modifications is presented by Gould et al. [41] and Wright and Nocedal [94]. Although convex problems can be solved directly through matrix factorization or decomposition, these techniques are unstable for ill-conditioned matrices

or numerically expensive for large-scale optimization. This holds for convex problems with equality constraints, by solving the KKT system, as well. There are almost innumerable implementations of optimization solvers for convex optimization. Hence, the solver types will be divided into the following categories:

Unconstrained optimization

- Line-Search Methods
- Trust-Region Methods

Constrained optimization

- Active-Set Methods
- Gradient-Projection Methods
- Interior-Point Methods

Line-search algorithms first compute a search direction $\boldsymbol{p}_k$ and then a step length α_k, which describes how much the current iterate $\boldsymbol{x}_k$ is updated along this direction. Contrastively, trust-region algorithms first approximate the objective f by a model m_k and estimate a trust-region, depending on the estimated accuracy of the model. Secondly, the search direction is computed by optimizing the model m_k within the trust-region.
Active-set, gradient-projection and interior-point methods are used for convex optimization with inequality constraints and internally apply line-search or trust-region methods. Active-set methods move along search directions until inequality constraints are reached. Once reached, these constraints are considered as equality constraints and added to an active set of equality constraints, thus the naming. Graphically speaking, these optimization techniques tend to move along the hull of the feasible set, which motivates another optimization technique called interior-point. Its idea is to stay within the feasible set, by restricting the update direction through barrier functions along the inequality constraints. This sounds impractical but is often more efficient than active-set methods for large problems [94].

Line-Search Methods

Line-search methods are probably the most intuitive optimization algorithms. They pursue an iterative strategy where the current iterate $\boldsymbol{x}_k$ is updated along a primarily computed search direction $\boldsymbol{p}_k$ by the amount α_k, called step size, outlined in Alg. 4. The search direction is computed using first-order and optionally second-order derivatives. While steepest-descent methods only use first-order derivatives (Jaco-

bians), Newton methods additionally use the objective's second-order derivatives (Hessians) for quadratic convergence. Quasi- Newton or Gauss- Newton methods differ from Newton methods as they do not compute, but approximate Hessians using the information gained from the Jacobian of prior iterates. This can be beneficial as the computation of Hessians can be expensive, erroneous, and cumbersome [94]. Its most well-known implementations are Broyden-Fletcher-Goldfarb-Shannon, Symmetric-Rank-1 and Davidon-Fletcher-Powell methods.
Once the search direction $\boldsymbol{p}_k$ is computed, it is necessary to compute the step size α_k. Exact line-search methods attempt to find the global objectives minimum for α_k. This is computationally expensive as many function evaluations are necessary. Inexact line-search methods follow another strategy, where the step size search is aborted once certain conditions on the objectives decrease and curvature are met. Wolfe or Goldstein conditions are the most established of these conditions.

Algorithm 4 Line Search Methods

```
1: Compute a search direction $\boldsymbol{p}_k$
2: Go along search direction from current iterate $\boldsymbol{x}_k$
3: Find a step length $\alpha$ for $\boldsymbol{p}_k$ by mimimizing
   $\min_{\alpha>0} f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k)$
4: Repeat until convergence
```

Trust-Region Methods

Trust-region methods do not minimize the objective itself, but a model m_k of the objective within a trusted region around the current iterate $\boldsymbol{x}_k$. The estimation of trust-regions is a nontrivial task and similar to the problem of estimating the step size α_k in line-search methods.
Trust regions are usually defined by an Euclidean distance to $\boldsymbol{x}_k$ or ball around $\boldsymbol{x}_k$. However, a weighted distance or Mahalanobis distance, interpreted as ellipsoid, or box-shaped trust region are also used. Choosing the right size for the trust region is of utmost importance. Small regions result in unnecessary plurality of steps, while large regions result in the necessity to decrease the trust region iteratively as the model does not behave as the objective. Thus, trust regions are typically chosen iteratively from prior steps. A deepening examination of trust-region algorithms can be found in [94].

Algorithm 5 Trust-Region Methods

```
1: Evaluate objective $f$
2: Construct model funtion $m_k$ which approximates $f$ near $\boldsymbol{x}_k$
3: Estimate a trust region where $m_k$ is a good approximation
4: Find $\boldsymbol{p}_k$ subject to $\boldsymbol{x}_k + \boldsymbol{p}_k$ stays in the trust region
   $\min_{\boldsymbol{p}_k} m_k(\boldsymbol{x}_k + \boldsymbol{p}_k)$
5: Optionally decrease trust region if decrease in $f$ is not sufficient
6: Repeat until convergence
```

Active-Set Methods

Active-set methods are used to iteratively solve constrained, convex problems. While the method of Lagrange multipliers can be applied to convex problems with equality constraints, active-set methods can further handle inequality constraints, by keeping a working set $\mathcal{W}_k$ of active constraints, updated throughout the active-set algorithm iterations. This set includes all equality constraints and those inequality constraints reached by the solver, imposed as equality constraints. The resulting equality constrained, convex subproblem is solved by solving the resulting KKT system, which can be computed directly through factorization or iteratively through conjugate gradients.
Evidently, adding and subtracting constraints to the active set is challenging. Alg. 6 outlines the main steps of active-set methods. Its most prominent representatives are primal, dual, primal-dual active-set methods and simplex methods.

Algorithm 6 Active-Set Methods

```
1: Create quadratic sub problem $q$ using the working set $\mathcal{W}_k$
2: Solve equality constrained quadratic sub problem $q$
3: if Computed step length $\alpha$ is blocked by constraints $\notin \mathcal{W}_k$ then
4:     Add blocking constraints to $\mathcal{W}_{k+1}$
5: if Constraints from $\mathcal{W}_k$ can be dropped then
6:     Remove constraints from $\mathcal{W}_{k+1}$
7: Repeat until iterate minimizes $q$ and $\mathcal{W}_k$ unchanged
```

Gradient-Projection Methods

A drawback on active-set methods is the slow change of the working set, which usually is only updated by a single index at each iteration. This is distinctively slow for large-scale problems with many constraints. Gradient-projection methods overcome this drawback by allowing rapid changes to the working set $\mathcal{W}_k$ of active constraints. Alg. 7 displays the main steps of gradient-projection methods. Each iteration consists of two steps. In the first step, a search direction $\boldsymbol{p}_k$ for the unconstrained problem is computed via steepest descent. This direction is bent around the violated constraints. The resulting point is called Cauchy point. In the second step, the equality constrained problem with all active constraints at the Cauchy point is solved. These types of algorithms are most effective for simple constraints, such as bounds on parameters, where the feasible set's hull is similar to a box.

Algorithm 7 Gradient Projection Methods

```
1: Search along steepest descent direction $\boldsymbol{p}_k$ from current iterate $\boldsymbol{x}_k$
2: while constraint violated do
3:     Bent $\boldsymbol{p}_k$ around constraint so that iterate $\boldsymbol{x}_k$ stays in the feasible set.
4: Add all active constraints at $\boldsymbol{x}_k$ to working set $\mathcal{W}_k$.
5: Solve the subproblem using the constraints from the working set $\mathcal{W}_k$
6: Repeat until convergence
```

Interior-Point Methods

As active-set methods solve equality constrained subproblems, they usually move along the hull of the feasible set. Gradient projection methods behave similarly by projecting the iterate direction on the hull of the feasible set. Interior-point methods iteratively approach the optimal solution from the interior of the feasible set. Iterates are forced to stay within the interior of the feasible set using barrier functions, commonly with logarithmic properties, which penalize small distances to the hull of the feasible set. The barrier parameter μ scales the influence of the barrier function in contrast to the objective. The resulting perturbed KKT system can then be solved by using *Newton's method*.
As $\mu \to 0$ the perturbed solution $x_{\mu,k}$ converges to the solution of the unperturbed problem. For convergence or stability reasons, a substitution, refered to as slack variable/vector s is introduced. Similarly, the problem can be rearranged to for a dual problem. These methods are referred to as primal-dual interior-point methods. More details and an extension of interior-point methods to nonconvex problems can be found in [94]. Alg. 8 outlines the main steps of an interior-point algorithm. Interior-point methods use more expensive steps than active set methods, but generally require less iterations [94], making them more efficient for very large problems in practice.

Algorithm 8 Interior Point Methods

```
1: Choose $\sigma_k \in (0,1]$
2: Update $\mu$ and
3: Compute step length $\alpha_k$ by solving the linear system derived from the perturbed
   KKT conditions
4: Update current iterate $\boldsymbol{x}_k$, Lagrangian multiplier and the slack variable
5: Repeat until convergence or no feasible direction possible
```

2.3.3 Differentiation

The fastest and most precise way to compute derivatives is to use analytic derivatives. However, calculation of analytic derivatives by hand or through symbolic variables is tedious and unfeasible as algorithm complexity increases. One solution is to approximate derivatives using numerical differentiation, such as forward or central differences

$$\dot{f}_{\text{forward}}(x) = \frac{f(x+h) - f(x)}{h} \tag{2.37}$$

$$\dot{f}_{\text{central}}(x) = \frac{f(x+h) - f(x-h)}{2h} \tag{2.38}$$

where h is a small step, ideally infinitesimal small $h \to 0$, but big enough to prevent numerical errors due to rounding in floating point arithmetic or machine epsilon. The latter postulates that derivatives cannot be computed exactly and their accuracy further depends on the absolute x-value[2]. Furthermore, the function has to be reevaluated. Fig. 2.5 shows the numerical error using forward and central differences. Central differences have a smaller error of $\mathcal{O}(h^2)$ than forward or backward differences, which have $\mathcal{O}(h)$. The error estimation of open and closed Newton-Cotes formulae are derived from the *Taylor series expansion*. Naturally, higher-order differences, such as *Ridders' Method* [72], come with a smaller error but increased computation time.

Another way to compute derivatives is Automatic Differentiation (AD). The idea is to use dual numbers due to the similarity between the polynomial derivative property and the binomial theorem using Pascal's triangle. Let $P(x)$ be a polynomial of the form

[2] fraction bits of IEEE 754 double-precision are fixed

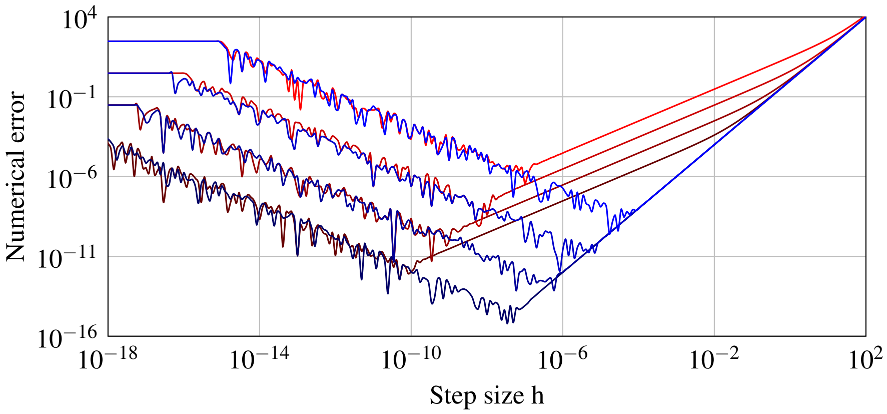


Figure 2.5: **Numerical Errors Using Forward and Central Differences on** x^3. This figure shows the numerical errors when using forward (red) and central (blue) differences to estimate the derivative of x^3, which is $3x^2$. For this function, the numerical error increases with increasing $x \in \{0.01, 0.1, 1.0, 10.0\}$ denoted by brighter colors. The error was computed on an Dell M4800 workstation, see Appx. A.3.

$$P(x) = p_0 + p_1 x + p_2 x^2 + ... + p_n x^n \tag{2.39}$$

Using a dual number $(x+\varepsilon)$ instead of the scalar x, expanding all polynomial terms using the binomial theorem, and removing all terms with ε^k if $k \geq 2$ results in

$$\begin{aligned} P(x) &= p_0 + p_1 x + p_2 x^2 + ... + p_n x^2 \\ &\quad + p_1 \varepsilon + 2 p_2 x \varepsilon + ... + n p_n x^{n-1} \varepsilon \\ &= P(x) + \dot{P}(x)\varepsilon \end{aligned} \tag{2.40}$$

The dual component inherits the derivative of $P(x)$. Intuitively, as many functions can be represented by an infinite Taylor Series Expansion around a radius of convergence, dual numbers can also be used for trigonometric, exponential, logarithmic functions as well. Computing units typically use Taylor Series Expansion to approximate these functions as well. Due to the chain rule of differentials multivariate functions can be handled as well. A profound introduction into derivatives is presented by Wright et al. [94]. AD is further used to estimate second-order derivatives using *hyper-dual numbers* [35].

3 Related Work

After a brief introduction on the development of autonomous vehicles, pioneer work and recent progress, a substantial survey on the field of localization and mapping for autonomous vehicles will be presented.

3.1 Autonomous Driving

In 1977, Tsukuba Mechanical Engineering Laboratory constructed a car that was able to follow white lane markers at 30 km/h. In 1986, the project *PROMETHEUS* was launched, intended to improve all areas of European road traffic. In the course of this project, Dickmanns et al. presented *VaMoRs-P* [32], [33], a Mercedes 500 SEL with automated lateral control driving at 130 km/h. It recognized lane markers and estimated its lane-relative pose. Their most prominent drive was in 1995 from Munich, Germany, to Odensee, Denkmark, a 1600 km drive with about 95 % traveled automatically. Simultaneously in 1995, researchers from the Robotics Institute of Carnegie Mellon University, drove a Pontiac Trans Sport, named *RALPH*, from Pittsburgh to San Diego in *No hands across America* [69], [68]. The vehicle's lateral controller was active 98.20 % of the time while longitudinal control was performed manually.
To foster the development of autonomous cars, the DARPA launched competitions from 2004 to 2007 [17]. Starting with a 240 km route through the Mojave Desert in 2004, which none of the participants was able to complete, a second competition in 2005 was launched. The new 212 km route in the Mojave Desert was completed by five cars and was won by *Stanley* [84]. In 2007, the DARPA launched a 60 km race in a nonpublic urban environment with human test drivers, which was won by a Chevrolet Tahoe named *Boss* [87].
Another milestone was set by Ziegler et al. [97] in 2013. They were able to adapt a Mercedes Benz S 500, called *Bertha*, so that it was able to drive autonomously from Mannheim to Pforzheim, Germany. A 103 km course covering rural roads, small villages and major cities, while relying merely on near-series sensors in combination with digital maps, vision, and radar sensors.
Nowadays, innumerable companies and universities are seeking to provide autonomous driving features to publicity. However, unclear legal situations, fail-safe

system requirements[1] and cost factors have prevented public introduction of autonomous systems so far, making a permanent human monitoring inevitable.

3.2 Localization and Mapping

The ability to obtain precise knowledge of vehicle system states and local environment is essential for ADAS. One crucial task of enhanced ADAS is the estimation of map- and lane-relative poses, also referred to as localization, and the creation of maps or mapping. Simultaneous Localization and Mapping (SLAM) [24] solves both the localization and mapping problem at the same time by incorporating land markers into the system states. The objective is to overcome the burden of inaccurate sensors by fusing their measurements over time.
Three main approaches for fusing sensor data will be reviewed in the following. The most prominent approach is the KF [52] and its successors, the EKF [9] and the UKF [50]. KFs assume Gaussian noise and linear systems. If these assumptions hold, KFs are optimal and computationally efficient. EKFs extend KFs to nonlinear systems by local linearization of system equations. UKFs, on the other hand, propagate multiple states and approximate their mean state and covariance using the propagated states. However, if measurements are not normally distributed, for example due to ambiguities, the filter cannot be applied directly and needs significant extensions to track multiple system states.
This motivates the second approach called SMC or PF [59]. Its basic idea is to approximate PDFs using multiple samples called particles. If more particles are used the approximation gets better but computational effort increases. Both KFs and PFs are Bayes filters and assume a Markov chain.
Last but not least, the localization problem can be solved via optimization of a nonlinear error function represented as a pose graph. An example is the Graph-SLAM [83]. All three approaches are compared in Sec. 3.2.4.

3.2.1 Kalman Filtering

The KF is one of the most widely used filters due to its plainness and ease of implementation. Its major assumptions are that state-space models are linear and system and measurement noise follow a Gaussian distribution. If these assumptions are met, the KF is an optimal estimator. There are many implementations of the KF for localization. Popular for aerial, but also used for terrestrial, robots are combinations of GNSS and Inertial Measurement Unit (IMU) sensors. Pose estimates are updated through numerical integration of the IMU using a Strapdown algorithm

[1] ISO 26262

[85]. The combination of an IMU with a computing unit, to solve the Strapdown algorithm, is called Inertial Navigation System (INS). Adding sensors, such as GNSS receivers, to limit the orientation, velocity, and position drifts in combination with INS are called aided INS. Typically an EKF is used to fuse the estimates. The INS is used in the Kalman update step and the Kalman innovation is performed using the GNSS pose estimate.
Barshan and Durrant-Whyte [12] extended these GNSS/INS systems to further estimate the inertial sensor errors, such as offset and scaling errors of the accelerometer and gyroscopes of the IMU, to further increase the accuracy of pose estimates.
Similarly Sukkarieh et al. [80] estimate low-frequency errors of IMUs as well as misalignments of IMU and GNSS units. Faulty GNSS pose estimates due to multipath scattering, are excluded by comparing the Mahalanobis distance of the Kalman update vector to a χ^2 threshold. The innovation covariance matrix is used for the Mahalanobis distance.
Werries and Dolan [92] propose an online adaption of covariance matrices of KFs to further increase the accuracy of GNSS/INS systems. For the covariance matrix, the estimation of the GNSS receiver is used. Those estimates typically rely on Dilution Of Precision (DOP) values. The system covariance matrix is computed from the Kalman update vector, while considering prior and posterior state uncertainties.
To increase the robustness of GNSS/INS systems Abuhashim et al. [2] propose a sequential combination of an innovation-based and a model-based Fault Detection and Identification (FDI). The innovation-based FDI relies on a Statistical Consistency Test, where the Mahalanobis distance in innovation steps of the KF is compared to a χ^2 threshold. When a measurement is classified as faulty, a subsequent model-based FDI checks the measurement. Here, the author chose a Multiple Model Adaptive Estimation technique, where several models are fused using weights computed from the similarity between the models. There are various approaches to extend Global Positioning System (GPS)/INS systems. Gao et al. [36] as well as Hazlett et al. [45] additionally incorporate a Wheel Speed Sensor (WSS) to improve odometry estimates in the KF update step and increase the vehicle position accuracy.
Toledo-Moreo et al. [86] have a similar setup but switch between odometry models of the WSS. A four-wheel model, for dynamic scenarios, and a simplified first-order model are used to estimate the odometry from the WSS. An Interacting Multiple Model (IMM) filter is used to switch between models depending on the scenario.
Laneurit et al. [55] furthermore add a camera to support the pose estimate accuracy. A simple lane tracker [8] is used to estimate vehicle poses within a road map. Additionally, object detections with a road membership, from a lidar sensor, are used to further improve the position accuracy. Similarly, Aeberhard et al. [3] use

a digital map together with a GNSS sensor, vehicle odometry, cameras and lidar sensors. A KF is used to estimate the vehicle pose within digital maps.
Dawood et al. [28] use a virtual 3D city model from a GIS as a map in combination with a camera for accurate pose estimates. Features from a camera image and a virtual image, created from a GIS, are extracted via a Harris Corner Detector and Scale-Invariant Feature Transform descriptors and are aligned via a POSIT algorithm [29]. The measurements are fused in an IMM-UKF.

3.2.2 Particle Filtering

PFs are SMC methods, which represents posterior PDFs by a set of samples called particles. Advantages of PFs compared to KFs are that state-space models can be nonlinear and PDFs do not have to be Gaussian. However, the PF has an increased computational effort depending on the number of particles. There are many approaches to solve the localization problem with PFs.
Gregory et al. [38] estimate vehicle poses using a GNSS/INS unit and vehicle odometer data. Results show that the PF performs better than a similar KF implementation. The authors motivate these results due to inherent errors of IMUs and their time-varying changes. Despite GNSS/INS system, PFs are frequently used when multimodal and other non Gaussian distributions are present. Multimodal states can occur when estimating map-relative poses in multi-lane environments with concealed perceptions.
Chausse et al. [23] apply a PF to estimate vehicle poses in lane-level maps using a camera, GNSS receiver, IMU, Steering Angle Sensor (SAS), and vehicle odometer. The application of a PF, compared to a priorly implemented KF [55], is motivated due to ambiguities in multi-lane environments. Likewise, Jo et al. [49] utilize a PF for vehicle localization using a low-cost GPS, a precise digital map with centimeter accuracy, a front and rear camera and vehicle on-board motion sensors such as WSS and yaw rate sensor. Their Root Mean Squared Error (RMSE) position for a recorded testfield dataset is 54 cm.
Levinson et al. [58] solve the localization problem using a PF together with a GPS sensor, an IMU, a WSS, and a lidar sensor together with a high-resolution environment map. The map was priorly recorded and optimized via Graph-SLAM [84] and has a map error of less than 10 cm. During a 20 min drive the author *suggests that lateral error were almost always within* 10 cm, although the errors *were sometimes as large as* 30 cm.
A PF approach is suggested by Schindler [73] to estimate poses within a HD map, where lane markers are represented by circular arc splines. Points of detected lane markers, from a monocular camera, are aligned via *prototype fitting*, which is a generalization of Iterative Closest Point (ICP) algorithms. Additionally, a four-

layer laser scanner provides hypothesis of surrounding landmarks and a GNSS receiver is used for initialization within the map. A localization error below1 m and an orientation error below 1 deg was achieved.
A similar localization sensor setup is used by Deusch et al. [30]. Tree trunks, road signs and reflector posts, extracted from three front-facing lidars and an additional camera that mainly detects lane markers, are used to estimate the vehicle pose within a grid map. The data is fused using an PF approach and the vehicle's final pose is extracted by computing a weighted particle mean. A key feature of this approach is that landmarks are efficiently stored in feature vector instead of a raw grid map itself, decreasing the necessary map storage to 22 kB/km. The localization module has been extensively tested in the *Autonomous Driving* project [54] on a 5 km campus track at Ulm University, resulting in a lateral Standard Deviation (STD) error of 17 cm with a mean error of 9 cm and an orientation error of 0.01 deg and 0.23 deg respectively.

3.2.3 Pose Graph Optimization

Pose graph optimization techniques estimate an optimal set of poses by optimization a nonlinear Least-Squares (LSQ) problem. Vehicle and landmark pose estimates are represented by graph vertices and constraints between poses are represented by graph edges. Solving an optimization problem instead of using classic BFs has the advantage that nonlinear error terms can be incorporated, parameter boundaries can be set, residuals can be monitored, loop-closures between detection can be incorporated and additional parameters can easily be solved. However, due to the high complexity, it is indispensable to choose the right solver and formulate the problem in an efficient manner.
Thrun et al. [83] were among the first to successfully solve the SLAM problem via pose graph optimization. They introduce an approach called Graph-SLAM, which represents the SLAM problem as a graphical network. One key idea is to reduce the number of variables of typical SLAM solvers by removing map features and shifting their information in sparse matrices accordingly. Thus, only pose variables remain. They evaluate their approach using a robot on a Segway RMP platform equipped with odometry sensor, lidar and a GNSS sensor.
When HD maps are already present and the objective is to estimate the pose within the map, one problem of pose graph optimization is the initialization within the map if no positioning sensor is provided. Olson et al. [67] use a variant of stochastic gradient descent for faster convergence on pose graph problems for poor initialization. However, a low-cost GNSS receiver, which can provide sufficiently precise initial estimates, is typically present.

Lategahn et al. [57] presented an approach for localization using a mono camera setup, an IMU and large-scale feature maps. Feature maps are created priorly using a high-precision GNSS receiver and a stereo camera setup. The feature map is computed via pose graph optimization. To localize the vehicle within the map, landmarks of the mono image are associated with features from the map and adjusted by optimizing the pose using motion constraints from the IMU to prior poses. Outliers are rejected by examining their back projection error to a threshold, while it is ensured that at least half of all measurements are retained. Lane-level maps confine the amount of data, which is of special interest for wide-area maps or frequent map updates, as lanes can be represented efficiently using parametric functions such as splines or clothoids. Jeong et al. [48] extract road markings from a stereo camera setup and fuse the extractions together with vehicle odometry estimates in a Graph-SLAM. To increase the robustness of loop closures and matching on sub-maps, a random forest method is used to classify road markings into six categories. An average accuracy of 1.10 m was achieved over 4.70 km.
As priorly motivated, lane marker classification can help to decrease, but cannot exclude, erroneous map matches and false loop closures. Thus, it is important to reduce the influence of outliers. In pose graph optimization techniques a loss function is typically used to degrade outlier influence. Suenderhauf et al. [81] present an approach for robust Graph-SLAM that detects and rejects outliers during optimization. By making constraints switchable, the graph representation is subject to optimization.
Abramov et al. [1] perceive multi-lane detections from cameras. Lane marker detections from a serial automotive camera are refined by a second experimental camera and processed by a Graph-SLAM algorithm to obtain a lane feature set and a lane-relative trajectory. Such systems can also be used, during long-term perturbation of digital maps, as fall-back solutions.

3.2.4 Comparison

This section is concluded by a discussion on advantages and disadvantages of pose graph optimization techniques to solve the localization problem, compared to classic BFs such as EKFs and PFs. The most important features are discussed below and summarized in Tab. 3.1.

Multimodal Distributions

A key assumption of KFs is the assumption of Gaussian noise, which is unimodal. Multimodal distributions are hard to model within KFs. If current states can be approximated by multiple Gaussian distributions, multiple KF states or Gaussian

Mixture Models can be filtered over time [89]. However, this introduces additional issues, such as determining when multimodal states can be merged or must be divided. These issues also arise when using pose graph optimization. PFs on the other side do not presume a PDF and can intuitively be used for multimodal distributions as well. Although more particles are required to adequately represent multimodal states. This is especially useful when localization in multi-lane environments with a lane marker detection system as presented by Chausse et al. [23] or Schindler [73]. However, the extraction of single states, typically required by subsequent modules, from a plurality of PF states is a nontrivial problem. G-means clustering algorithms [42] can be used to extract single system states from an unknown amount of point cloud agglomerations.

Ambiguous Measurements

Ambiguous measurements and multimodal states are closely related as the latter usually arises from the former. Similar to multimodal distributions, ambiguous measurements are rather intricate to model in conventional KF techniques, e.g. through Multi Hypothesis Tracking [82], and are intuitively included in PF. For pose graph optimization, it is possible to model ambiguous measurements directly in the residual or let the solver switch between ambiguous measurements, removing the need of multiple potential states at equal time stamps.

Parameter Constraints and Boundaries

Parameter constraints or boundaries can be incorporated straightforward into optimization problems. There are sophisticated solvers for constrained optimization, such as the method of Lagrangian multipliers for equality constraints or active-set and interior-point methods for equality and inequality constraints. Although there are activities in including constraints in KFs [75] or PFs [21], they solve an ambivalent problem and are usually not distinct. Simon and Simon [75] present a method to incorporate inequality constraints into a KF. The idea is to project unconstrained KF estimates on a constraint surface. This procedure is similar to quadratic programming. Whereas Chao et al. use constraints to reweigh particles to eliminate those that do not fulfill the proposed constraints.

Nonlinear Systems

EKFs and UKFs are, among other derivatives of KFs, BFs that can handle moderate nonlinear systems with limited linearization errors. PFs are better suited when faced with higher-order nonlinearity. When compared to UKFs this property is obvious as SMC methods use more particles than a few sigma-points of UKFs.

Similarly, pose graph optimization techniques can readily be applied to nonlinear systems. When applying optimization techniques, the problem of convexity arises and convergence can be difficult to prove or may not be provable at all.

Computational Effort

Computational efficiency is of particular importance for real-time applications, especially when cost, heating, or space plays a decisive role. A detailed discussion on computational complexity of mathematical operations and algorithms is presented by Arora and Barak [6].

KFs and most of its derivatives are among the most efficient filters concerning computational effort. Its major computational task is the inversion of an n-dimensional matrix, where n denotes the number of system states, which is $\mathcal{O}(n^3)$ for Gauss-Jordan elimination. By using fast matrix multiplication [77], the inverse of a matrix can be computed with $\mathcal{O}(n^{2.38})$ [27]. Note that most recent extensions of this algorithm perform slightly faster.

For the single-agent localization problem, the state space dimension n is generally very small. Hence, for PFs the main computational complexity arises from the number of particles m. The most computationally expensive tasks are sampling and resampling techniques [82], [15], [22]. While independent sampling, where a random particle has to be searched m times with $\mathcal{O}(\log m)$ time, requires $\mathcal{O}(m \log m)$ time, low variance sampling, where particles are aligned and equidistantly sampled, requires $\mathcal{O}(m)$ time [82]. Bolic et al. investigated computational complexity of several resampling techniques for PFs. A rule of thumb is to assume that computational complexity increases linearly with the number of particles. However, as the PF problem is well suited for parallelization in presence of many particles [22], [47], the computational workload for the CPU can be further decreased. However, this comes with an additional computational workload for the GPU. Hendeby et al. [47] show that resampling can be done in $\mathcal{O}(\log(m))$ time on a CPU with an additional $\mathcal{O}(\log(m/n_{proc}))$ on the GPU, where n_{proc} is the number of GPU processors. For $m \approx n_{proc}$ this has a constant complexity.

For most optimization solvers, the complexity of an iteration and the convergence rate can be estimated quite well, e.g. Quasi-Newton methods have superlinear convergence and each iteration is performed in $\mathcal{O}(\log n)$ time [94], where n is the number of parameters subject to optimization. However, in real-world applications, the amount of steps necessary until a convergence criteria is met highly depends on the nonlinearity of the system, the initial values, machine epsilon[2], problem formulation, solver type and convergence criteria itself. Hence, hard real-time properties

[2] see IEEE 754

are difficult to guarantee and usually only computational time estimates, from empirical investigations, can be provided.

Outlier Rejection

For linear systems and uncorrelated measurements with Gaussian noise, KFs are optimal. There is no better estimator in presence of known uncertainties. But real-world applications almost always violate these properties. Misclassification, sensor disruption, failed convergence and additive noise yield, along with other errors, false measurements, referred to as outliers. The quality of estimates highly depends on robustly detecting and rejecting these outliers. This typically leads to a trade-off between rejection rate and availability of measurements.
For KF approaches, most algorithms screen the update step, see Alg. 2, line 4. Rejecting outliers can be implemented by applying a hypothesis test on the update. For MND a χ^2-test can be used to reject new measurements. The Euclidean length of the update step is converted to the statistical Mahalanobis distance, using the measurement covariance matrix $\boldsymbol{Q}_t$. If the statistical distance exceeds the χ^2-threshold, the update is rejected. This test is also known as Normalized Innovation Squared (NIS)-test [4]. An essential task herein is to choose an adequate significance level, comprising the trade-off between outlier rejection and availability of measurements. Further analysis on robustification of KFs can be found in [82] and [63]. PFs are, by their design, more robust against outliers than KFs as long as there are no or significantly fewer particles close to outliers. There is a trade-off between rejecting outliers and accepting multimodal states, by increasing the particles noise during propagation. For pose graph optimization techniques robustness against outliers can be achieved by applying an adequate *loss* function, which diminishes the influence of outliers. Moreover, pose graph optimization can be designed, so that outliers can be switched off [81] to exclude their influence.

	EKF	PF	Pose graph optimization
Multimodal Distributions	o	+	o
Ambiguous Measurements	o	+	+ +
Parameter Constraints and Boundaries	o	+	+ +
Nonlinear Systems	+	+ +	+ +
Computational Effort	+ +	-	- -
Outlier Rejection	+	+	+ +

Table 3.1: **Pose Graph Optimization Versus Bayes Filters**.

4 Robust Pose Estimation

This chapter presents an approach to robustly estimate rover poses within HD maps as well as within simultaneously created local maps of lane markers, as depicted in Fig. 4.1.

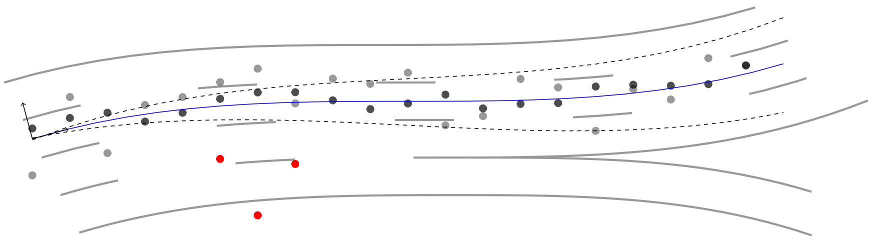


Figure 4.1: **Robust Pose Estimation**. The objective of robust pose estimation is to robustly and accurately estimate the rover's driven trajectory (blue line), given raw pose (gray shaded circles) and odometry (dashed black lines) estimates from sensor readings, while simultaneously detecting and rejecting outliers (red circle).

The localization problem will be modeled as a graph or graphical network, illustrated in Fig. 4.2. A graph connects vertices through edges. For localization problems poses, prepresented by vertices, are connected through constraints, represented by edges. These type of graphs will be called pose graphs. Before presenting the pose graph optimization and the key challenges when applying this technique for the localization and mapping problem, the pose representation and coordinate frame will be introduced.

Coordinate Frames

Earth-Centered Inertial (ECI) frames are practical when estimating satellite poses, which orbit the earth, through ephemeris data or by integrating differential equations, and will be denoted using an *eci* superscript. ECI frames have their origins at the earth's center of mass but do not rotate with respect to the stars. Earth-Centered, Earth-Fixed (ECEF) frames are convenient when estimating earth-relative poses. These frames rotate with respect to the earth. One of the most prominent ECEF frames is the World Geodetic System 1984 (WGS84) reference frame, denoted by

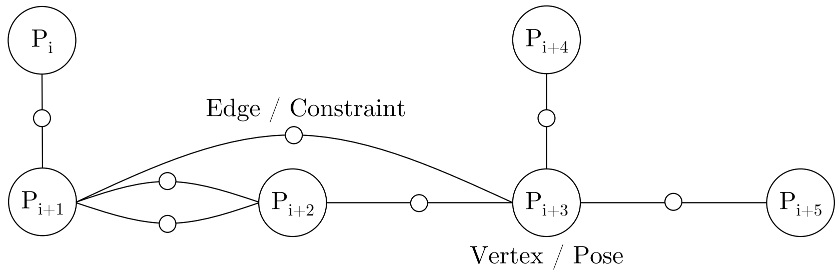


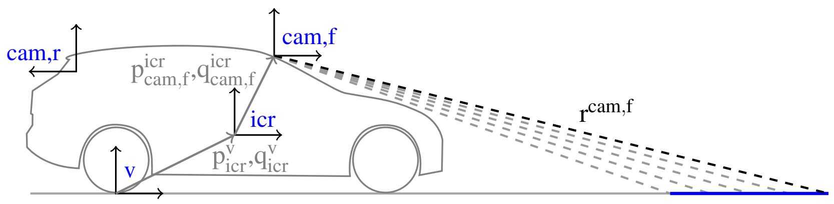
Figure 4.2: **Pose Graph Scheme**. Displayed is a graphical network with poses P as vertices, which are linked via constraints represented as edges.

w. The WGS84 reference ellipsoid is a model of the earth's surface and is convenient when estimating poses using GNSS satellites. However, computing a rover's trajectory in spherical coordinates such as WGS84 is cumbersome as longitude angles correspond to varying distances depending on the latitude angle. The Universal Transverse Mercator (UTM) frame[1] projects the curved earth's surface on a plane surface through Mercator projection. The UTM frame is locally isogonal as orientations can be computed directly from the rover's velocity vector using basic trigonometry. This property is useful when estimating robot movements on the earth's surface. All vectors in UTM coordinates are denoted using u or no subscript. This will be the default frame and the pose graph will be represented in this frame. For the vehicle v frame the rear axle center is projected on a ground plane. The Instant Center of Rotation (ICR) frame describes displacements of the vehicle chassis towards the vehicle body, which is displayed in Fig. 4.3. Displacements can result from high lateral or longitudinal accelerations. The indices cam, f and cam, r denote the front and rear camera coordinate frame. The camera's center is represented by its focal point. The cam frame will be used to denote vectors either in the front or rear camera frame. The conversion of a pose with position $\boldsymbol{p}$ and orientation quaternion $\boldsymbol{q}$ from UTM u to vehicle v coordinates is as follows

$$\begin{aligned} \boldsymbol{p}^u &= \boldsymbol{q}_v^u \cdot \boldsymbol{p}^v + \boldsymbol{p}_v^u \\ \boldsymbol{q}^u &= \boldsymbol{q}_v^u \cdot \boldsymbol{q}^v \end{aligned}$$

[1] MGRS Zone Number "32", Band Letter "U" for Frankfurt

where $(\cdot)^{u}_{v}$ denotes the vehicle pose in w-coordinates and $\cdot$ the Hamilton product between quaternions, introduced in Eq. 2.10. This holds for conversions between other frames correspondingly.



cam,r
cam,f
$p^{icr}_{cam,f}, q^{icr}_{cam,f}$
icr
$r^{cam,f}$
v
p^{v}_{icr}, q^{v}_{icr}

Figure 4.3: **Coordinate Frames**. The coordinate frame names are displayed in blue. Each detected lane marker (blue) is represented by a set of light rays $\boldsymbol{r}$. Lateral and longitudinal vehicle acceleration results in roll and pitch rotations of the *icr* frame towards the *v* frame.

4.1 Pose Graph Optimization

The idea of pose graph optimization algorithms is to represent and solve the localization problem as an optimization problem. The goal is to find an optimal set of poses $\mathcal{P}$ that minimizes a constrained nonlinear least squares problem, represented by a graph. The graph itself consists of poses with constraints between them. It is the optimizer's task to find the optimal poses P, during a specified time interval, depicted in Fig. 4.4. Pose estimates M from a GNSS receiver or matched lane marker detections on HD maps are not subject to optimization. Graph edges or constraints between or within these poses represent residuals in the optimizer, described in detail in Sec. 4.2.

All map matching and GNSS pose estimates M_i are added to a set of estimates $\mathcal{M}_{\text{map}}$ and $\mathcal{M}_{\text{gnss}}$ with indices sets $\mathcal{J}_{\text{map}}$ and $\mathcal{J}_{\text{gnss}}$. Additionally a parameter block P_i is added for each estimate to the set $\mathcal{P}_{\text{map}}$ and $\mathcal{P}_{\text{gnss}}$ with indices i_{map} and i_{gnss} respectively. A map matching pose estimate denotes the pose resulting from matching lane marker detections, in the vehicle v frame with HD maps or prior lane marker detections.

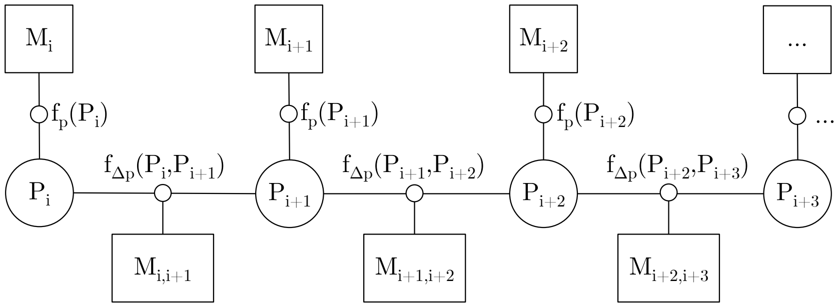


Figure 4.4: **Simplified Pose Graph**. The localization problem can be represented by a graphical network where vehicle poses P (circles) are subject to optimization. Poses P are linked to pose estimates M_i using a residual function f_{textp} or to other poses using odometry estimates $M_{i,i+1}$ and the corresponding residual $f_{\Delta p}$.

$$
\begin{aligned}
&\mathcal{P} = (P_1 \dots P_n) && \mathcal{M} = (M_1 \dots M_n) \\
&\mathcal{P}_{\text{map}} : \mathcal{P} \forall i \in \mathcal{J}_{\text{map}} && \mathcal{M}_{\text{map}} : \mathcal{M} \forall i \in \mathcal{J}_{\text{map}} \\
&\mathcal{P}_{\text{gnss}} : \mathcal{P} \forall i \in \mathcal{J}_{\text{gnss}} && \mathcal{M}_{\text{gnss}} : \mathcal{M} \forall i \in \mathcal{J}_{\text{gnss}}
\end{aligned}
$$

While parameter blocks P_i from the set $\mathcal{P}_{\text{gnss}}$ only consist of a vehicle position in UTM coordinates $\boldsymbol{p}_i$ and orientation $\boldsymbol{q}_i$, poses from the set $\mathcal{P}_{\text{map}}$ also contain the ICR rotation of the vehicle chassis towards the vehicle body $\boldsymbol{q}^v_{icr,i}$. ICR rotations are solely used to project lane marker detections from monocular cameras on the surface ground plane.

$$P_i = (\boldsymbol{p}_i, \boldsymbol{q}_i) \ \ \forall i \in \mathcal{J}_{\text{gnss}} \tag{4.1}$$

$$P_i = (\boldsymbol{p}_i, \boldsymbol{q}_i, \boldsymbol{q}^v_{icr,i}) \ \ \forall i \in \mathcal{J}_{\text{map}} \tag{4.2}$$

As motivated in the introduction of this thesis, outlier rejection and resolving ambiguities plays an important role when trying to achieve robust pose estimates. To embed these capabilities in the optimization problem, a loss function ρ is applied. Loss functions typically have linear properties for small values but their slope decreases with increasing values to weaken the influence of outliers. The generic, constrained nonlinear least squares problem is proposed as follows

$$\begin{aligned}
\min_{\mathcal{P}} \frac{1}{2}(& \sum_{i \in \mathcal{J}_{\text{gnss}}} \rho_{\text{gnss}}(d^2_{\text{gnss}}(P_i)) + \\
& \sum_i \rho_{\text{odo}}(d^2_{\text{odo}}(P_i, P_{i+1})) + \\
& \sum_{i \in \mathcal{I}_{\text{map}}} \rho_{\text{map}}(d^2_{\text{map},1}(P_i, P_{i+1}), \ldots, d^2_{\text{map},k}(P_i, P_{i+1}))) \\
& \text{s.t.} \quad l_i \leq P_i \leq u_i
\end{aligned} \tag{4.3}$$

where l_i and u_i are the lower and upper bounds for the parameter block P_i. d denotes the raw residual between pose estimates, presented in Sec. 4.2. The loss function ρ, presented in Sec. 4.3, is applied on the quadratic raw residual d^2 to reduce the influence of outliers

$$f(\cdot) = \rho(d^2(\cdot)) \tag{4.4}$$

The resulting adapted residual f is returned to the optimizer. To solve the localization problem online, only the most recent sensor measurements are taken into consideration, to keep computational effort low and constant. Therefore, loop closures are not taken into consideration, as they are highly unlikely to occur during small time intervals on highways. Consequently, constraints are stated only within subsequent parameter blocks or the block itself, although in general, constraints between any subset of parameter blocks can be implemented. Fig. 4.5 displays the final structure of the pose graph. Pose estimates exceeding a residual threshold are indicated as outliers by red dots. Blue dots denote deactivated ambiguities, e.g. for ambiguous map matches of detected lane markers.

4.2 Measurement Processing and Residuals

Residuals are used to give feedback to the optimization solver whether a parameter block variation is desirable or not. Residuals are represented in the pose graph as edges. For statistical data, it is feasible to use a statistical distance measure instead of using Euclidean distances. For multivariate normally distributed data, with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, the Mahalanobis distance [61] is used. It can be interpreted as a distance in STDs instead of a metric unit. The Mahalanobis distance

$$d_{\text{mahal}} = \sqrt{d^{\text{T}}_{\text{eucl}} \Sigma^{-1} d_{\text{eucl}}} \tag{4.5}$$

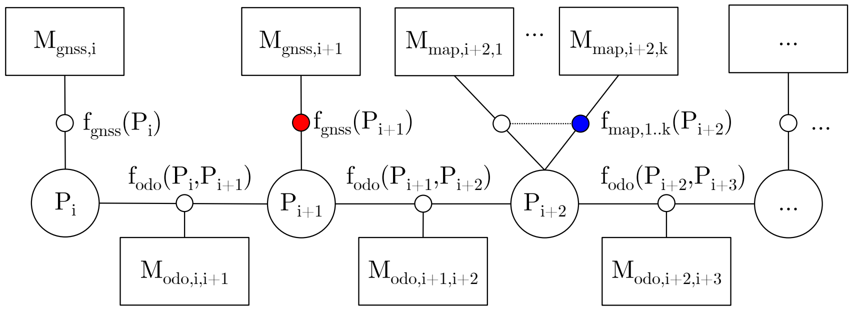


Figure 4.5: **Pose Graph**. For each map match or GNSS pose estimate M_{map} and M_{gnss} a vehicle pose P is added, which is subject to optimization. Poses are connected through adapted residuals f. Outliers (red) and ambiguities (blue) are detected and resolved by the solver.

is computed by scaling the Euclidean distance with the inverse root of the covariance matrix. Commonly the information matrix

$$\mathcal{I} = \Sigma^{-\frac{1}{2}} \tag{4.6}$$

which is the inverse square root of the covariance matrix, is used for convenience instead of the covariance matrix. The respective literature and this thesis, uses a calligraphic $\mathcal{I}$ for the information matrix and a calligraphic $\mathcal{J}$ for index sets. The information matrix $\mathcal{I}$ is computed by inversion and Cholesky decomposition of the covariance matrix. In the following, covariance matrices $\mathbf{\Sigma}$ are 6x6 matrices if not stated otherwise. The top-left 3x3 matrix represents the position uncertainty and the bottom-right 3x3 matrix the orientation uncertainty respectively. It will be assumed that position and orientations of raw sensor measurements are uncorrelated. Estimation of covariance matrices is explained in detail in Sec. 4.6.1.

4.2.1 GNSS

GNSS receivers can be used to estimate geospatial positions and velocities from transmission time or pseudoranges and Doppler frequency shifts or deltaranges of electromagnetic waves ($\approx$ 1.50 GHz) transmitted from a system of satellites. Each satellite is equipped with several high-precision atomic clocks. The time of signal transmission, of each satellite, is encoded in electromagnetic waves through phase-shifting. To confine each satellite signal from a mixture of signals and to precisely estimate signal start and signal frequency shift, a cross-correlation of a unique 1023 bit Pseudo Random Noise (PRN) code is conducted. The receiver uses the time of arrival to estimate distances to satellites. To prevent the necessity of high-precision, and expensive, atomic clocks in GNSS receivers, the receiver's clock[2] drift dt_c is also estimated, increasing the amount of necessary satellites, to solve for all unknowns, from three to four. Thus, GNSS receivers are also used as high-precision clocks for network synchronization, communication network, and data encryption[3]. Beside the time stamp itself, GNSS signals also contain imprecise satellite orbit data called almanacs, precise orbit data called ephimeris and other informations, such as satellite conditions.

In the following, the Single Point Positioning (SPP) algorithm for GNSS pseudoranges will be shortly outlined and the corresponding residual for the pose graph optimization will be presented. The program package RTKLIB[4] is employed, which provides most of the algorithms for GNSS processing.

Position Estimation

The objective of SPP is to find an optimal 3D position $\boldsymbol{p}^w$ and clock bias dt_c, denoted by the vector $\boldsymbol{x}$, given m pseudorange measurements P_r

$$\boldsymbol{x} = (\boldsymbol{p}^{w\mathrm{T}}, dt_c)^\mathrm{T} = (x, y, z, dt_c)^\mathrm{T} \tag{4.7}$$

$$\boldsymbol{y} = (P_{r,1}, P_{r,2}, ..P_{r,m})^\mathrm{T} \tag{4.8}$$

$$\boldsymbol{y} = h(\boldsymbol{x}) + \boldsymbol{v} \tag{4.9}$$

where $h(\boldsymbol{x})$ is the nonlinear measurement equation and $\boldsymbol{v}$ the random measurement error. Pseudoranges are pseudo distances between the receiver and the satellites. The term *pseudo* stems from the fact that pseudoranges are computed by plain multiplication of the speed of light c with the time passed between the time stamp of sending t_s and receiving t_r the satellite signal. Pseudoranges do not account for satellite clock biases $dt_{\mathrm{c},i}$, ionospheric delays I_i, tropospheric delays T_i, and other pseudorange measurement errors ε_pr. The geometric range ρ_i

[2] typically a quartz oscillator

[3] `https://gssc.esa.int/navipedia/index.php/Precise_Time_Reference`

[4] RTKLIB V.2.4.3, An Open Source Program Package for GNSS Positioning, `http://www.rtklib.com`

$$P_{r,i} = c(t_\mathrm{r} - t_\mathrm{s}) = \rho_i + c(dt_c(t_\mathrm{r}) - dt_{\mathrm{c},i}(t_\mathrm{s})) + I_i + T_i + \varepsilon_\mathrm{pr} \tag{4.10}$$

$$\rho_i = ||\boldsymbol{p}^{eci}(t_\mathrm{r}) - \boldsymbol{p}^{eci}_{s,i}(t_\mathrm{s})|| \tag{4.11}$$

represents the true physical distance between satellite and receiver antenna phase center $\boldsymbol{p}_\mathrm{s}$ and $\boldsymbol{p}_\mathrm{r}$. Rovers moving on the earth surface are subject to the earth rotation. The movement of constant ECEF points towards ECI frames during a time interval $\Delta t_i = (t_\mathrm{r} - t_\mathrm{s})$ can be, depending on the ECEF point and time Δt_i, multiple meters. This effect is illustrated in Fig. 4.6.

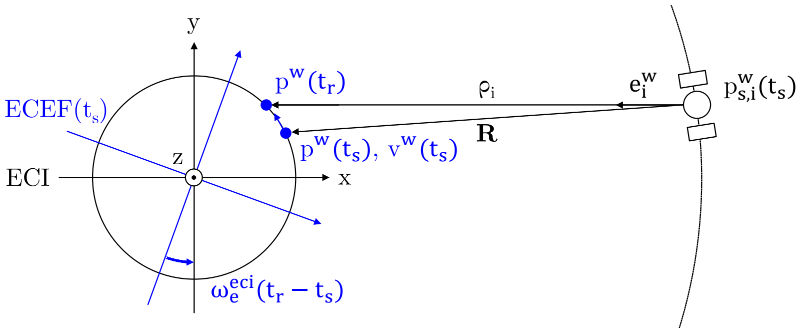


Figure 4.6: **Earth rotation correction**.

Assuming constant speed of light, the earth rotation correction can be approximated, using the approximation presented by Ashby et al. [7]

$$\boldsymbol{p}^{eci}_{\mathrm{rs},i} = \boldsymbol{p}^{eci}(t_\mathrm{s}) - \boldsymbol{p}^{eci}_{s,i}(t_\mathrm{s}) \tag{4.12}$$

$$\Delta t_i = \frac{\boldsymbol{p}^{eci}_{\mathrm{rs},i}}{c} + \frac{\boldsymbol{v}^{eci}\boldsymbol{p}^{eci}_{\mathrm{rs},i}}{c^2} \tag{4.13}$$

where $\boldsymbol{p}_{\mathrm{rs},i}$ represents the distance vector between the satellite position $\boldsymbol{p}_{\mathrm{s},i}$ and receiver position $\boldsymbol{p}_\mathrm{r}$. The equation simplifies to

$$\boldsymbol{v}^{eci} = \omega_\mathbf{e}\boldsymbol{e}^{eci}_z \times \boldsymbol{p}^{eci} \tag{4.14}$$

$$\rho_i \approx ||\boldsymbol{p}^{eci}_{\mathbf{rs},i}|| + \underbrace{\frac{\omega_\mathbf{e}}{c}\boldsymbol{e}^{eci}_z(\boldsymbol{p}^w(t_\mathbf{s}) \times \boldsymbol{p}^{eci}_{\mathbf{rs},i})}_{\textbf{earth rotation correction}} = \boldsymbol{p}^{eci}_{\mathbf{rs},i}|| + \frac{\omega_\mathbf{e}}{c}(y_{\mathbf{rs},i}x - x_{\mathbf{rs},i}y) \tag{4.15}$$

if the receiver has a fixed ECEF position or a negligible velocity within the ECEF coordinate system. $x_{\text{rs,i}}$ and $y_{\text{rs,i}}$ are the x and y component of $\boldsymbol{p}_{\text{rs},i}$ and ω_{e} the earth's angular rate. The nonlinear measurement function $\boldsymbol{h}(\boldsymbol{x})$ and its Jacobi matrix $\boldsymbol{H}$ is

$$\boldsymbol{h}(x) = \begin{bmatrix} \rho_1 + cdt_c - cdT_{c,1} + I_1 + T_1 \\ \rho_2 + cdt_c - cdT_{c,2} + I_2 + T_2 \\ \vdots \\ \rho_m + cdt_c - cdT_{c,m} + I_m + T_m \end{bmatrix} \qquad \boldsymbol{H} = \begin{bmatrix} -\boldsymbol{e}_1^w & 1 \\ -\boldsymbol{e}_2^w & 1 \\ \vdots & \vdots \\ -\boldsymbol{e}_m^w & 1 \end{bmatrix} \tag{4.16}$$

where $\boldsymbol{e}_i^w$ is the Line-Of-Sight (LOS) vector from receiver to satellite i. Details on computing the satellite position $\boldsymbol{p}_{s,i}^w$ by solving the Keppler equations[5] or by integrating differential equations[6] as well as the ionospheric I_i and tropospheric model T_i are described in detail in the appendix E.4 and E.5 of the RTKLIB manual [7]. The pseudorange measurement vector is approximated using the Taylor series expansion.
The elevation angle of satellite direction El_i, ephemeris data uncertainty σ_{eph}, PRN code bias σ_{bias} and ionospheric and tropospheric delay uncertainty σ_{ion} and σ_{trop} vary between satellites. To account for these varying uncertainties, an estimation of the pseudorange uncertainty is computed in Eq. 4.17 by adding up the individual uncertainties, which is valid for MNDs with uncorrelated uncertainties. The inverse of these uncertainties are diagonally aligned in a weight matrix W to compute the statistical Mahalanobis distance of each pseudorange measurement

$$\sigma_i^2 = F_i R_i (a_\sigma + b_\sigma / \sin El_i) + \sigma_{\text{eph},i}^2 + \sigma_{ion,i}^2 + \sigma_{\text{trop},i}^2 + \sigma_{\text{bias},i}^2 \tag{4.17}$$

$$W = diag(\sigma_1^{-2}, \sigma_2^{-2}, \ldots, \sigma_m^{-2}) \tag{4.18}$$

where F_s is the satellite system error factor[8], R_i the code-carrier-phase error ratio and a_σ and b_σ the carrier-phase error factor. The nonlinear, weighted LSQ

$$x_{k+1} = x_k + (H^{\text{T}} W H)^{\text{-1}} H^{\text{T}} W (y - h(x_k)) \tag{4.19}$$

[5] GPS and GALILEO
[6] GLONASS
[7] RTKLIB Manual Ver. 2.4.2
[8] 1.00 for GPS, GALILEO and BeiDou. 1.50 for GLONASS

can be solved iteratively. After the solver converged to a solution, a χ^2-test with a significance level α of 0.10 % is applied on the resulting distance between pseudorange measurements and the measurement function $h(x)$. If the χ^2-test fails, the estimation is recomputed by excluding satellites using a Receiver Autonomous Integrity Monitoring (RAIM) Fault Detection and Exclusion (FDE) algorithm. RAIM FDE techniques are useful to exclude single erroneous pseudorange measurements, but require at least six satellites and are not effective with mutliple erroneous measurements.

Velocity Estimation

The computation of the velocity vector from Doppler frequency shifts is similar to the position estimation. Henceforth, an unknown vector $\boldsymbol{x}$, which contains the rover velocity $\boldsymbol{v}$ and clock drift $c\dot{t}_c$, is to be found given a measurement vector $\boldsymbol{y}$ inheriting the Doppler measurements

$$\boldsymbol{x} = (\boldsymbol{v}^w, cd\dot{t}_c) = (v_x, v_y, v_z, \dot{t}_c) \tag{4.20}$$

$$\boldsymbol{y} = (-\lambda_1 D_{r,1}, -\lambda_2 D_{r,2}, \dots, -\lambda_m D_{r,m}) \tag{4.21}$$

$$\boldsymbol{y} = h(\boldsymbol{x}) + \boldsymbol{v} \tag{4.22}$$

where $D_{r,i}$ is the Doppler frequency shift measurement of satellite i and λ_i the carrier wave length. $D_{r,i}\lambda_i$ is commonly referred to as a deltarange. The measurement function $\boldsymbol{h}(\boldsymbol{x})$ and its Jacobi matrix $\boldsymbol{H}$ is given by

$$\boldsymbol{h}(x) = \begin{bmatrix} r_1 + cd\dot{t}_c - c\dot{T}_{c,1} \\ r_2 + cd\dot{t}_c - c\dot{T}_{c,2} \\ \vdots \\ r_m + cd\dot{t}_c - c\dot{T}_{c,m} \end{bmatrix} \qquad \boldsymbol{H} = \begin{bmatrix} -\boldsymbol{e}_1^w & 1 \\ -\boldsymbol{e}_2^w & 1 \\ \vdots & \vdots \\ -\boldsymbol{e}_m^w & 1 \end{bmatrix} \tag{4.23}$$

where r_i is the satellite range rate, which can be regarded as the velocity analogon to the geometric distance ρ_i for distances. Deriving the correction term in Eq. 4.15 with respect to time yields

$$r_i = \boldsymbol{e}_i^w(\boldsymbol{v}_i^w - \boldsymbol{v}^w) + \underbrace{\frac{\omega_e}{c}(\dot{y}_{\text{rs},i}x + y_{\text{rs},i}\dot{x} - \dot{x}_{\text{rs},i}y - x_{\text{rs},i}\dot{y})}_{\text{Rotation correction}} \tag{4.24}$$

A detailed derivation on relativistic error terms is presented by Zhang et al. [95]. All delta velocities between the satellite and rover perpendicular to their LOS-vector have no influence on the frequency shift, thus the multiplication in Eq. 4.24 with $\boldsymbol{e}_i^w$. The system is solved iteratively by Eq. 4.19, where the weight matrix $\boldsymbol{W}$ is set to an identity matrix $\boldsymbol{I}$.

Distance

After GNSS position $\boldsymbol{p}_{\text{gnss}}$ and velocity $\boldsymbol{v}_{\text{gnss}}$ are estimated, the vehicle orientation $\boldsymbol{q}_{\text{gnss}}$ is derived from the velocity vector. This presumes that the vehicle movement aligns with the vehicle orientation, which is valid for ground vehicles subject to moderate lateral accelerations. As roll angles are decoupled from the velocity vector, they are set to be zero. For each pose or parameter block $P_i = \boldsymbol{p}_i, \boldsymbol{q}_i$ with $i \in \mathcal{J}_{\text{gnss}}$ the Mahalanobis distance to the corresponding GNSS pose estimate $M_{\text{gnss},i}$ with position $\boldsymbol{p}_{\text{gnss},i}$ and orientation $\boldsymbol{q}_{\text{gnss},i}$ is computed

$$d_{\text{gnss}}(P_i) = \mathcal{I}_{\text{gnss}} \begin{bmatrix} \boldsymbol{p}_i - \boldsymbol{p}_{\text{gnss},i} \\ vec(\overline{\boldsymbol{q}}_{\text{gnss},i} \cdot \boldsymbol{q}_i) \end{bmatrix} \tag{4.25}$$

where $\overline{\boldsymbol{q}}$ is the complex conjugate quaternion, which represents an inverse rotation. $vec(\cdot)$ returns the imaginary part (x, y, z) of the quaternion. The GNSS information matrix $\mathcal{I}_{\text{gnss}}$ is derived from the estimated covariance in Eq. 4.17 and Eq. 4.18.

4.2.2 Odometry

Odometry estimation is the process of estimating ego-motion, or position and orientation variation, of an agent during a specified time interval. Odometry estimators can rely on a variety of sensors, such as cameras, dynamic sensors, GNSS receivers, radars and lidars. Most estimators either compute pose variation by numerically integrating velocities and angular rates, by aligning subsequent landmark detections or by computing transformations between pose estimates through numerical differentiation. Odometry estimates are of particular relevance for pose or object measurements, recorded at different time steps, of moving rovers. Using the odometry, pose estimates can be aligned to equal time stamps and fused to provide estimates of increased accuracy. In pose graph optimization algorithms, odometry estimates are used as constraints between pose estimates. Given the sensor setup in Sec. 5.1 three odometer categories are presented.

Vehicle Models

Numerical integration of vehicle models are sensitive to scaling and bias errors, especially for higher-order derivatives. Thus, velocity, acceleration, and angular rate sensors need to be accurately calibrated. The sensor calibration is outlined in Ch. 5. Two kinematic vehicle models for odometry estimation are presented in the following.
WSSs encode wheel rotations in small increments using electric or magnetic pulses. In the presence of WSS on all vehicle tires both the velocity v and yaw rate $\dot{\psi}$ can be estimated by comparing left and right encoder ticks or pulses

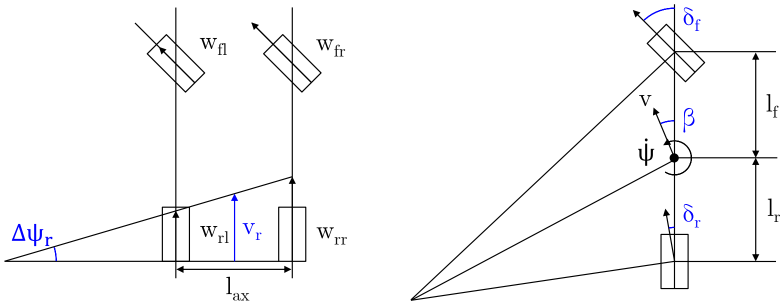


Figure 4.7: **Dynamic Odometry Models**. The movement can be estimated from wheel ticks (left) or steering angles (right).

$$v\Delta t = \frac{1}{4}\frac{w_{\mathrm{rr}}d_{\mathrm{rr}} + w_{\mathrm{rl}}d_{\mathrm{rl}} + w_{\mathrm{fr}}d_{\mathrm{fr}} + w_{\mathrm{fl}}d_{\mathrm{fl}}}{n_w} \tag{4.26}$$

$$\tan(\Delta\psi_{\mathrm{r}}) = \frac{\sin(\Delta\psi_{\mathrm{r}})}{\cos(\Delta\psi_{\mathrm{r}})} \approx \Delta\psi_{\mathrm{f}} = \frac{w_{\mathrm{fr}}d_{\mathrm{fr}} - w_{\mathrm{fl}}d_{\mathrm{fl}}}{l_{ax}} \tag{4.27}$$

$$\tan(\Delta\psi_{\mathrm{f}}) = \frac{\sin(\Delta\psi_{\mathrm{f}})}{\cos(\Delta\psi_{\mathrm{f}})} \approx \Delta\psi_{f} = \frac{w_{\mathrm{fr}}d_{\mathrm{fr}} - w_{\mathrm{fl}}w_{\mathrm{rl}}}{l_{ax}} \tag{4.28}$$

where n_w is the sensor-specific number of ticks per wheel rotation, w are the wheel ticks during the specified time interval, d is the respective wheel diameter and l_{ax} the axle track, displayed in Fig. 4.8. However, this WSS odometry is sensitive to wheel slippage.
A single-track odometry, based on a kinematic single-track model, uses the velocity estimation v together with the front steering angle δ_{f} to estimate yaw rate $\dot{\psi}$ and slip angle β

$$\dot{\psi} = \frac{v \cos \beta}{l_\mathrm{f} + l_\mathrm{r}} (\tan \delta_\mathrm{r} - \tan \delta_\mathrm{f}) \tag{4.29}$$

$$\beta = \tan^{-1} \left(\frac{l_\mathrm{f} \tan \delta_\mathrm{r} + l_\mathrm{r} + \tan \delta_\mathrm{f}}{l_\mathrm{f} + l_\mathrm{r}} \right) \tag{4.30}$$

where l_f and l_r are the distances between front and rear axle center to the vehicle's center of gravity as depicted in Fig. 4.8. For front steering vehicles the rear steering angle δ_r is set to zero. The derivation of the kinematic single-track model, and more complex models, can be found in [70]. However, more complex vehicle models were implemented but did not perform significantly better as the crucial errors are introduced by the sensors' accuracy.

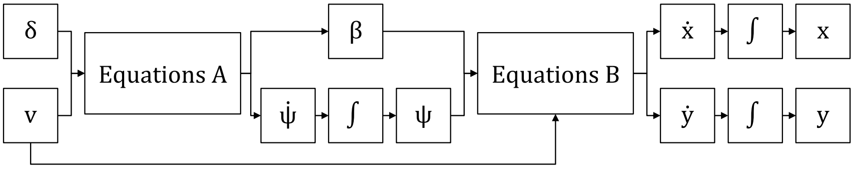


Figure 4.8: **Kinematic Single-Track Model Schema**. Equations A: Eq. 4.29 and Eq. 4.30. Equations B: Eq. 4.31 and Eq. 4.32.

Both odometry estimators above observe yaw rates $\dot{\psi}$ from wheel ticks w or steering angles δ. However, in presence of gyroscope sensors, yaw rates may also be measured directly. New positions $\boldsymbol{p}$ and orientations $\boldsymbol{q}$ from observed or measured yaw angles ψ and velocities v are computed by using an Euler integration step

$$\Delta \boldsymbol{q}_i = \boldsymbol{q}(\dot{\psi}_i \Delta t_i) \tag{4.31}$$

$$\Delta \boldsymbol{p}_i = \begin{pmatrix} \Delta x_i \\ \Delta y_i \\ \Delta z \end{pmatrix} = \begin{pmatrix} v \Delta t_i \cos \left(\frac{\psi_i + \psi_{i+1}}{2} + \beta_i \right) \\ v \Delta t_i \sin \left(\frac{\psi_i + \psi_{i+1}}{2} + \beta_i \right) \\ 0 \end{pmatrix} \tag{4.32}$$

where $\boldsymbol{q}(\psi_i \Delta t_i)$ is the quaternion rotation of angle $\psi_i \Delta t_i$ around the z-axis. The yaw rate $\dot{\psi}$ can be observed through the single-track model, wheel tick comparison or directly from gyroscope sensor readings. All yaw rate estimates can be considered uncorrelated and can be fused. For the velocity estimation v four wheel tick estimates can be compared and fused.

Visual Odometry

Another technique to estimate the odometry during a specified time interval is visual odometry [66]. The key idea is to estimate translation and rotation between subsequent camera images by minimizing the distance between corresponding image features in the images. The monocular odometry version of the library LIBVISO2[9] is used throughout this work and is presented by Geiger et al. [37], see Fig. 4.9. Instead of using computationally demanding Speeded Up Robust Features features, simple blob and corner masks are employed together with nonmaximum and nonminimum-suppression [65]. Once corresponding features are detected, a 2D Delaunay triangulation is used to establish neighborhood relations to other features. These relations are used to remove outliers, by checking neighbors of correspondences. For ego-motion estimation, a subset of these features, uniformly distributed over the image domain, is selected. Using an eight-point algorithm [60] the essential matrix, a particular case of the fundamental matrix which contains all the epipolar geometry, is estimated. It is assumed that the camera is moving at a known and fixed height over ground, which is a reasonable assumption for a fixed camera setup in a passenger vehicle. Small pitch and roll angles of the vehicle chassis towards its body have a negligible influence on the camera height over ground. As the Eight-point algorithm is sensitive to noise, the rotation $\boldsymbol{r}$ and translation $\boldsymbol{t}$ is estimated 50 times from 8 randomly drawn correspondences, and a Random Sample Consensus (RANSAC) algorithm is applied to reject outliers. The final transformation is refined by the winning RANSAC iteration.

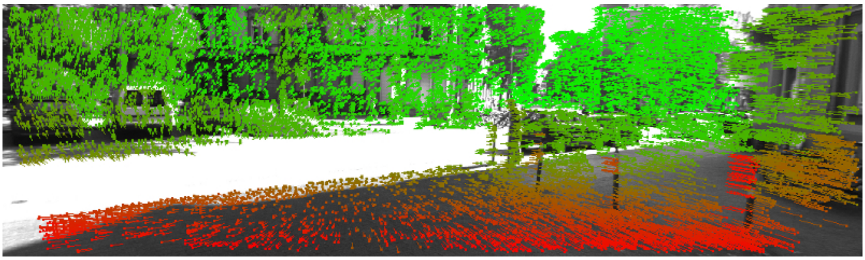


Figure 4.9: **Visual Odometry (LibViso)**. Feature matching of two frames while moving. Colors encode disparities [37].

[9] Libvisio V.2, C++ Library for Visual Odometry, `http://www.cvlibs.net/software/libviso`

Odometry from GNSS Singals

Numerical differentiation of GNSS pose estimates can be used to compute vehicle movements The simplest approximation of pose derivatives between two poses P_i and P_{i_1} are forward differences

$$\Delta \boldsymbol{q}_{i,i+1} = \overline{\boldsymbol{q}}_i \cdot \boldsymbol{q}_{i+1} \tag{4.33}$$

$$\Delta \boldsymbol{p}_{i,i+1} = \overline{\boldsymbol{q}}_i \cdot (\boldsymbol{p}_{i+1} - \boldsymbol{p}_i) \tag{4.34}$$

GNSS position and velocity estimates are subject to offset errors, which result from imprecise ephemeris, ionospheric and tropospheric data, corrupted GNSS signals due to multipath or other errors such as simplified models and neglecting relativistic effects. It stands to reason that differentiation of subsequent pose estimate inherits the advantage that constant offset errors cancel out. However, adding pose estimates as vertices to the pose graph while simultaneously adding their numerical differentiation or odometry as edges is nonoptimal as their uncertainties are highly correlated.

Distance

Given two pose estimates P_i and P_{i+1} and the odometry estimation $\Delta \boldsymbol{p}_{\text{odo},i,i+1}$ and $\Delta \boldsymbol{q}_{\text{odo},i,i+1}$ in the respective time interval, the distance between the poses is

$$d_{\text{odo}}(P_i, P_{i+1}) = \mathcal{I}_{\text{odo}} \begin{bmatrix} \overline{\boldsymbol{q}}_i \cdot (\boldsymbol{p}_{i+1} - \boldsymbol{p}_i) - \Delta \boldsymbol{p}_{\text{odo},i,i+1} \\ vec(\Delta \boldsymbol{q}_{\text{odo},i,i+1} \cdot \boldsymbol{q}_i \cdot \overline{\boldsymbol{q}}_{i+1}) \end{bmatrix} \tag{4.35}$$

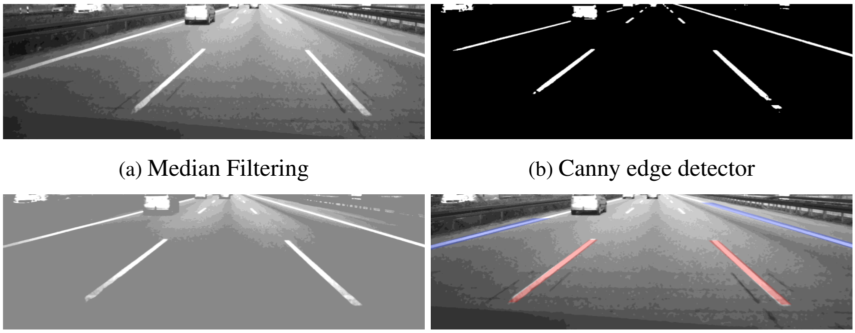
where $\mathcal{I}_{\text{odo}}$ is the information matrix, derived from the covariance matrix, of the odometry estimate. As presented above, a variety of odometry estimates, both 2D and 3D, was presented. A straightforward approach is to add a residual for each odometry estimator. However, as all vehicle models use the velocity given by the WSS, correlated signals are introduced, making the optimization problem nonoptimal. Introducing only the uncorrelated orientation estimation $\Delta \boldsymbol{q}_{\text{odo},i,i+1}$ and the position separately, avoids this problem, but increase the amount of residuals. This increases the amount of function evaluations for each optimization step and solver time. Alternatively, the odometry estimates are fused in advance as presented in Sec. 4.6.

4.2.3 Lane Marker Detection and Alignment

This section is divided into three parts. The first part will shortly outline the lane marker detection system used herein. The second and third focus on matching the lane marker detection on both an HD digital map that was recorded priorly, and priorly recorded lane marker detections. Multiple pose graphs are solved simultaneously, each containing only a subset of residuals.

Lane Marker Detection

The lane marker detection system was implemented in a preliminary study [Int1] and will be outlined shortly in this section to motivate the advantages and disadvantages of the respective approach. The input to the lane marker detection algorithm are gray-scale camera images, irrelevant whether from front or rear cameras. The camera images are rectified using the camera extrinsic parameters[10] and several filter masks are applied to prepare the image for the clustering step as depicted in Fig. 4.10.


(a) Median Filtering (b) Canny edge detector

(c) Intensity Average (d) Clustering

Figure 4.10: **Lane Marker Detection**. After image rectification and the application of various image filters, the resulting region of interests are clustered as presented by Harr et al. [Int1].

A median filter diminishes outlier pixels in the image to generate thoroughly filled lane markers. A Canny edge detector [19] with an expected width between lane marker boundaries is used to extract regions of interest in the image. Simultaneously, the intensity average of adjacent areas are compared to a threshold to cluster

[10] See Sec. 5.1.2

lane markers in the image. Clusters are projected in vehicle coordinates via Inverse Perspective Mapping on a ground plane to compute their 3D position. The resulting clusters are checked for plausibility via fuzzy-classification. The resulting lane marker clusters are represented by a set of lane marker elements, containing position, orientation and width as displayed in Fig. 4.11.

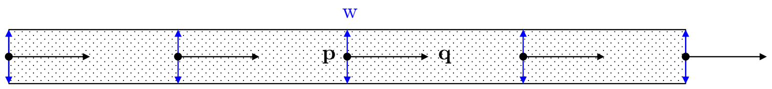
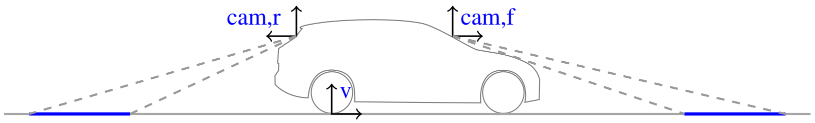


Figure 4.11: **Lane Marker Representation**. Each lane marker is represented by a set of elements. Each element contains a position $\boldsymbol{p}$, orientation $\boldsymbol{q}$ and width w.

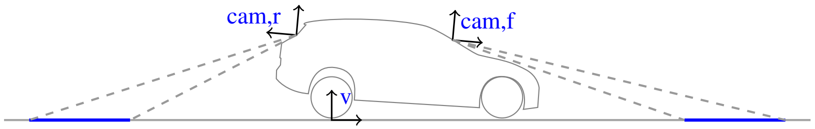
Map Matching

Map matching refers to the task of matching recorded geographic objects or ego-coordinates to a model or map of the real world. Following the scope of this thesis, the term *map matching* will be used to denote the alignment of detected lane markers from a camera system, presented in Sec. 4.2.3, onto a digital map of lane markers, presented in Sec. 5.1.3. The objective of the map matching algorithm is to compute map-relative pose estimates that can be included in the pose graph. Landmark detections from monochrome cameras, expressed in image coordinates, lack the information of spatial positions. Without further processing the detections can only be restricted to a light ray in the camera coordinate frame, as depicted in Fig. 4.3. Various approaches exist, to overcome this issue. In presence of overlapping visual ranges, stereo vision algorithms can be applied to estimate 3D positions of image features. Although a stereo camera setup was suspended priorly, due to cost reduction reasons, a 3D pose estimate of a lane marker can be computed by Structure from Motion, where subsequent images are used as a virtual stereo camera setup, recorded at different times. These approaches are feasible when an exact odometry estimate is present, to precisely estimate the virtual camera alignment, necessary to compute the essential matrix of stereo vision algorithms. Nonetheless, this approach is difficult to apply on highway lane marker detections as their features are poorly conditioned in longitudinal directions, especially for solid lane markers.
It is a reasonable assumption that all lane markers are are on a ground plane, as highway roads have a negligible vertical curvature. Hence, light rays can be projected on a ground plane, to compute lane marker positions in the vehicle v frame, as shown in Fig. 4.12a. However, these approaches are sensitive to imprecise esti-

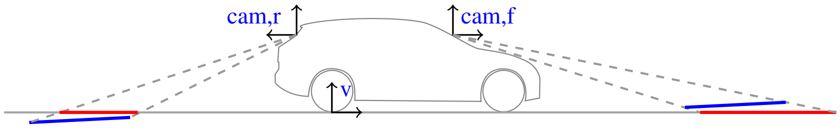
mations of the ground plane in camera coordinates. Neglecting considerable pitch and roll angles of the *icr* frame towards the vehicle *v* frame, as depicted in Fig. 4.12b and Fig. 4.12c, yield erroneous projections of the light rays on the ground plane.



(a) Lane marker detection during no or marginal accelerations with negligible pitch angles.



(b) Lane marker detection during moderate to full break application with perceptible pitch angles.



(c) Projection of lane marker detection light rays, recorded with perceptible pitch angles (blue), on a static ground plane, assuming a zero pitch angle, results in distorted lane marker detections (red).

Figure 4.12: **Light Ray Projection on a Static Ground Plane**. Neglecting perceptible pitch angles results in distorted lane marker projections.

Fig. 4.13 shows the projection error of front and rear camera lane marker detections on a ground plane when alternating the pitch angle of the *icr* frame towards the vehicle *v* frame. The camera intrinsic and extrinsic parameters are taken from the vehicle setup presented in Sec. 5.1. The projection error increases with increasing distances of lane marker detection to the *icr* frame.
In absence of lane-relative or absolute pitch and roll angle estimators, lane markers can be projected on a tilted ground plane, assuming that all markers are parallel. Wang et al. [90] present an approach to compute parallel lines on a ground plane by utilizing the perspective effect of parallel lines in the image domain. Though this assumption is valid for most highway and interstate lanes, it is substantially violated in highway intersections, exit and entry lanes, constructions zones and whenever lanes begin or end.

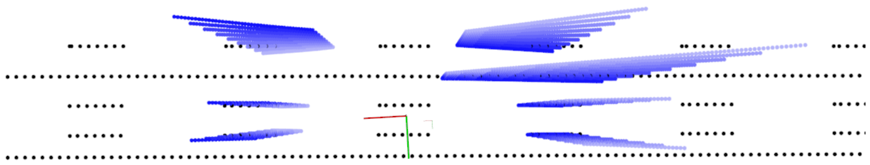


Figure 4.13: **Light Ray Projection**. Light rays of detected lane markers are projected on a ground plane. The pitch rotation of the ICR frame towards the vehicle v frame is linearly increased from -2 *deg* (blue) to 2 *deg* (light blue).

Priorly recorded digital maps can be used to estimate lane-relative pitch and roll angles. The presented map matching algorithm, outlined in Alg. 10, estimates the map-relative pose as well as the pitch and roll angle of the vehicle chassis towards the vehicle body by aligning lane marker light rays with the digital map. In the following, all lane marker detections are represented by a set $\mathcal{S}_{\text{ray}}$ of n light rays $\boldsymbol{r}$

$$\mathcal{S}_{\text{ray}} = \{\boldsymbol{r}_1^{cam}, \ldots, \boldsymbol{r}_n^{cam}\} \tag{4.36}$$

The light rays are represented as normalized direction vectors $\boldsymbol{r}^{cam,f/r}$ through the front or rear camera frame origin. Both front and rear camera light rays are transformed to UTM u-coordinates

$$\boldsymbol{r}_i = \boldsymbol{p}_v^u + \boldsymbol{q}_v^u(\boldsymbol{q}_{icr}^v \boldsymbol{p}_{cam}^{icr} + s\boldsymbol{q}_{icr}^v \boldsymbol{q}_{cam}^{icr} \boldsymbol{r}_i^{cam}) \tag{4.37}$$

where s is the scaling variable of the normalized light ray $\boldsymbol{r}$. The only Degree Of Freedom (DOF) for the optimization are the yaw angle in $\boldsymbol{q}_v^u$, the pitch and roll angle in $\boldsymbol{q}_{icr}^v$ and the x- and y- component of the vehicle position $\boldsymbol{p}_v^u$.
Using an initial pose estimate, lane marker points $\boldsymbol{m}$ are sampled from the digital map in a wide surrounding area. The map samples are equally spaced and represented by a set $\mathcal{S}_{\text{map}}$ of k points in UTM coordinates

$$\mathcal{S}_{\text{map}} = \{\boldsymbol{m}_1^u, \ldots, \boldsymbol{m}_k^u\} \tag{4.38}$$

The digital map and lane marker detections are sampled equally to reduce quantization noise. Ultimately, quantization errors from the sampling cancels out if equally distributed. The distance of light rays to the map point cloud is computed via

$$\boldsymbol{n}_d = \boldsymbol{r} \times (\boldsymbol{m} - \boldsymbol{p}_{cam}) \times \boldsymbol{r} \tag{4.39}$$

$$d_{\text{ray}}(\boldsymbol{m},\boldsymbol{r}) = \frac{||\boldsymbol{r} \times (\boldsymbol{p}_{cam} - \boldsymbol{m})||}{||\boldsymbol{r}||} \tag{4.40}$$

where $\boldsymbol{p}_{cam}$ is the front or rear camera frame position, d is the distance between the light ray $\boldsymbol{r}$ and digital map point $\boldsymbol{m}$ and $\boldsymbol{n}_d$ the distance direction.

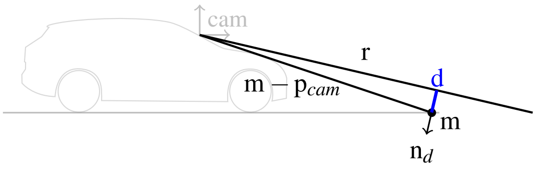


Figure 4.14: **Distance Computation Between Digital Map and Lane Marker Detector**. The distance d (blue) between a lane marker detection light ray $\boldsymbol{r}_j$ and corresponding map point $\boldsymbol{m}_i$ is shown.

For all detected lane marker rays $\boldsymbol{r}$, a corresponding map marker point $\boldsymbol{m}$ is searched by Nearest Neighbor Search (NNS) using the corresponding distance d and stored in two sorted lists $\tilde{\boldsymbol{m}}$ and $\tilde{\boldsymbol{r}}$ of size l. The resulting distance of all light rays is

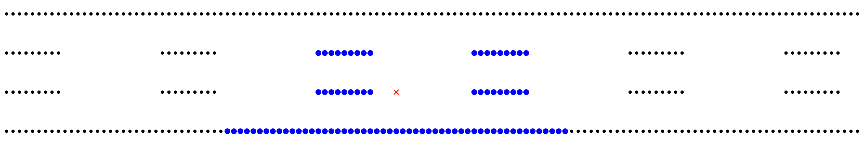
$$d_{\text{map}} = \frac{1}{l} \sum_{j=1}^{l} |d_{\text{ray}}(\tilde{\boldsymbol{m}}_j, \tilde{\boldsymbol{r}}_j)| \tag{4.41}$$

The map-relative pose P_i can now be optimized using the distance measure d_{map}, between the light rays and map points of lane markers. During the optimization process, the sorted lists $\tilde{\boldsymbol{m}}$ and $\tilde{\boldsymbol{r}}$ need to be adjusted. The recursive process is outlined in Alg. 9.

Algorithm 9 Light Ray Alignment

```
Input: $\hat{P}_i$, $\mathcal{S}_{\text{ray}}$, $\mathcal{S}_{\text{map}}$
Output: $P_{i,\text{opt}}$
Search map point correspondences for all rays of $\mathcal{S}_{\text{ray}}$ in $\mathcal{S}_{\text{map}}$
while Correspondence lists $\tilde{\boldsymbol{m}}$ and $\tilde{\boldsymbol{r}}$ have changed do
    Optimize $P_i$ by minimizing Eq. 4.41
    Compute distance of each ray in $\mathcal{S}_{\text{ray}}$ to points in $\mathcal{S}_{\text{map}}$ using Eq. 4.40
    assign new correspondences in sorted lists: $\tilde{r}$ and $\tilde{m}$
```

For good initial estimates Alg. 9 converges. However, two problems arise in general. Fig. 4.15 shows the scalar field of distances between the lane marker detections and map lane markers when shifting the vehicle position through the map. It can be seen that there are multiple local minima present. It is evident that the periodicity of highway lane markers induces ambiguities. Initial estimates on wrong lanes or with a longitudinal position error bigger than the respective marker period, can lead the optimization solver to converge to another local minimum. This holds especially for point set registration techniques, such as ICP or Robust Point Matching. The figure also shows smaller periodic minimum, resulting from aliasing effects, as the sampling rate of the map and lane marker detection are finite and equal. Especially for low-cost GNSS receivers together with digital map offsets, an initial estimate of below half a lane width[11] cannot be permanently guaranteed, specifically in perturbed GNSS environments, such as multipath scattering. Although approaches to track multiple driving lanes using multi-object Bayes filters have been presented [31], these approaches only resolve lateral ambiguities, impose the necessity of a separate module and cannot be used to resolve a broader range of ambiguous pose estimates, e.g. from ambivalent point cloud registrations.


(a) Lane marker detections (blue) of the digital map (black) at the vehicle position (red).

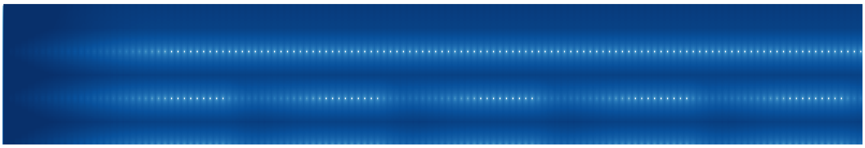

(b) Distance of lane marker detections towards the map when shifting the vehicle position.

Figure 4.15: **Scalar Field for Map Matching**. The scalar field in (b) is created by shifting the vehicle pose (red cross) through the map and accumulating the distance of each lane marker point (blue) to its nearest map marker point (black).

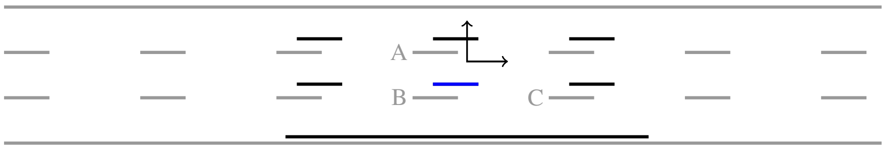
Another problem emerges during evasive vehicle maneuvering, with high lateral and longitudinal accelerations, resulting in perceptible pitch and roll angles, as depicted in Fig. 4.12. High pitch and roll angles and poor initial pose estimates lead to assignment of lane marker rays to multiple map lane markers. The priorly presented light ray and point alignment algorithm Alg. 9 is sensitive to ambiguous

[11] Approximately 1.75 m

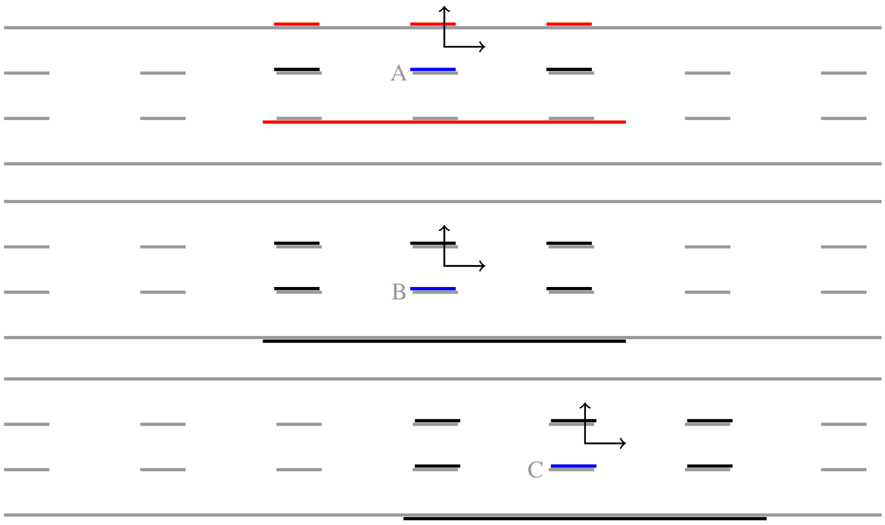
assignments and tends to be stuck in local minimum. Therefore, correspondences between detected and map lane markers are selected and assessed in a prior step. For each dashed and solid lane marker detection in the set $\mathcal{S}_{\text{marker}}$ a subset of valid dashed and solid map lane marker candidates from $\mathcal{S}_{\text{map}}$ is assigned by projecting the lane marker rays on a ground plane, using an initial pose estimate $\hat{P}_i$, and spaciously selecting its closest map lane markers. Each candidate pair is aligned by adjusting the pose estimate, as depicted in Fig. 4.16. For all remaining detected lane markers, using the adjusted pose estimate, the distance to their nearest map lane marker is computed. Each correspondence pair is assessed by checking how many detected and map lane markers align well for its pose estimate. The worst pairs are removed iteratively until all pairs are sufficiently supported by the remaining pairs. This procedure can result in ambiguities if solid lanes are concealed in multi-lane environments motivated in Fig. 1.2. The matching algorithm will then return multiple pose estimates. It is the solvers task to select the right map match from *k* valid matches.

Algorithm 10 Map Matching

```
Input: $\hat{P}_i$, $\mathcal{S}_{\text{lane}}$, $\mathcal{S}_{\text{map}}$
Output: $P_i$
Project detected lane markers $\mathcal{S}_{\text{lane}}$ on a ground plane using $\hat{P}_i$
Tolerantly search lane marker pairs in $\mathcal{S}_{\text{map}}$
repeat
    for all candidate pairs do
        Update $P_i$ by aligning candidate pair
        Update remaining markers $\mathcal{S}_{\text{lane}}$ using $P_i$
        Search new map pairs for remaining markers
        Sum up error of remaining marker pair distances
    if highest candidate pair error exceeds threshold then
        remove candidate
until no candidate removed
repeat
    align all candidates pairs using Eq. 9
    if highest alignment error of candidate exceeds threshold then
        remove candidate
until No candidate removed
Check solutions
if All solutions valid then
    Return solutions
```



(a) For each detected lane marker (blue) from the lane marker perception (black) map correspondences (A, B, C) are computed using NNS.



(b) The selected marker (blue) is aligned with its map correspondences (A, B, C) and cross-validated using the remaining detected markers. For each remaining markers the smallest distance to their next map lane marker is computed. A is a nonvalid match, as it is supported by two markers (black) but rejected by four markers (red), while B and C are valid matches.

Figure 4.16: **Exclusion of False Lane Marker Detections**.

Distance

It is possible to directly input the solver pose estimate as $\hat{P}_i$ in Alg. 10 and return the distance of detected lane marker light rays to the digital map samples. However, the evaluation of Alg. 10 is computationally demanding and has to be evaluated several times, for each incremental pose update by the optimizer. Thus, the Mahalanobis distance

$$d_{\text{map},k}(P_i) = \mathcal{I}_{\text{map}} \begin{bmatrix} \boldsymbol{p}_i - \boldsymbol{p}_{\text{map},i,k} \\ vec(\overline{\boldsymbol{q}}_{\text{map},i,k} \cdot \boldsymbol{q}_i) \end{bmatrix} \tag{4.42}$$

to the aligned pose estimates will be computed instead. $\mathcal{I}_{\text{map}}$ is the 3x3 information matrix for the map matching error. k denotes the map match if the marker pair assignment is ambiguous. The return value of the residual for multiple map matches is presented in Sec. 4.3.

Lane Marker Matching

Since digital map content can be obsolete or erroneous due to construction work, it is necessary to provide a map-independent pose estimate as a fall-back solution. Multiple pose graphs will be solved simultaneously to provide fallback solutions. When digital maps are permanently corrupted, lane-relative pose estimates can help the lateral and longitudinal controller to perform inferior maneuvering task, such as lane centering or emergency stops. To estimate lane-relative poses robustly, subsequent lane marker detections need to be aligned. The lane-relative pose estimation from these aligned detections is presented in Sec. 4.5.
Aligning light rays of subsequent lane marker detections is similar to the priorly presented alignment of map lane markers with lane marker detections but has two essential differences. First, computing the distance between two subsequent sets of light rays is not feasible as the residual is zero when subsequent camera origins overlap, resulting in a zero movement. Second, finding correspondences between subsequent detections is a lot easier. It is sufficient to estimate incremental pose delta $\Delta P_{i,i+1}$ to assign subsequent marker pairs, removing the necessity of global pose estimates, which are more inaccurate and often subject to offset errors. An initial pose delta is estimated by the presented odometry estimation and is within a few centimeters. Marker pairs can be robustly assigned by NNS, and do not need to be cross-validated with other pairs. Therefore, ambiguities do not have to be taken into account.
The alignment of subsequent rays is impractical as the error is zero when camera origins overlap. Hence, the light rays of both images $\mathcal{S}_{\text{ray},i}$ and $\mathcal{S}_{\text{ray},i+1}$ are projected on a rotated ground plane, using their body to chassis rotation $\boldsymbol{q}^{v}_{icr,i}$ and $\boldsymbol{q}^{v}_{icr,i+1}$. This results in two sets of 3D points $\mathcal{S}_{\text{ray},p,i}$ and $\mathcal{S}_{\text{ray},p,i+1}$. Lane markers, displayed in Fig. 4.11, are recorded with equal sample rates. Therefore the assignment of dashed marker sampling points can be done directly. For partially detected and solid lane markers the projected light rays are assigned using their distance. The assigned

rays are stored in two sorted lists $\tilde{\mathcal{S}}_{\mathrm{ray},i}$ and $\tilde{\mathcal{S}}_{\mathrm{ray},i+1}$ of size l. The optimization algorithm that aligns subsequent markers, is outlined in Alg. 11.

Algorithm 11 Lane Error Residual

```
Input: $\Delta\hat{P}_{i,i+1}$, $\hat{\boldsymbol{q}}^{v}_{icr,i}$, $\hat{\boldsymbol{q}}^{v}_{icr,i+1}$, $\mathcal{S}_{\mathrm{ray},i}$, $\mathcal{S}_{\mathrm{ray},i+1}$
Output: $\Delta P_{\mathrm{lane},i,i+1}$, $\boldsymbol{q}^{v}_{icr,i}$, $\boldsymbol{q}^{v}_{icr,i+1}$
while optimality conditions not reached do
    Adjust $\Delta P_{i,i+1}$, $\boldsymbol{q}^{v}_{icr,i}$ and $\boldsymbol{q}^{v}_{icr,i+1}$
    Project $\mathcal{S}_{\mathrm{ray},i}$ on a rotated ground plane using $\Delta P_{i,i+1}$ and $\boldsymbol{q}^{v}_{icr,i}$
    Project $\mathcal{S}_{\mathrm{ray},i+1}$ on a rotated ground plane using $\boldsymbol{q}^{v}_{icr,i}$
    Assign projected points within corresponding lane markers
    Compute distance between assigned points
return optimized $\Delta P_{\mathrm{lane},i,i+1}$, $\boldsymbol{q}^{v}_{icr,i}$, $\boldsymbol{q}^{v}_{icr,i+1}$
```

The optimized rotation quaternions $\boldsymbol{q}^{v}_{icr,i}$ and $\boldsymbol{q}^{v}_{icr,i+1}$ and the delta position $\Delta P_{\mathrm{lane},i,i+1}$ describes the transformation between two lane marker detections. Thus, the residual from the alignment of subsequent lane markers is represented as an edge in the pose graph, similar to Eq. 4.35

$$\Delta P_{\mathrm{lane},i,i+1} = \{\boldsymbol{p}_{\mathrm{lane},i,i+1}, \boldsymbol{q}_{\mathrm{lane},i,i+1}\} \tag{4.43}$$

$$d_{\mathrm{lane}}(P_i, P_{i+1}) = \mathcal{I}_{\mathrm{lane}} \begin{bmatrix} \overline{\boldsymbol{q}}_i \cdot (\boldsymbol{p}_{i+1} - \boldsymbol{p}_i) - \boldsymbol{p}_{\mathrm{lane},i,i+1} \\ vec(\boldsymbol{q}_{\mathrm{lane},i,i+1} \cdot \boldsymbol{q}_i \cdot \overline{\boldsymbol{q}}_{i+1}) \end{bmatrix} \tag{4.44}$$

Where $\mathcal{I}_{\mathrm{lane}}$ is the information matrix for the pose error for subsequent lane marker alignments.

4.3 Loss Functions

Nonlinear Least Squares problems are sensitive to measurement outliers. Due to the quadratic error term, outliers tend to significantly shift the solution from the true value. Thus it is feasible to implement a composition of the residual with a loss function ρ, to delimit the influence of outliers. The most common loss functions are

$$\text{Huber}: \qquad \rho(s) = \begin{cases} s & s \leq 1 \\ 2\sqrt(s) - 1 & s > 1 \end{cases} \tag{4.45}$$

$$\text{Cauchy}: \qquad \rho(s) = \log(1+s) \tag{4.46}$$

$$\text{Acrtan}: \qquad \rho(s) = \arctan(s) \tag{4.47}$$

Another loss function is presented by Suenderhauf et al. [79], where the optimization solver can switch off constraints. A switching variable, subject to optimization, is introduced that allows to linearly decrease the constraint cost while simultaneously increasing a constant penalty. If constraint costs exceed a penalty value, the solver will eventually turn off the constraint. A similar effect can be achieved by applying a switch loss function

$$\text{Switch}: \qquad \rho(s) = \begin{cases} s & s \leq a \\ a & s > a \end{cases} \tag{4.48}$$

where a is a tuning parameter, which can be chosen using a χ^2-value if the respective variable is normally distributed. Applying the switch loss function has a similar effect than rejecting new measurements using NIS-testing [4].
Fig. 4.17 displays the loss functions. When applied on a quadratic residual, the Huber loss is convex as its slope is monotonously increasing. Cauchy and arctan loss functions are strictly quasi-convex. For the GNSS and odometry residual the Cauchy loss will be applied as it showed efficient containment of outlier impact. The switch loss function is quasi-convex, but not strictly, and has no distinct derivative at a.
When ambiguous measurements are present, adding the distances to all measurements will shift the optimal solution to a point that may not be conform with any measurement, as depicted in Fig. 4.18.
In presence of ambiguous map matches, the distance of pose P_i, which is subject to optimization, to its nearest map match $M_{\text{map},i,k}$ will be returned, which has similar properties than the switch loss. Fig. 4.18 depicts the residual when applying a trivial loss function on the pose distance. The presented residual function is not convex. Theoretically, it cannot be assured that the optimization solver converges to the global minimum. However, all local minima are global minima. Furthermore, it will be assumed that ambiguous matches are not constantly present and the solver will eventually select those matches that are sufficiently supported by prior and posterior matches connected through odometry constraints.

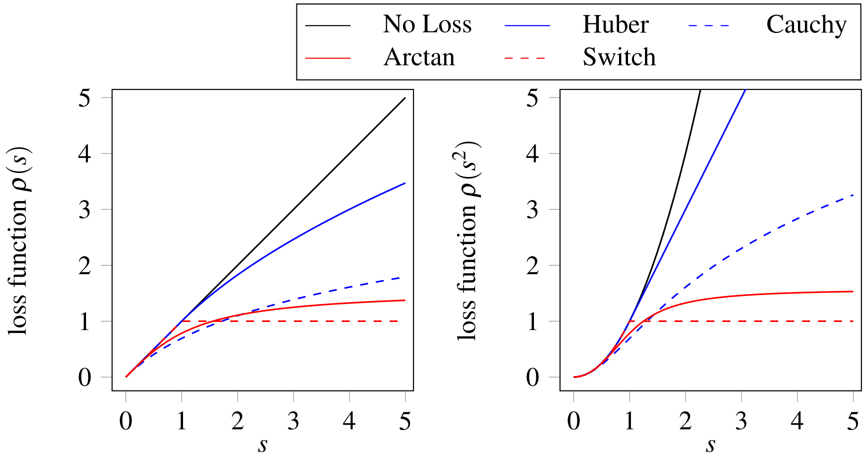


Figure 4.17: **Loss Functions**. The left image displays the loss functions, while the right image depicts the loss function of a squared input, to display the convexity of a quadratic error term.

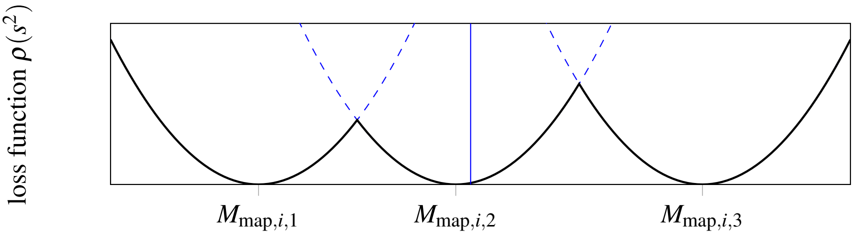


Figure 4.18: **Loss Function for Ambiguous Measurements**. The map matching loss function (black) for a pose estimate P_i returns the distance to the nearest match $M_{\text{map},i}$ (blue, dashed). The blue vertical line depicts the minimal distance to all three matches.

4.4 Constraints and Parameterizations

Constraints and parameterizations assist the optimization solver in finding valid solutions and are more convenient than checking the optimization solution in a subsequent validation step. Setting entire parameter blocks constant and adding parameterizations can further increase the optimization speed.

Quaternion Parameterization

All orientations are represented by unit quaternions due to their compact storage, efficient composition and stable spherical interpolation, for details see Sec. 2.1.1. Quaternions can be computed from a rotation axis $\boldsymbol{u}$ and rotation angle α, as depicted in Eq. 2.7. As the rotation axis $\boldsymbol{u}$ is normalized, the quaternion has three DOF, which is apparent as a 4D quaternion represents a 3D orientation. Thus, its four values cannot be altered independently by the optimization solver. Thus, a parameterization is needed for each quaternion parameter block. Although it is possible to add a parameter constraint within each quaternion, to fulfill the properties in Eq. 2.7, it is more efficient to restrict the quaternion update[12]. The idea is to only allow orthogonal updates to quaternions. Let $\Delta\boldsymbol{u}$ be the update quaternion's Euler axis and $|\Delta\boldsymbol{u}|$ the corresponding total rotation angle. An update quaternion $\Delta\boldsymbol{q}$ is defined, using Eq. 2.7, by

$$\Delta\boldsymbol{q} = \begin{bmatrix} cos(|\Delta\boldsymbol{u}|) \\ sin(|\Delta\boldsymbol{u}|)\frac{\Delta\boldsymbol{u}}{|\Delta\boldsymbol{u}|} \end{bmatrix} \tag{4.49}$$

The updated quaternion $\tilde{\boldsymbol{q}}$ is computed via the Hamilton product

$$\tilde{\boldsymbol{q}} = \Delta\boldsymbol{q} \otimes \boldsymbol{q} \tag{4.50}$$

Constraints

A 3D representation is used throughout this work, to represent vehicle position and orientation. However, to decrease the optimization problem complexity and increase the solver speed, positions and orientations will be restricted to a 2D space. Given most in-series sensor setups, especially the one used throughout this work in Sec. 5.1, it is hard to estimate global pitch and roll angles as they can neither be observed directly nor decoupled stably from vehicle to chassis rotations. Although this is a strong assumption for missilery or aerial rovers, restricting the position and orientation of a terrestrial rover, or passenger cars in particular, to a 2D space introduces negligible errors.

The z-component of all global positions $\boldsymbol{p}_i$ in the set of poses $\mathcal{P}_{\text{map}}$ and $\mathcal{P}_{\text{gnss}}$ are set to zero and kept constant during the optimization process. Similarly, all global orientations $\boldsymbol{q}_i$, of both sets, will only allow nonzero yaw angles, by keeping the quaternion rotation vector $\boldsymbol{u}$ to the unit z-axis vector. This is achieved by setting q_x and q_y to zero and keeping them constant throughout the optimization.

For the vehicle chassis to body rotation $\boldsymbol{q}^{v}_{icr,i}$ for poses in the set $\mathcal{P}_{\text{map}}$, the yaw

[12] For details refer to the CERES[13] manual

angle will be set to zero and kept constant, as the damping system between the vehicle chassis towards its body does only allow negligible yaw angles. This is implemented by setting q_z to zero, keeping the rotation vector in the (x,y)-plane. The pitch and roll angle of $\boldsymbol{q}^{v}_{icr,i}$, estimated by aligning lane markers with the map, is constrained to be within -5 deg to 5 deg.

4.5 Lane-Relative Pose Estimation

Estimating lane-relative poses from previously aligned lane marker detection can be used to provide fall-back pose estimates. These fall-back estimates are of utmost importance during permanent corruption of map-relative pose estimates due to invalid digital map contents or poor initials. The lateral and longitudinal controller can then perform degraded maneuvering, such as lane centering or emergency stops, to keep the vehicle in a safe operation state or require a hand-over request to the vehicle passengers.
Given a point cloud of priorly aligned lane marker detections in UTM coordinates, the objective is to estimate lane-relative vehicle poses. Initially, the respective point cloud is converted into v coordinates and distant points are removed.
The initial step is to cluster the lane marker point cloud to separate lane markers. Distinctively separated dashed and solid lane markers can be clustered by basic NNS algorithms. However, in presence of false positives as well as when lines split or merge, NNS algorithms are deficient.
Bo et al. [14] presented an approach that robustly handles clustering with intersections. A Delaunay triangulation, where long edges are removed, is used to generate the α-shape of the point cloud. The α-shape is thinned until only a skeleton representation remains. The resulting skeleton is cleaned to remove spurious tails and merge crossing necks similar to [64]. The cleaned skeleton is converted in a weighted graph and its remaining vertices are connected by examining turning angle and curve energy between vertices. Once the graph segments are connected, the initial data points are assigned to its nearest segment. The clustering procedure is depicted in Fig. 4.19.
Third-order polynomials are fitted through segmented point clouds by ridge regression. Lane markers are connected, by examining their polynomial representation and the RMSE between lane markers, to estimate lane boundaries. Points of connected lane markers are assigned to the corresponding lane boundary. Again, third-order polynomials are fitted through the points, to estimate the lane boundary shape, using ridge regression. By taking lane boundary polynomials instead of sin-

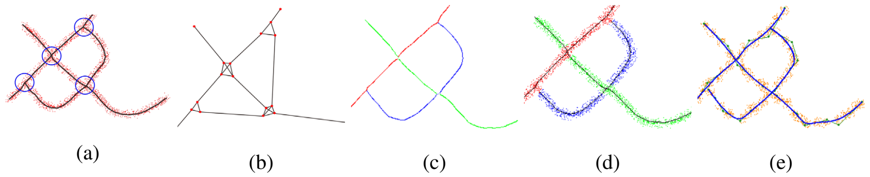


Figure 4.19: **Clustering of Detected Lane Marker Point Clouds**. The images show the data point segmentation pipeline presented by Bo et al. [14]. Point clouds (a) are converted to a skeleton representation (b) which is segmented (c). These segmented skeletons are used to cluster the point clouds (d) and to fit B-splines (e).

gle lane marker polynomials, increases the robustness and accuracy of lane-relative pose estimation.
The distance to lane boundaries is approximated, by sampling from the respective polynomial from the minimal to the maximal x value of the assigned points and computing the distance to these samples. The closest left and right lane boundary is chosen by using their approximated distance, the intersection at the y-axis and validating the distance between the lane boundaries to be conform within the highway guideline.
The intersection of both lane boundary normals with the other lane boundary is computed. The mean of the resulting pair of points and orientations are computed to approximate the lane center. Based on these estimations, the lateral distance and orientation of the vehicle pose towards the estimated lane center is computed. The lane-relative pose estimation procedure is outlined in Fig. 4.20.

4.6 Integrity Monitoring

Monitoring the integrity and accuracy of sensor data is of key importance for data fusion techniques to provide faultless estimates. Especially when vital systems rely on the integrity of these estimates. For safe maneuvering of autonomous vehicles, it is of utmost importance for the vehicle controller to perceive robust and accurate pose estimates. However, in real-world applications sensors can be temporarily disturbed, leading to imprecise or corrupted estimates. Thus, it is important to further provide information about the pose estimate accuracy and integrity to help the controller decide when to switch to fall-back solutions. Fall-back pose estimations, such as lane-relative pose estimates instead of global pose estimates, can significantly help the controller to return to safe driving conditions, such as lane centering for hand-over requests or emergency stops.

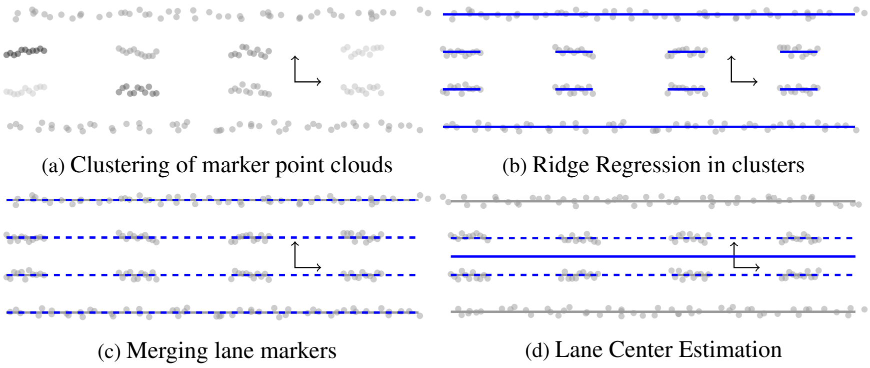

(a) Clustering of marker point clouds (b) Ridge Regression in clusters (c) Merging lane markers (d) Lane Center Estimation

Figure 4.20: **Lane-Relative Pose Estimation**. a) Priorly aligned lane marker detections are clustered using [14]. b) Third-order polynomials are fitted through these clusters using ridge regression. c) Lane Boundaries are estimated by comparing polynomials of all lane markers. d) The left and right lane boundaries are used to estimate the lane center and lane-relative pose.

This chapter will present a generic approach to estimate the accuracy of odometry and pose estimates. To decrease the computational load and algorithm complexity, it will be assumed that all sensor measurements are normally distributed and the sensor accuracy will be represented by covariance matrices. In a preliminary study a χ^2 test has been used to support this assumption given the vehicle's sensor measurements compared to a ground truth system with negligible uncertainty. Further limitations, such as the amount of required estimates, depending on the correlation between estimates, will be discussed.
Based on these covariance matrices, outliers in odometry estimates are rejected using a χ^2-test. The remaining estimates will be fused using the composite probability of MND as presented in Sec. 2.2.2. The resulting robust odometry estimate will be used to assess the optimized poses of the pose graph optimization. A variety of pose graphs are solved simultaneously, each with a different subset of sensors, to provide fall-back solutions.

4.6.1 Covariance Estimation

This section focuses on estimating uncertainties of odometry and pose estimates presented in Sec. 4.2. It will be assumed that all estimates are normally distributed. MNDs are represented by a mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, as presented in Eq. 2.23. Given the samples $\boldsymbol{x}_i$ of MNDs, the mean vector and covariance matrix is computed with

$$\tilde{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i \tag{4.51}$$

$$\tilde{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i-\boldsymbol{\mu})(\boldsymbol{x}_i-\boldsymbol{\mu})^{\mathrm{T}} \tag{4.52}$$

$$= \frac{1}{n+1}\sum_{i=1}^{n}(\boldsymbol{x}_i-\tilde{\boldsymbol{\mu}})(\boldsymbol{x}_i-\tilde{\boldsymbol{\mu}})^{\mathrm{T}} \tag{4.53}$$

The estimated mean $\tilde{\boldsymbol{\mu}}$ follows a Student's t-distribution around the true mean $\boldsymbol{\mu}$. For an infinite amount of samples $n \to \infty$ the estimated mean converges to the true mean $\tilde{\boldsymbol{\mu}} \to \boldsymbol{\mu}$. Eq. 4.52 uses the true mean $\boldsymbol{\mu}$ to estimate the covariance matrix. However, the true mean is usually unknown and approximating the true mean using a high-precision ground truth is infeasible. Eq. 4.53 computes the covariance matrix from an estimated mean $\tilde{\boldsymbol{\mu}}$. Using an estimated mean instead of the true mean increases the covariance matrix, as the true mean has the smallest distance normally distributed samples. The term $(n+1)$ accounts for the inaccuracy of $\tilde{\boldsymbol{\mu}}$. The major problem that arises when estimating the uncertainty of pose or odometry estimates is that the mean is time-dependent $\boldsymbol{\mu}(t)$, see Fig. 4.21.

Covariance Estimation of Time-Dependent Estimates

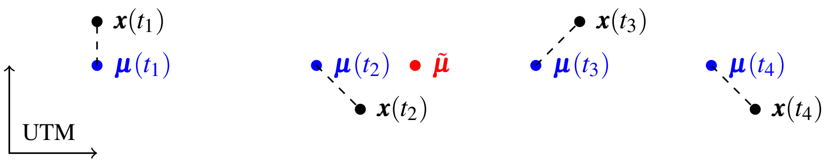


Figure 4.21: **Covariance Estimation with Moving Mean**. The rover's true movement is depicted by $\boldsymbol{\mu}(t)$. To estimate the covariance using Eq. 4.53 the estimates $\boldsymbol{x}(t_i)$ need to be subtracted from the corresponding $\boldsymbol{\mu}(t_i)$, which is usually unknown. Assuming a constant mean and using Eq. 4.51 results in an estimate $\tilde{\boldsymbol{\mu}}$.

Estimating $\boldsymbol{\mu}(t)$ is the core objective of localization problems. Filtering techniques aim to resolve this problem. However, these filters postulate knowledge about the covariance matrices. This causality dilemma can be resolved by priorly estimating the sensor uncertainty using a ground truth sensor that provides precise estimates of

$\boldsymbol{\mu}(t)$ and neglecting the ground truth uncertainty. However, this procedure results in a conservative estimation, is time consuming, expensive, and prone to alignment errors when mounting the sensor. For some sensors, such as GNSS receivers, the uncertainty can be approximated using Dilution Of Precision (DOP) [56] values. DOP values assess the geometrical positions of observed satellites, which largely influence the condition number of the weighted LSQ in Eq. 4.19. Although DOP values help to increase the accuracy of filter algorithms [92], this technique cannot be applied to other sensors.

In presence of multiple sensors, the sensor uncertainties can be computed without knowing the true, time-dependent mean $\boldsymbol{\mu}(t)$. Assuming MNDs for all sensor estimates and given Eq. 2.25, it follows that the difference between sensor estimates, depicted in Fig. 4.22, is normally distributed around a zero mean. The covariance matrix $\boldsymbol{\Sigma}_{AB}$ between sensor A and B can thus be estimated by

$$\tilde{\boldsymbol{\Sigma}}_{AB} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_{AB,i} - \mu_{AB,i})(\boldsymbol{x}_{AB,i} - \mu_{AB,i})^{\mathrm{T}} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_{AB,i})(\boldsymbol{x}_{AB,i})^{\mathrm{T}} \tag{4.54}$$

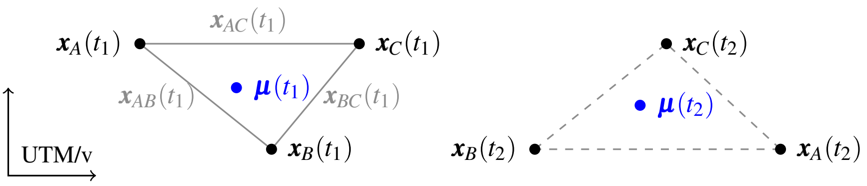


Figure 4.22: **Covariance Estimation Using Multiple Estimates**. Given multiple sensors (A, B, C) and their samples $\boldsymbol{x}_A(t)$, $\boldsymbol{x}_B(t)$, and $\boldsymbol{x}_C(t)$, the sensor covariance matrix can be derived from deltas between sensor estimates $\boldsymbol{x}_{AB}(t)$, $\boldsymbol{x}_{AC}(t)$, and $\boldsymbol{x}_{BC}(t)$ without knowing the true mean $\boldsymbol{\mu}(t)$.

Given Eq. 2.26 and assuming that sensors A and B are uncorrelated, the covariance matrix computed in Eq. 4.54 is the sum of sensor covariance matrices $\boldsymbol{\Sigma}_A$ and $\boldsymbol{\Sigma}_B$

$$\boldsymbol{\Sigma}_{AB} = \boldsymbol{\Sigma}_A + \boldsymbol{\Sigma}_B \tag{4.55}$$

For k uncorrelated sensor estimates, denoted by the letters A-Y, where Y denotes an arbitrary letter, there are $k(k-1)/2$ possible combinations similar to Eq. 4.55. This results in a linear system of equations

$$\underbrace{\begin{bmatrix} \hat{\boldsymbol{\Sigma}}_{AB} \\ \hat{\boldsymbol{\Sigma}}_{AC} \\ \vdots \\ \hat{\boldsymbol{\Sigma}}_{AY} \\ \hat{\boldsymbol{\Sigma}}_{BC} \\ \vdots \\ \hat{\boldsymbol{\Sigma}}_{XY} \end{bmatrix}}_{\boldsymbol{\Sigma}_\Delta} = \underbrace{\begin{bmatrix} \boldsymbol{I}_j & \boldsymbol{I}_j & 0 & 0 & 0 & \cdots \\ \boldsymbol{I}_j & 0 & \boldsymbol{I}_j & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\ \boldsymbol{I}_j & 0 & 0 & \vdots & 0 & \boldsymbol{I}_j \\ 0 & \boldsymbol{I}_j & \boldsymbol{I}_j & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\ 0 & 0 & \cdots & 0 & \boldsymbol{I}_j & \boldsymbol{I}_j \end{bmatrix}}_{\boldsymbol{A}_\Sigma} \underbrace{\begin{bmatrix} \hat{\boldsymbol{\Sigma}}_A \\ \vdots \\ \hat{\boldsymbol{\Sigma}}_Y \end{bmatrix}}_{\boldsymbol{\Sigma}_S} \tag{4.56}$$

where $\hat{\boldsymbol{\Sigma}}$ denotes the column vector of the upper triangular matrix of $\tilde{\boldsymbol{\Sigma}} \in \mathbb{R}^{m\times m}$ with size $j = (m(m-1)/2)$. I_j is the identity matrix of $\in \mathbb{R}^{j\times j}$. Eq. 4.56 can be solved for $k = 3$ and is overdetermined for $k > 3$. The covariance matrices are computed by solving the overdetermined system

$$\boldsymbol{\Sigma}_S = (\boldsymbol{A}_\Sigma^{\mathrm{T}} \boldsymbol{A}_\Sigma)^{-1} \boldsymbol{A}_\Sigma^{\mathrm{T}} \boldsymbol{\Sigma}_\Delta \tag{4.57}$$

The computational complexity to solve Eq. 4.57 is constant in time and covariance matrices in Eq. 4.54 can be updated recursively. Thus the overall computational complexity to estimate covariance matrices does not increase with the number of samples.

Correlated Estimates

Sensor estimates can be correlated, due to the sensor subset or algorithms used to compute estimates. Without loss of generality, if sensor A and B are correlated, given Eq. 2.26, the first line of Eq. 4.56 is modified

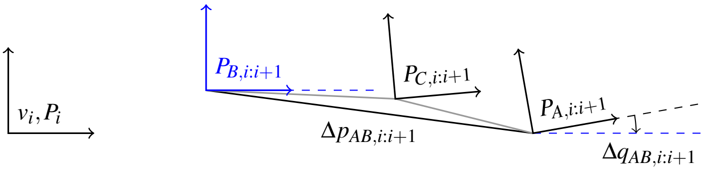
$$\underbrace{\begin{bmatrix} \hat{\boldsymbol{\Sigma}}_{\text{corr}} \\ \vdots \\ \hat{\boldsymbol{\Sigma}}_{XY} \end{bmatrix}}_{\boldsymbol{\Sigma}_\Delta} = \underbrace{\begin{bmatrix} \boldsymbol{I}_j & 2\boldsymbol{I}_j & \boldsymbol{I}_j & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\ 0 & 0 & \cdots & 0 & \boldsymbol{I}_j & \boldsymbol{I}_j \end{bmatrix}}_{\boldsymbol{A}_\Sigma} \underbrace{\begin{bmatrix} \hat{\boldsymbol{\Sigma}}_A \\ \hat{\boldsymbol{\Sigma}}_{AB,\text{corr}} \\ \hat{\boldsymbol{\Sigma}}_B \\ \vdots \\ \hat{\boldsymbol{\Sigma}}_Y \end{bmatrix}}_{\boldsymbol{\Sigma}_S} \tag{4.58}$$

where $\hat{\mathbf{\Sigma}}_{AB,\text{corr}}$ represents the correlation matrix between sensors A and B. For k_{corr} correlations between sensors combinations, the number of unknowns is increased by k_{corr}. Eq. 4.58 can be solved if

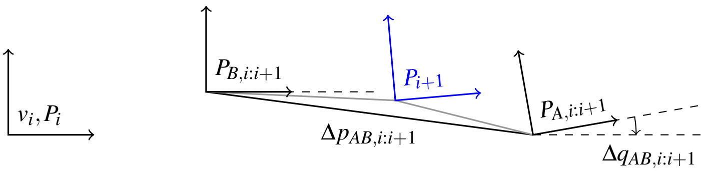
$$\frac{k(k-1)}{2} \geq k + k_{\text{corr}} \tag{4.59}$$

Covariance Estimation of Pose and Odometry Estimates

A necessary condition to solve Eq. 4.56 is that at least three uncorrelated estimates are provided. However, most localization sensor setups, including the setup used herein, do not provide three or more uncorrelated pose estimators. To overcome this issue, pose and odometry estimates are used to estimate covariance matrices by embedding the delta pose of subsequent pose estimates in Eq. 4.56. Fig. 4.23a depicts the creation of odometry sample deltas used to estimate their covariance matrices. Pose estimates can also be included, depicted in Fig. 4.23b.



(a) The delta position Δp and delta orientation Δq between odometry estimates P_A, P_B, and P_C are used to compute the estimators' A, B, and C covariance matrix.



(b) The pose estimate P_{i+1} is transformed in the prior estimate's P_i coordinates and compared to other odometers.

Figure 4.23: **Samples for Covariance Estimation.**

However, while odometry estimates are represented in the vehicle v frame, pose deltas need to be converted from a global coordinate frame to the v frame. This conversion add the pose's P_i rotation uncertainty upon the estimate P_{i+1}. For simplicity and as these rotation uncertainties are significantly smaller than position uncertainties this effect can be neglected.

Estimating the pose estimator's covariance matrix by computing delta poses, removes static offsets of the estimator, depicted in Fig. 4.23b. Offsets occur in GNSS pose estimates due to imprecise ephemeris data, ionospheric and tropospheric delays or in digital maps due to plate tectonics. This seems like a drawback but is actually a desirable feature when estimating global or map-relative poses. For example the accuracy of a map-matching algorithm does not depend on the static offset of the map itself, as long as the pose priors are sufficiently well for the matcher to converge. Although, to fuse pose estimates of various pose estimators with differing offsets, these offsets need to be known or estimated to prevent systematic errors in pose estimates.
This concludes the covariance estimation algorithms. It is assumed that the covariance and correlation matrices are not time-dependent. Although this assumption is not optimal for some sensors, it is an acceptable approximation. Modeling time-dependent effects such as daytime, weather and environmental conditions and how they influence the estimate's uncertainty is extremely complex and not feasible for the presented setup.

4.6.2 Robust Odometry Estimation

To decrease the amount of graph vertices, the odometry estimates presented in Sec. 4.2.2 are fused, resulting in a robust odometry estimation. As motivated in Sec. 4.4 the localization problem will be solved in a 2D space. Therefore, the z-component p_z of the position and the quaternion parts q_x and q_y for 3D odometry estimates will be set to zero. The covariance matrix $\mathbf{\Sigma}_A$ of the odometry estimator A, and for the other estimator correspondingly, will be represented by

$$\mathbf{\Sigma}_A = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & 0 \\ \sigma_{yx} & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\psi^2 \end{bmatrix} \qquad \mathbf{\Sigma}_{p,A} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{yx} & \sigma_y^2 \end{bmatrix}$$

Fig. 4.24 shows the utilized models and sensors used for the robust odometry estimation. Before fusing odometry estimates their covariance matrices need to be computed. Depicted in Fig. 4.23a, covariance matrices of odometry estimates can be calculated by computing the respective difference during a specified time interval and solving Eq. 4.56. For delta orientation or yaw rate uncertainty estimation the single-track model, the WSS odometry, the GNSS odometry, the visual odometry, and the gyroscope are used. Delta position covariance matrices of the odometry estimators are computed using delta poses of the single-track model, the

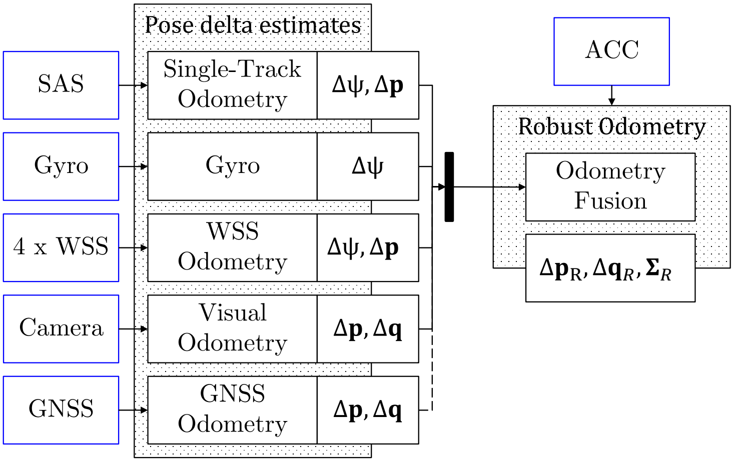

Pose delta estimates
SAS
Gyro
4 x WSS
Camera
GNSS
Single-Track Odometry
$\Delta\psi, \Delta\mathbf{p}$
Gyro
$\Delta\psi$
WSS Odometry
$\Delta\psi, \Delta\mathbf{p}$
Visual Odometry
$\Delta\mathbf{p}, \Delta\mathbf{q}$
GNSS Odometry
$\Delta\mathbf{p}, \Delta\mathbf{q}$
ACC
Robust Odometry
Odometry Fusion
$\Delta\mathbf{p}_\mathrm{R}, \Delta\mathbf{q}_R, \mathbf{\Sigma}_R$

Figure 4.24: **Robust Odometry Estimation**. Given position and orientation deltas and their uncertainties a robust position delta $d\boldsymbol{p}_\mathrm{r}$ and orientation delta $d\boldsymbol{q}_\mathrm{r}$ is computed. The utilized sensor are depicted as blue boxes.

WSS odometry, the visual odometry and the GNSS odometry. For both the orientation and position uncertainty estimates it is assumed that WSS odometry estimates and single-track odometry estimates are correlated as these estimates use the WSS. Their correlation matrices are also estimated.
Once the odometry covariance matrices are computed the odometry estimates can be fused. Before fusing estimates of the respective odometer, outliers are rejected using a χ^2-test on the Mahalanobis distance of the estimate toward the other estimates

$$\chi^2_{\mathrm{odo},p} \geq \frac{1}{n-1} \sum_{j \neq i}^{n} (\Delta\boldsymbol{p}_i - \Delta\boldsymbol{p}_j)^\mathrm{T} \mathbf{\Sigma}^{-1}_{p,j} (\Delta\boldsymbol{p}_i - \Delta\boldsymbol{p}_j) \tag{4.60}$$

$$\chi^2_{\mathrm{odo},\psi} \geq \frac{1}{n-1} \sum_{j \neq i}^{n} \sigma^{-2}_{\psi,j} (\Delta\psi_i - \Delta\psi_j)^2 \tag{4.61}$$

where n is total number of odometry estimators, $\Delta\boldsymbol{p}_i$ and $\Delta\psi_i$ the delta position and orientation estimate of the odometry estimator and $\mathbf{\Sigma}_{p,i}$ and $\sigma_{\psi,j}$ their uncertainties respectively. If Eq. 4.60 is violated, the delta pose estimate is rejected, see Fig. 4.25. This procedure is done correspondingly for delta orientations. All valid position and orientation delta estimates are fused using composite probabilities of MNDs presented in Sec. 2.2.2. The robust odometry algorithm is outlined in Alg. 12.

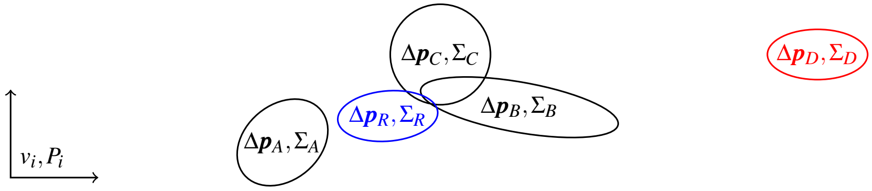

$\Delta \boldsymbol{p}_C, \Sigma_C$
$\Delta \boldsymbol{p}_D, \Sigma_D$
$\Delta \boldsymbol{p}_R, \Sigma_R$
$\Delta \boldsymbol{p}_B, \Sigma_B$
$\Delta \boldsymbol{p}_A, \Sigma_A$
v_i, P_i

Figure 4.25: **Odometry Fusion**. The robust delta position estimate $\Delta \boldsymbol{p}_R$ (blue) results from fusing the delta position estimates $\Delta \boldsymbol{p}_A$ to $\boldsymbol{p}_D$ using their covariance matrices $\boldsymbol{\Sigma}_A$ to $\boldsymbol{\Sigma}_D$. Outliers (red) are rejected by comparing the Mahalanobis distance to other estimates and a χ^2 test.

Algorithm 12 Robust Odometry Estimation

```
Input: $\Delta \boldsymbol{p}_A$-$\Delta \boldsymbol{p}_Y$, $\Delta \psi_A$-$\Delta \psi_Y$, $\boldsymbol{\Sigma}_A$-$\boldsymbol{\Sigma}_Y$
Output: $\Delta \boldsymbol{p}_R$, $\Delta \boldsymbol{q}_R$, $\boldsymbol{\Sigma}_R$
Compute $\chi_j^2$ for all delta positions $\Delta \boldsymbol{p}$ using Eq. 4.60
while Highest $\max_j(\chi_j^2) > \chi^2_{\text{odo}}$ do
    Remove delta position with highest $\chi^2$
    Recompute Eq. 4.60 for remaining pose deltas
Fuse remaining pose deltas to $\boldsymbol{p}_\text{R}$
Compute $\chi_j^2$ for all delta orientations $\Delta \psi$ using Eq. 4.61
while Highest $\max_j(\chi_j^2) > \chi^2_{\text{odo}}$ do
    Remove delta orientation with highest $\chi^2$
    Recompute Eq. 4.61 for remaining orientation deltas
Fuse remaining orientation deltas and convert to quaternion $\boldsymbol{q}_\text{R}$
```

4.6.3 Integrity Monitoring of Pose Estimates

Although large efforts are conducted to ensure reliability of sensor measurements and system state estimations, reliability of all vehicle sensors and algorithms is costly and may even be infeasible for some sensors or algorithms. A practicable approach to overcome this issue is to constantly monitor the integrity between sensor estimates to assess their functionality and provide, if single sensors are permanently corrupted, fall-back estimates. This section will present an approach to monitoring the integrity of pose estimators and optimized pose estimates, given the priorly estimated covariance matrices.

Assessing both sensor pose estimates and optimized poses is indispensable to ensure the integrity of pose estimates and solver convergence. Assuming that the orientation error of the pose estimate P_i is within a few degrees so that the small angle approximation (linearity)

$$\sin\alpha \approx \alpha \tag{4.62}$$

holds and both the pose estimate P_i and robust odometry estimate $P_{R,i:i+1}$ are unbiased estimators, the covariance matrix $\mathbf{\Sigma}_{\mathrm{PR}}$ between the difference of subsequent sensor pose estimates P_i and P_{i+1} towards the robust odometry estimate $P_{R,i:i+1}$

$$d_{p,euler} = \boldsymbol{p}^v_{i+1} - (\boldsymbol{p}^v_i + \boldsymbol{q}^v_i \cdot \boldsymbol{p}^v_{\mathrm{R},i:i+1}) \tag{4.63}$$

can be directly computed from $d_{p,euler}$ (see Sec. 4.6.1). The Mahalanobis distance between subsequent pose estimates and the robust odometry estimate can then be computed

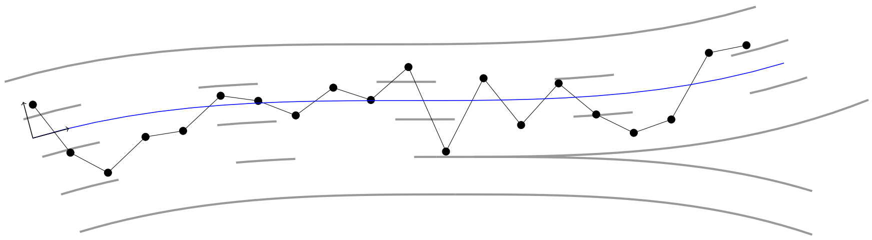
$$d^2_{\mathrm{p}} = (\boldsymbol{p}^v_{i+1} - \boldsymbol{p}^v_i - \boldsymbol{q}^v_i \cdot \boldsymbol{p}^v_{\mathrm{R},i:i+1})^{\mathrm{T}} \mathbf{\Sigma}^{-1}_{\mathrm{p,PR}} (\boldsymbol{p}^v_{i+1} - \boldsymbol{p}^v_i - \boldsymbol{q}^v_i \cdot \boldsymbol{p}^v_{\mathrm{R},i:i+1}) \tag{4.64}$$

$$d^2_{\mathrm{r}} = (\psi^v_{i+1} - \psi^v_i - \psi^v_{\mathrm{R},i:i+1}) \sigma^{-2}_{\psi,\mathrm{PR}} (\psi^v_{i+1} - \psi^v_i - \psi^v_{\mathrm{R},i:i+1}) \tag{4.65}$$

The squared position distance d^2_{p} and rotation distance d^2_{r} is compared to a χ^2-threshold to monitor the sensor integrity. This procedure is performed for all pose estimators presented in Sec. 4.2:

- GNSS pose estimates
- Map matching pose estimates
- Lane marker matching delta pose estimates

Furthermore, three pose graphs are solved simultaneously to increase the robustness and accuracy of the pose estimators. Each pose graph inherits a subset of sensors as listed in Tab. 4.1. Although, fusing GNSS pose estimates and map matches is nonoptimal due to systematic offsets between the digital map and GNSS pose estimates, GNSS delta poses can be included directly in the robust odometry estimation as the offset cancels out. Note that the GNSS pose graph uses a robust odometry estimation without GNSS delta poses. Similarly, fusing subsequent **lane** matches and map matches is nonoptimal as these estimates are highly correlated.


(a) The Mahalanobis distance within pose estimates (black) is computed to assess the sensor integrity.

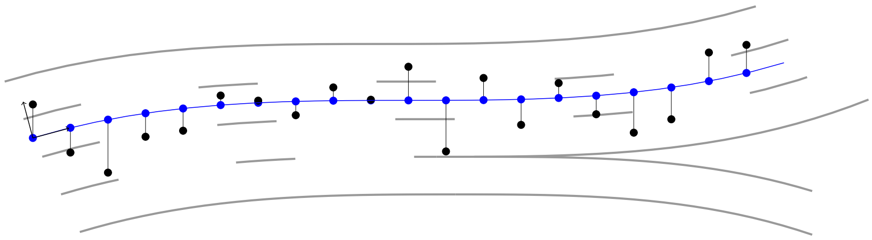
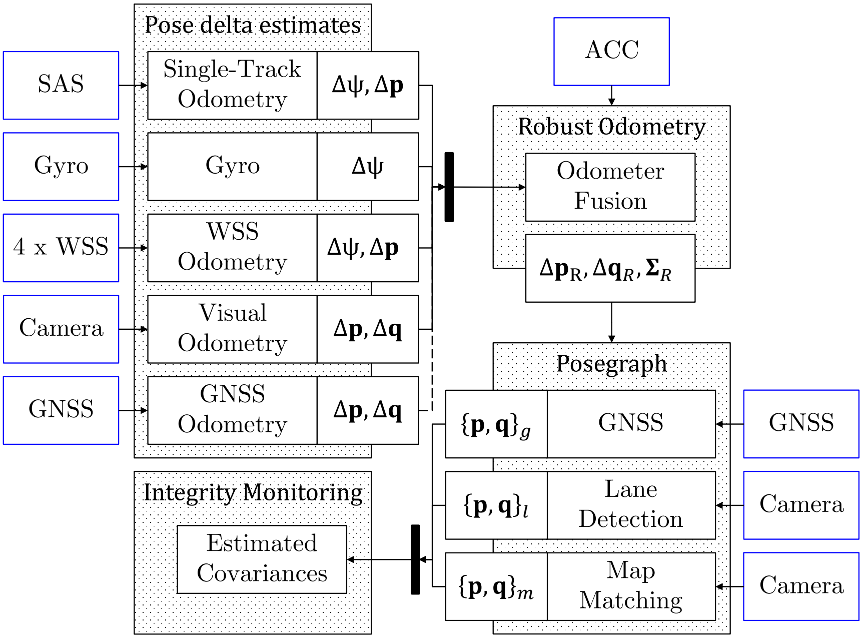

(b) The Mahalanobis distance of optimized pose estimates (blue) towards sensor pose estimates (black) is computed to assess the optimization solution.

Figure 4.26: **Integrity Monitoring of Poses**. The sensor pose estimates and the optimized pose estimates are monitored to ensure that both the sensor estimates are of integrity and the optimization solver converged.

	GNSS	Map	Lane Marker	Odometry
GNSS pose graph	x			x
map-relative pose graph		x		x
lane-relative pose graph			x	x

Table 4.1: **List of Pose Graphs**.

The GNSS pose graph is used to initialize the map matcher. While the map-relative pose graph computes the map-relative pose estimates. If permanent corruptions of map matches or digital map content are detected, the controller may fall back to the lane-relative pose estimate provided by the lane-relative pose graph.



Pose delta estimates
SAS
Single-Track Odometry
$\Delta\psi, \Delta\mathbf{p}$
Gyro
Gyro
$\Delta\psi$
4 x WSS
WSS Odometry
$\Delta\psi, \Delta\mathbf{p}$
Camera
Visual Odometry
$\Delta\mathbf{p}, \Delta\mathbf{q}$
GNSS
GNSS Odometry
$\Delta\mathbf{p}, \Delta\mathbf{q}$
ACC
Robust Odometry
Odometer Fusion
$\Delta\mathbf{p}_R, \Delta\mathbf{q}_R, \mathbf{\Sigma}_R$
Posegraph
$\{\mathbf{p},\mathbf{q}\}_g$
GNSS
GNSS
$\{\mathbf{p},\mathbf{q}\}_l$
Lane Detection
Camera
$\{\mathbf{p},\mathbf{q}\}_m$
Map Matching
Camera
Integrity Monitoring
Estimated Covariances

Figure 4.27: **Integrity Monitoring Scheme**.

4.7 Optimization Properties and Pose Update

Optimization Properties

The presented optimization problem is a constrained nonlinear LSQ problem with a finite dimensional and continuous parameter space and thus belongs to the category of Nonlinear Programming.
All parameter constraints, presented in Sec. 4.4, are equality constraints, except for the inequality constraint of the total pitch and roll angle of $\boldsymbol{q}^{v}_{icr}$. However, the vehicle chassis to body rotation $\boldsymbol{q}^{v}_{icr}$ will be optimized, as outlined in the light ray alignment algorithm Alg. 9, for each single-shot **lane** detection separately, either on a prior **lane** detection or the digital map, depending whether the digital map is utilized or not. The distance to the aligned pose will be included in the pose graph using Eq. 4.42 or Eq. 4.44 respectively. Therefore, the pose estimation optimization problem in Eq. 4.3 is an equality constrained problem. This has the benefit that there is no need to tracking a working set of active constraints, as active-set or interior-point methods do, and the method of Lagrangian multipliers can be

applied.
All presented pose distances compute the Mahalanobis distance of the pose or odometry estimates and are convex functions. However, for the loss functions, only the Huber loss is convex. Cauchy and arctan loss are strictly quasi-convex.The loss function for ambiguous map matches, depicted in Fig. 4.18, is not convex. Therefore, it cannot be guaranteed that the solver converges to the global optimum, except when map matching ambiguities are absent or can be resolved before the optimization progress starts. The derivative for the optimization solver will be computed by using AD. Compared to numerical differentiation, AD has the benefit of computing exact derivatives, see Sec. 2.3.3. Another advantage of AD is that the computation of the objective using dual numbers is faster than reevaluating the objective function to approximate derivatives, as used by numerical differentiation.
A Levenberg-Marquardt (LM) algorithm is used to solve the presented optimization problem. Similarly to Gauss-Newton methods, the LM algorithm estimates the Hessian matrix, which saves a significant amount of computational time. As the LM algorithm uses the same Hessian approximation as Gauss-Newton methods, they locally behave similar. However, LM uses a trust-region algorithm instead of a line-search algorithm internally, making it more robust against rank-deficient or nearly rank-deficient Jacobians. Thus, the LM algorithm finds a solution in many cases, even when further away from the local minimum as Gauss-Newton methods. More details on LM algorithms and numerical optimization can be found in the work of Wright and Nocedal [94].

Pose Update

Camera and GNSS pose estimates can be attained with more or less constant frequencies. However, these frequencies are not sufficiently high for lateral and longitudinal vehicle control algorithms, which need pose updates at constant frequencies of 20 - 100 Hz, depending on the controller. Furthermore, camera and GNSS pose estimates may be invalid or unavailable during short time intervals.
While the application of optimization techniques increases robustness and accuracy of pose estimates, the optimization solver time highly depends on the quality of initial estimates and nonlinearity of the optimization problem. Therefore, the optimization solver either consumes varying amounts of time or must be aborted before reaching a convergence criteria such as a minimal parameter step size. Neither of these properties are desirable for vehicle controllers.
Given a set of poses $\mathcal{P}$, the k-th latest optimized pose P_{opt} with its time stamp t_{opt} will be updated using the robust odometry estimates until the current system time t_{curr} as depicted in Fig. 4.5. Choosing k is a trade-off between the amount of uncertainty thate the odometry estimate introduces versus the increase of accuracy of

an optimized pose with prior and posterior pose estimates. The pose update algorithm is outlined in Alg. 13. To ensure constant pose updates at high frequencies, the optimization algorithm Alg. 14 is performed on a parallel thread towards the optimization solver.

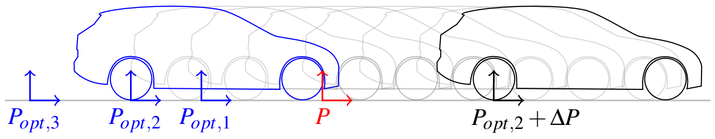


Figure 4.28: **Vehicle Pose Update for the Controller**. The k-th latest pose $P_{opt,2}$ ($k = 2$), of optimized poses (blue) is updated by ΔP from the odometer to provide the current pose estimate (black). There might be more recent poses, which have not been optimized yet (red).

Algorithm 13 Pose Update Thread

```
**Input:** $\mathcal{P}$, odometry estimates, digital map
**Output:** $P_{curr}$
Get the $k$-latest optimized pose and associated time stamp from $\mathcal{P}$
Update the optimized pose to the current time stamp using the odometry
Estimate the covariance and publish the pose $P_{curr}$
```

Algorithm 14 Map-relative Pose Graph Optimization Thread

```
Input: sensor data cache, digital map
Output: $\mathcal{P}$, estimated covariances, aggregated map
if new lane marker detections available then
    Add a new pose to $\mathcal{P}_{\text{map}}$ and remove or fix obsolete poses
    Compute odometry constraints to prior, valid pose from $\mathcal{P}_{\text{map}}$
    if initialization complete then
        Initialize pose estimate with prior pose + odometry
    else
        Initialize pose estimate using GNSS estimate
    Add odometry error term and constraints
    Perform map matching and ray alignment algorithms
    if New match conform with prior matches then
        Estimate covariance of matches
        Add error term to pose graph problem
    else
        Reject match
    Optimize the pose graph
```

5 Experimental Evaluation

To evaluate the presented algorithms in real-world scenarios, a test vehicle was built up and equipped with additional sensors, depicted in Fig. 5.1. A Differential GPS (DGPS) reference station and a high-precision aided INS is used to evaluate the presented algorithms quantitatively. For the evaluation multiple datasets were recorded, both on the OTC test track[1] and public highway track, Frankfurt, Germany. The datasets cover low- and high-dynamic scenarios as well as perturbed scenarios with traffic and differing weather conditions. The datasets are stored as ROS[2] bagfiles and contain raw camera images, CAN data, and raw pseudorange data.

5.1 Pose Estimation Setup

A 4WD Opel Insignia[3] was chosen as sensor platform. A fundamental requirement was to use a near-series, cost-efficient sensor setup. Fig. 5.1 shows the main components of the test vehicle displayed in blue. The sensor positions can be found in Appx. A.1.

5.1.1 Vehicle Sensor Setup

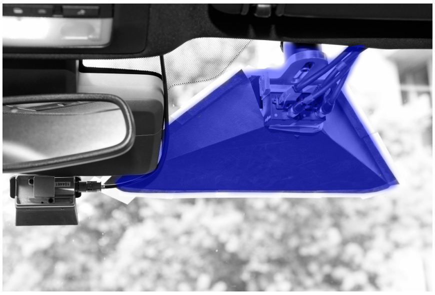
The test vehicle has three dynamic sensors available at 100 Hz on the CAN bus.

- Steering Angle Sensor (SAS)
- Four Wheel Speed Sensors (WSSs)
- 2D Inertial Measurement Unit (IMU)

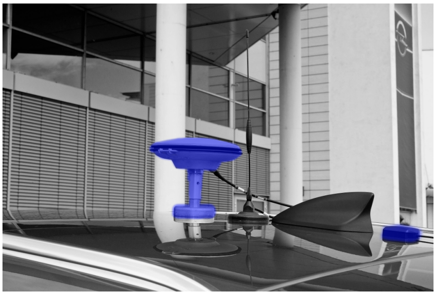
The sensor specifications are included in Appx. A.2. The SAS measures both steering angle and angle gradient of the steering wheel. The WSSs are mounted at each wheel and are equipped with magnetic encoders. 48 pulses are equivalent to a full wheel rotation. The IMU has a 2D accelerometer in x- and y- direction and a

[1] Opel Test Center (OTC), Dudenhofen, Germany
[2] ROS Kinetic Kame, Robot Operating System, `http://www.ros.org`
[3] Insignia MY16, Opel Automobile GmbH

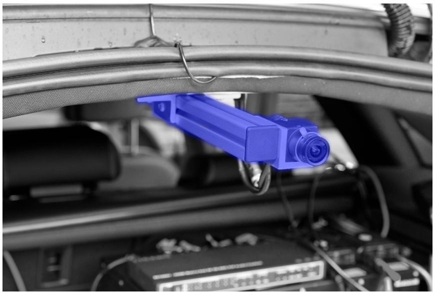
(a) Test vehicle



(b) Front camera setup



(c) GNSS antennas



(d) Rear camera setup

Figure 5.1: **Additional Sensors**. The sensor suite was extended by a front and rear camera and a low-cost GNSS receiver as well as a high-precision aided INS as ground-truth.

gyroscope in z-direction measures the yaw rate.

The setup was extended by a front and rear camera as well as a low-cost GNSS receiver.

- Front and rear 2.8 MP, 12-bit monochrome camera[4]
- low-cost GNSS evaluation kit[5]

The EVK-M8T utilizes a nonpublic EKF internally. Thus, estimated poses can only be incorporated via covariance intersection [25], as correlations between pose estimates are unknown. This is a conservative and nonoptimal approach. Thus, multi-

[4] FL3-GE-28S4M-C, Point Grey Research (now: FLIR Integrated Imaging Solutions Inc.)
[5] EVK-M8T, u-Blox Holding AG

GNSS raw measurement data, broadcast navigation data subframes, and SBAS status data[6] are processed in the SPP algorithm as presented in Sec. 4.2.1 to receive uncorrelated pose and velocity estimates.

5.1.2 Vehicle Sensor Calibration

Sensor calibration algorithms include estimation of time delays, sensor alignment or rigid transformation, and sensor error model parameters, such as offset and scaling parameters. This section briefly outlines the utilized sensor calibration algorithms. A more detailed description was presented in [Int2].

Time Delay Estimation between Sensors

The ability to obtain precise time stamps of measurements is important for fusing time-dependent sensor data. Dynamic sensor data, available on the CAN bus, have negligible or no preprocessing time. For the given setup time delays are below 2 ms per specification sheet. However, image processing and GNSS pseudo- and deltarange calculus impose increased computational burden. Thus, it is necessary to compute the time delay of the GNSS, map-relative and lane-relative pose estimates with respect to the dynamic sensor data.
The time delay of pose estimators towards dynamic sensor data is computed by cross-correlating their time-series data, as presented by Carter et al. [20]. Peaks in the resulting correlation graph indicate time delays. In this approach, delta orientations of the GNSS, lane-relative, and map-relative orientation estimates and the visual odometry's angular rotation are correlated with the mean IMU yaw rate in respecting time intervals. Sensor clock drifts and jitter, comprised by more elaborate temporal synchronization methods [93], have negligible influence, as all measurements are time-stamped by a central unit and it is assumed that each sensor provides measurements with an approximately constant time delay.

Camera Calibration

The front and rear camera's intrinsic and extrinsic parameters are estimated using an approach presented by Strauss et al. [78]. Multiple calibration patterns are detected by the camera setup and the association problem is solved over time using a sparse nonlinear LSQ solver. The main challenge, that this approach solves, is that cameras with nonoverlapping field of views can be calibrated.

[6] UBX-RXM-RAWX, UBX-RXM-SFRBX, UBX-NAV-SBAS

IMU Offset and Scale Estimation

The in-series vehicle dynamic sensors are mounted in well-known positions with negligible time delay. However, these sensors suffer from offset and scaling errors. Offset and scaling errors can be directly estimated using recursive linear regression. Delta velocities and delta orientations of the GNSS receiver are used to estimate the offset and scale of both the accelerometer and gyroscope of the IMU as presented in [Int2].

SAS Offset and Scale Estimation

The steering angle ratio and offset between the steering wheel and the steer angle of the front wheels is calibrated using recursive linear regression. The recursive algorithm estimates the ratio and offset by minimizing the difference between GNSS delta orientations and the yaw rate estimation of the kinematic single-track model presented in Eq. 4.29, with a small angle approximation $\tan(\delta_f) = \delta_f$ for the steering angle.

Wheel Diameter Estimation

The path estimation computed from GNSS position signals are used to estimate wheel diameters. However, distances between normally distributed position signals are larger than the true traveled path due to sensor noise. This effect is compensated by approximating the integration of Maxwell Boltzmann distributions as presented in [Int2].

5.1.3 Digital Map and Testing Grounds

Two test tracks were selected both for this thesis and in the project Ko-HAF. The first test track is a nonpublic highway test track used to investigate the algorithm performance under critical maneuvers. The second test track is a public highway used to test the presented algorithms under real-world perturbations. For both test tracks a high-precision digital map[7] was recorded priorly. Fig. 5.2 shows the mapped test tracks. The maps are stored in the OpenDRIVE file format [34].

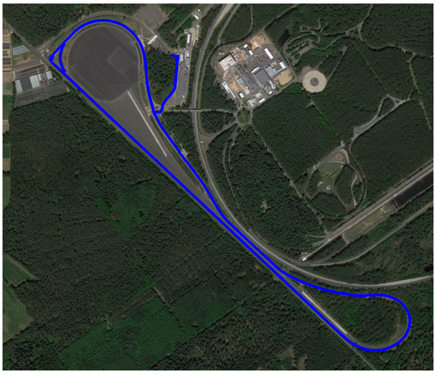
Opel Testcenter

The OTC test track is located on the Opel Test Center[8] and is used for preliminary testing and validation. It covers a 2 km straight highway test track and a return

[7] The digital map has been recorded in the course of the project Ko-HAF.

[8] Opel Test Center, Dudenhofen, Germany

track. The track can be divided in three parts. The entry party covers three lanes and a slip road. The middle part covers three normal lanes and the exit part an additional exit lane. A more detailed view of the map can be found in Appx. A.4. A preceding banked turn permits entering the highway track at high velocities. Additionally this test track is equipped with a DGPS system, presented in detail in Sec. 5.1.4, used to validate the pose accuracy quantitatively together with a validation system in the test vehicle.


(a) OTC test track [39]

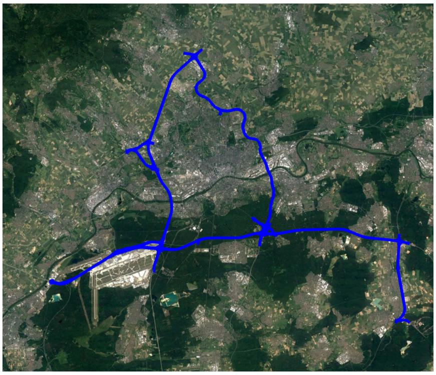

(b) Public highway track [40]

Figure 5.2: **Bird View of the Digital Map**. The OTC test track (a) covers approximately 2 km of highway roads. The public highway track (b) covers highway road A3, A5, A661 and federal highway B45 with approximately 140 km.

Public Highways

The public highway track covers 140 km of highway roads, including roads between Frankfurter, Bad Homburger, and Offenbacher Cross (A3, A5, A661, B45) and is displayed in Fig. 5.2b. These roads were selected as they cover some of the most complex highway roads with multi-lane environments, various interchange types, and tunnels. This test set is used to test the presented algorithm's robustness towards traffic, tunnels, washed-out lane markers, interchanges and GNSS multipath scattering. However, for this track a close DGPS reference station is missing. Receiving DGPS correction data via mobile network from reference stations further away, significantly reduces their accuracy[9]. Thus the pose estimates on this track will be evaluated qualitatively.

[9] approximately 1 cm in accuracy loss per km distance from the reference station

5.1.4 Ground Truth

To assess the accuracy of pose estimates, a ground truth that provides high-precision pose estimates is needed. Fig. 5.3 displays the ground truth system located at the OTC test track.
A DGPS reference station, receives GNSS signals via a Geodetic antenna[10]. A multi-channel, multi-frequency OEM GNSS receiver[11] computes the GNSS correction data from these signals and distributes the resulting Radio Technical Commission for Maritime Services (RTCM)-3.0 correction messages[12] via Networked Transport of RTCM via Internet Protocol (NTRIP) protocol over the Ethernet interface. The NTRIP casting is performed by a high-performance NTRIP broadcaster[13]. Two transmitting stations, located at the middle and southern part of the OTC test track, receive these messages and transmit them on a 433.50 MHz carrier wave using a high-frequency narrow band radio modem[14].
The test vehicle receives these messages using a second radio modem and forwards them to a high-precision aided INS[15], which directly process the RTCM-3.0 messages. An INS consist of a computing unit and an IMU and estimates orientation, velocity, and position, given a prior, by solving navigation equations, such as the Strapdown algorithm [85]. These estimates drift over time due to errors in the IMU's gyroscope and accelerometer. Aided INSs limit these drifts by adding additional sensors that observe these states directly and fuse their estimates. The ADMA consists of an INS with a high-precision 3D accelerometer and gyroscope and a DGPS receiver. Internally, an EKF is used to filter the data. Its geodetic antenna is displayed in Fig. 5.1c.

5.1.5 Software Framework and Computing Units

The algorithms presented in Ch. 4 were implemented in the general-purpose programming language C++ under Ubuntu. C++ was selected as it is one of the most developed programming languages. It provides high-performance object code, highly developed libraries and source code, several programming paradigms, and direct access to hardware resources while simultaneously ensuring high portability. Some additional tools for post processing, visualization, and configuration are implemented in Python and Java.

[10] Zephyr II, Trimble Inc.
[11] BX982, Trimble Inc.
[12] RTCM-3.0 Messages: 1004, 1005(9), 1008(9), 1012
[13] Ntrip Caster, Alberding GmbH
[14] Satelline-EASy 869, SATEL
[15] ADMA-G Entry Level, GeneSys Electronik GmbH

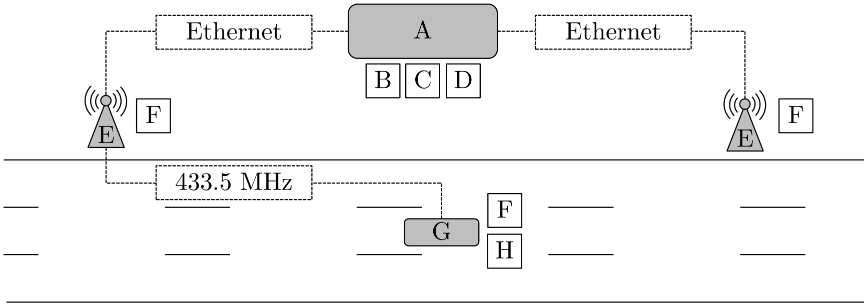


Figure 5.3: **Ground Truth System Overview**. DGPS reference station (A). GNSS antenna (B). Trimble GNSS receiver (C). NTRIP caster (D). Transmitting stations (E). Radio modem (F). Test vehicle (G). ADMA (H).

Software Framework

ROS is used as a software framework. An intensive study on ROS as a software framework for automated driving is presented by Hellmund et al. [46]. It was found that ROS is a modular and extensible framework with low overhead, supports fault-tolerant development and provides a rich ecosystem of supporting tools. Furthermore, latency and jitter of data signals in ROS were investigated. ROS was found to be a strong framework for developing and prototyping purposes, although it has some flaws regarding real-time performance and guaranteed message transmission. However, these flaws are addressed in the new ROS version 2.0.

Optimization Framework

The pose graph is modeled and solved using CERES, an open source C++ library which has been extensively tested by Google. It is actively maintained by its community, well documented, portable, and has been extremely optimized towards computational time. CERES was found to yield a higher solution quality compared to other solvers, such as HBN or MinPack. It further provides a variety of optimization solvers, linear solvers, loss functions and supports AD.

Computation Units

It is infeasible to guarantee hard real-time properties of the presented approach, especially since Ubuntu is used as an operating system. However, for live demonstrations the computational time is of utmost importance. Thus, the CPU usage

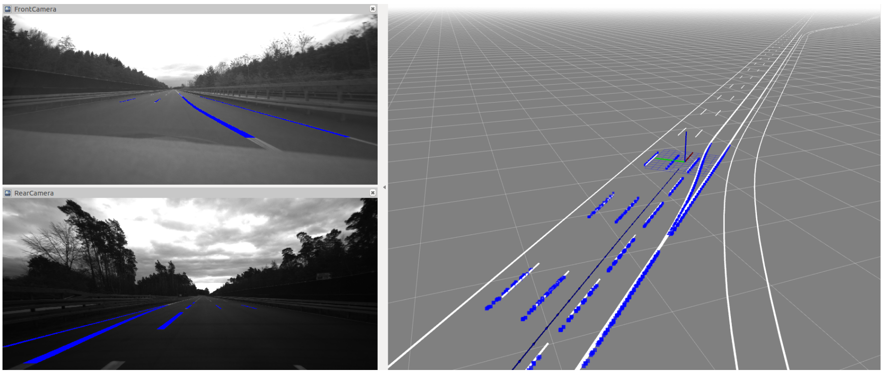


Figure 5.4: **Visualization in Rviz**. Detected lane markers (blue) are aligned with the digital map (white) by optimizing the ray distance towards map samples. Prior optimized poses (blue path) and odometry estimates are used to estimate the most recent pose (coordinate frame).

and the computational time of solving the optimization problem will be constantly monitored and evaluated in Sec. 5.3.4. The computational time depends on the computing unit. For runtime evaluations the algorithms are run on a mid range Dell M4800. The test vehicle is equipped with a superior computation unit. The hardware details are listed in Appx. A.3.

5.2 Test Sets

To evaluate the presented approach 15 datasets were recorded on the OTC test track. The dataset bulk was chosen to cover the vehicle's full dynamic range from lane keeping at moderate accelerations to full brake appliance and evasion. The sets were recorded over a time span of approximately one year and cover different weather conditions, such as cloudy, sunny, and rainy weather.
The OTC test track has been chosen to evaluate the approach as some scenarios, such as full brake application and evasion, imposes sincere hazards on public roads. Additionally, the OTC test track has a ground truth to evaluate the position and orientation accuracy quantitatively. Moreover, similar testing conditions can be assured, to depict the filter susceptibility to weather conditions, velocities, and differing dynamic maneuvers. Last but not least, the repeatability rate to continuously improve the algorithms is much higher on a test track than on public roads with alternating traffic load. Additionally, two sets are added to qualitatively investigate

the filter performance in multi-lane environments and by driving multiple loops in a cloverleaf interchange.

Dataset Representation

Each dataset is described by the lateral vehicle position, absolute velocity, and lateral and longitudinal acceleration. The time-series plots are condensed to violin plots as depicted in Fig. 5.5. Violin plots can be interpreted as vertical, symmetric histogram plots of time-series data. The violin plot's vertical axis denotes the absolute value while the horizontal axis describes the value's occurrence. Thus, violin plots can be interpreted as value distributions. Fig. 5.12 shows the time-series and violin plots of dataset C where the vehicle performs lane changes in a sinusoidal manner at a constant velocity. During the last section the vehicle performs a lane change on the exit lane and reduces its velocity.

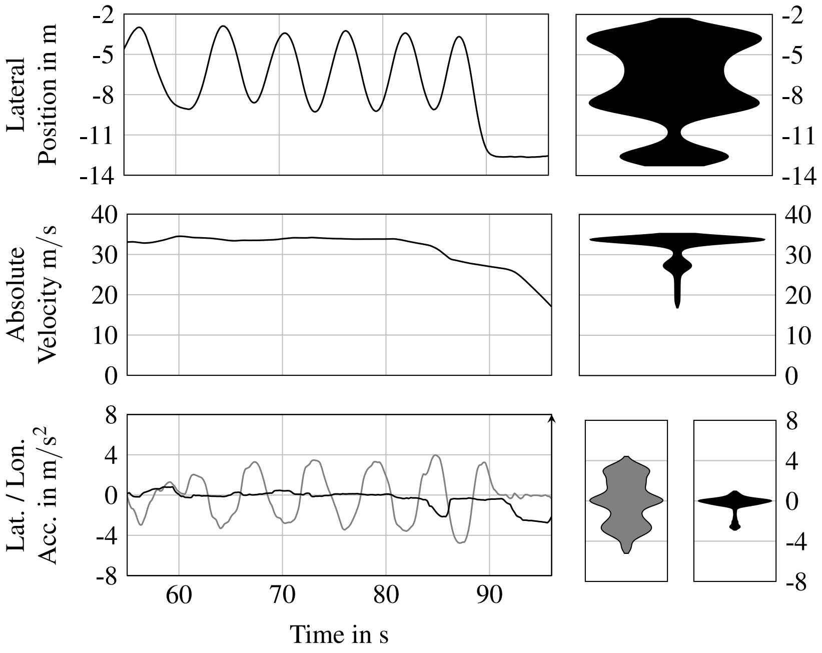
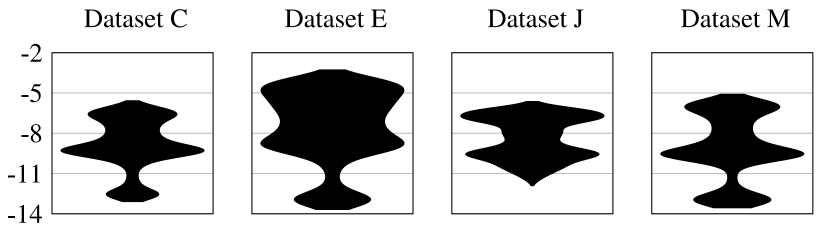
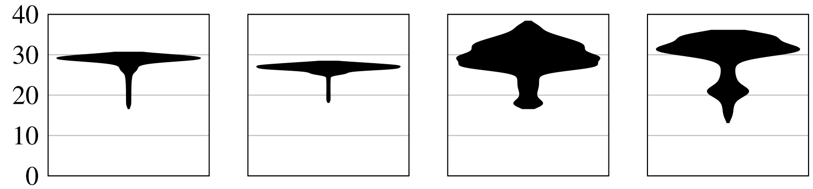


Figure 5.5: **Dataset Representation Plots**. The vehicle position, velocity, and acceleration timeseries plots (left) are converted to violin plots (right) to qualitatively describe the test set. Violin plots are vertical histograms and visualize the rate of occurance.

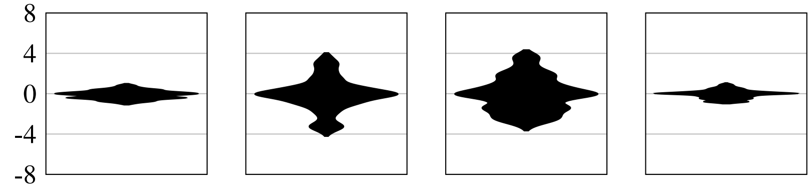
All datasets are separated in four groups described in the following. Violin plots of dataset samples of each group are depicted in Fig. 5.6. All violin plots of the datasets can be found in Appx. A.5. 5.7



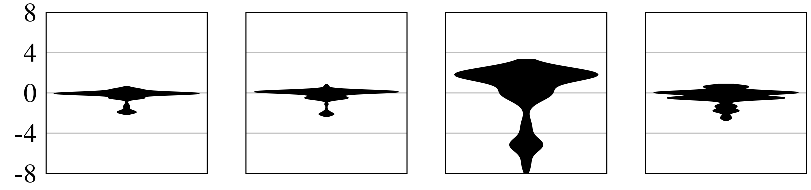
(a) Lateral position in m



(b) Absolute velocity in m/s



(c) Lateral acceleration in m/s^2



(d) Longitudinal acceleration in m/s^2

Figure 5.6: **Dataset Examples**. Depicted are position, velocity, and acceleration violin plots of a sample set of each dataset group. All violin plots are included in the appendix Appx.A.5.

Dataset Group A

The initial group of datasets covers four full drives on the OTC test track. In dataset A and dataset B the lane is kept at a mean velocity of $29.26\,\mathrm{m/s}$ and $20.81\,\mathrm{m/s}$ respectively. In dataset C and dataset D 8 to 12 lane changes are performed with moderate lateral and longitudinal accelerations at a vehicle velocity of $30.09\,\mathrm{m/s}$ and $28.45\,\mathrm{m/s}$. The purpose of these sets is to investigate the performance in typical highway driving scenarios.

Dataset Group B

The second group of datasets covers more dynamic driving maneuvers, which are more demanding for pose estimators as nonzero pitch and roll angles occur. In dataset E and dataset F 23 to 26 lane changes are performed by following a sinusoidal path. The lateral accelerations are between $-4.76\,\mathrm{m/s^2}$ and $3.97\,\mathrm{m/s^2}$. The lateral position violin plots of this group show that for all sets the vehicle is constantly changing lanes as the lateral positions vary significantly. While dataset E and dataset F are used to investigate the influence of nonzero roll angles, dataset G and dataset H are used to investigate the influence of nonzero pitch angles. For the latter two sets only 4 to 10 lane changes are conducted at perceptible longitudinal accelerations.

Dataset Group C

The third group of datasets covers highly dynamic scenarios where both high lateral and longitudinal accelerations are applied. These sets represent evasive maneuvers in critical situations. Lateral accelerations from $-4.53\,\mathrm{m/s^2}$ to $4.01\,\mathrm{m/s^2}$ and longitudinal accelerations from $-9.33\,\mathrm{m/s^2}$ to $3.99\,\mathrm{m/s^2}$ were applied. These sets are very demanding for pose estimators as nonzero pitch and roll angles are constantly present. Furthermore, the assumption of linear models is violated and fluctuating velocities intensify interpolation or synchronization errors.

Dataset Group D

The last group of datasets covers additional scenarios that are very demanding for visual pose estimators. Dataset M is recorded during rainy weather with significant surface reflections on the road, which is caused by a perfectly flat test track with no lateral curvature, as it is mandatory for public highways. In dataset N an emergency stop maneuver on the marginal strip is performed. These maneuvers are performed when hand-over requests to the driver cannot be successfully completed. Dataset O covers the 3 km long four- and five-lane environment on the public, German

(a) Dataset L: Subsequent image overlay visualizes the pitch angle

(b) Dataset M: Rain

(c) Dataset C: Dazzling sunlight

(d) Dataset O: Five-lane environment

(e) Dataset O: Small bridges

(f) Dataset P: Highway interchange

(g) Dataset P: Highway interchange

Figure 5.7: **Dataset Sample Images**.

highway A3. Dataset P covers a 4 km drive through six quaters of the cloverleaf interchange *Aschaffenburger Cross*. The last two sets were recorded in absence of DGPS correction data.

5.3 Experimental Results and Evaluation

This section focuses on the pose estimation results of the presented algorithms. An EKF and PF approach were implemented for the given setup and presented in [Int1]. Their performance was found to be inferior compared to the presented approach [Int3] and will not be consulted for an extensive study.

The algorithms were applied on the datasets described previously. The pose error is computed by calculating the difference to the ground truth system presented in Sec. 5.1.4. It was assumed that the ADMA has negligible noise. The covariance estimation algorithms are investigated in a simulation run and the computed covariance matrices eigenvalues are compared to the standard deviation of the lateral and longitudinal position and orientation errors. Last but not least, the approach will be evaluated qualitatively in demanding scenarios on the public highway track where both camera and GNSS estimates are significantly perturbed and the integrity monitoring system will be investigated in these scenarios.

5.3.1 Pose Estimation

Due to high velocities in longitudinal direction, small velocities in lateral direction, and since the lane marker matching problem is well-conditioned in lateral but ill-conditioned in longitudinal direction, the lateral and longitudinal uncertainty differ significantly. Thus, the position error is split in a longitudinal and lateral part. The longitudinal position error is computed by projecting the pose estimate and ground truth on the reference line of the digital map. The lateral position is computed by the distance to the reference line, respectively.
It was found that the absolute UTM position error is similar to the longitudinal position error, as the latter is significantly higher than the respective lateral error. Thus, in the following only the lateral position and total UTM will be displayed. Although DGPS is available at the OTC test track, a small position offset[16] between the ADMA and the digital map is frequently present, thus the ADMA trajectory and the optimized pose trajectory are aligned in a 20 s sliding window. The orientation error is computed by comparing the yaw angle of the pose estimate and ADMA quaternion directly.

[16] within 20 cm

Pose Estimation Results Representation

Similar to the datasets, the pose estimation results are described by violin plots. The time-series and violin plots of the lateral position, UTM position, and orientation error for the errors on dataset J are qualitatively displayed in Fig. 5.8.

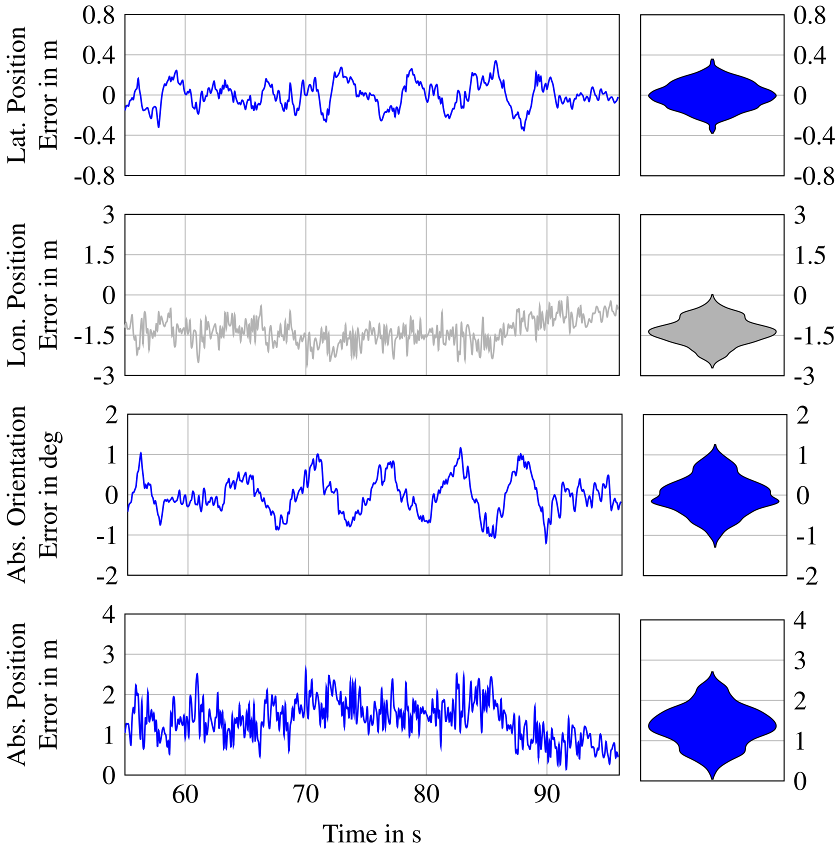
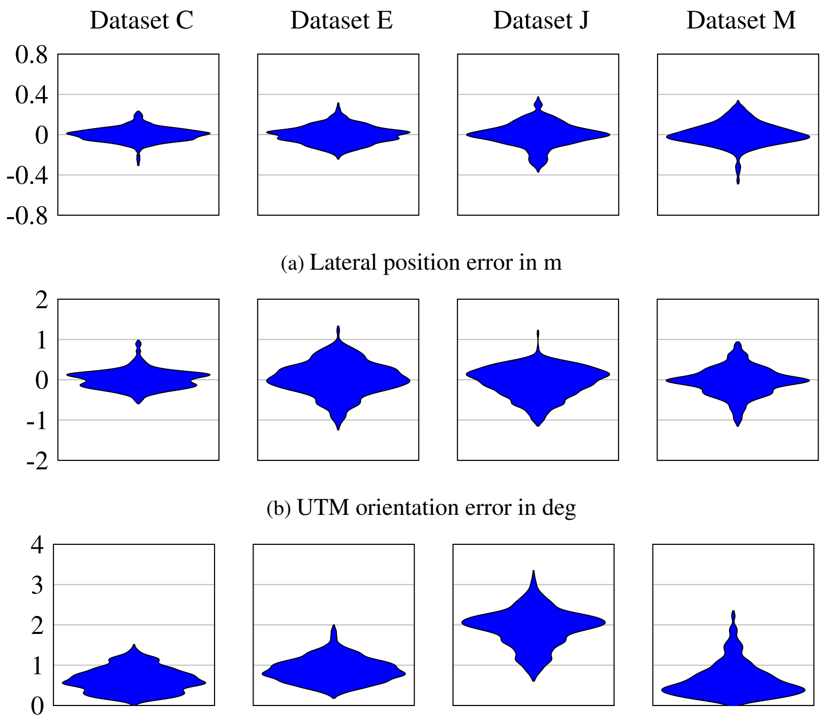


Figure 5.8: **Dataset Pose Estimation Plots for Dataset J**. The lateral position, UTM orientation, and UTM position error timeseries plots (left) are converted into violin plots (right) to visualize errors rate of occurance.

The pose estimation results are separated according to the dataset groups. Violin plots of pose estimation results of each dataset group are depicted in Fig. 5.9. All violin plots of the datasets can be found in Appx. A.6.



(a) Lateral position error in m

(b) UTM orientation error in deg

(c) UTM position error in m

Figure 5.9: **Dataset Example Results**. Depicted are the pose estimation error violin plots on a sample set of each dataset group. All violin error plots are included in the appendix Appx.A.6. .

Dataset Group A

The position estimation STD of the dataset group A is approximately 10 cm and the yaw orientation STD is below 0.20 deg with a maximal orientation error below 0.80 deg for all sets. Although the maximal position error is below 35 cm for three sets, the maximal position error in dataset A is 58.37 cm. All lateral position and orientation values are listed in tables in Appx. A.7.

Dataset Group B

Although the pose estimation accuracy is slightly increased in the dataset group B compared to the prior set, there is no considerable outlier in the both the lateral position and orientation estimate. The lateral position STD is below 13 cm with a maximum error below 39 cm and the orientation STD is approximately 0.40 deg or below with a maximum error below 1.10 deg.

Dataset Group C

Although the position and orientation STD in the dataset group C are below 13 cm and 0.35 deg, similar to the dataset group B, the maximum errors are a significantly increased, with a maximum lateral position error of approximately 60 cm in dataset I and a maximal orientation error of approximately 1.90 deg in dataset L. However, as the dataset covers highly dynamic scenarios with full brake applications, errors also arise due to imprecise time stamping and interpolating errors that are prone to rapid changes in the observed quantities.

Dataset Group D

Visual pose estimators are susceptible to viewing conditions. However, heavy rain in dataset M does not significantly downgrade the pose estimation accuracy. The set has a maximum lateral error of 31.59 cm and STD of 11.18 cm and a maximum orientation error of 0.87 deg. The accuracy of the pose estimates during the emergency stop, are inferior compared to the prior sets. A lateral error STD of 20.64 cm with a maximal value of 50.75 cm was computed. Dataset O and dataset P were recorded in absence of DGPS correction data. It was found that the ADMA suffers from significant offset errors when projected on the map during test drives. However, to perform a qualitative analysis, the ADMA pose estimates were aligned with the optimized poses before computing the error. For dataset O a lateral position error STD of 12.69 cm with a maximum value of 45.64 cm was computed. The orientation estimate had a maximum error of 0.38 deg. For dataset P an orientation STD of 0.84 deg with a maximum error of 2.98 deg was computed. As the cloverleaf interchange is represented by a magnitude of reference lines and the ADMA is susceptible to multipath scattering, a distinct projection on the reference line is impeded and the UTM position error is computed. A UTM position STD of 68.85 cm 541.85 cm was computed.

The presented approach showed a good overall performance on the datasets. Even though a 2D map is used and high lateral and longitudinal accelerations are applied together with differing weather conditions. However, although the STD in the lateral position in dataset A, dataset I, and dataset O is within reasonable limits, the maximum error is above the tolerable range for vehicle maneuvering. As indicated by the lateral error violin plots of the respective sets in Appx. A.6, the maximum occurred during a very short time period. If malfunctioning of the map-relative pose estimator is detected, presented in the following section, the localization module may temporarily switch to a lane-relative pose estimation module or any other localization module that is present.

5.3.2 Covariance Estimation

To evaluate the covariance estimation algorithm in simulation, samples from four MNDs, depicted in Fig. 5.10 as black ellipses, represent the sensor measurements and are added upon another MND representing the vehicle movement. By comparing differences of the resulting samples in each step, as presented in Fig. 4.22, the covariance matrices were estimated using Eq. 4.56. Additionally, another MND is added upon two sensor MNDs to simulate the correlation between two sensors. With increasing number of samples, the sensor covariance estimation converges without knowing the underlying vehicle movement MND. Fig. 4.22 shows the convergence of the covariance matrix estimates, visualized as gray and blue ellipes, with increasing number of samples.
The simulation results showed that the presented algorithms are able to robustly estimate covariance matrices of MNDs. To evaluate the covariance estimation on real-world data, the algorithm is executed on the pose estimates of the map matching algorithm during a sliding window of 2 s and compared to the pose error presented in Sec. 5.3.1 for all sets. The results are listed in Tab. 5.1.
It can be seen that the lateral position uncertainty of the map matches are about twice as high as the filtered result. However, the rotation uncertainty is approximately equal. This inconsistency stems from the fact that map matches on the wrong lane introduce a significant lateral error while the orientation error is negligible. The effect can be directly seen in the time-series of the uncertainty estimation in Fig. 5.12 for dataset A. Feeding back the outlier classification of pose estimates after the optimization process can make the uncertainty estimator more stable against these ambiguities.

5.3.3 Integrity Monitoring

Once sensor uncertainties are estimated and pose estimates are optimized, it is necessary to assess the solution. Therefore, the Mahalanobis distances of graph edges are evaluated to compute an estimate of the localization module's performance.

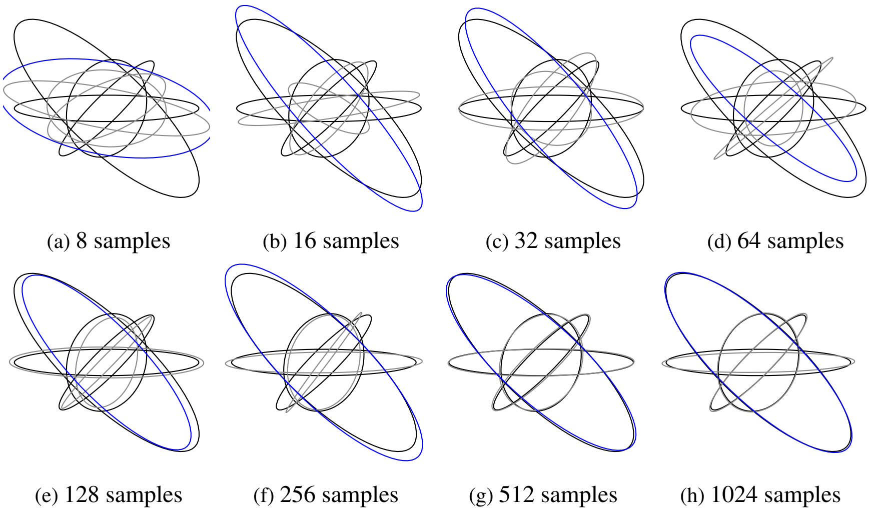


Figure 5.10: **Covariance Estimation**. The vehicle movement is simulated by drawing samples from a MND. Sensor estimates are simulated by adding additional noise, from MND (black) with correlations. The covariance matrices (gray and blue) are estimated given only the sensor estimates. With increasing samples the covariance matrix estimates become more accurate.

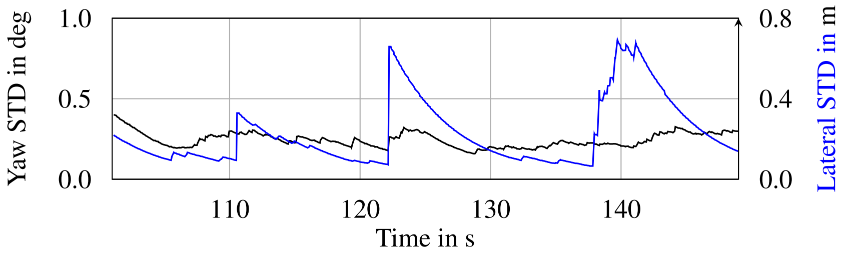


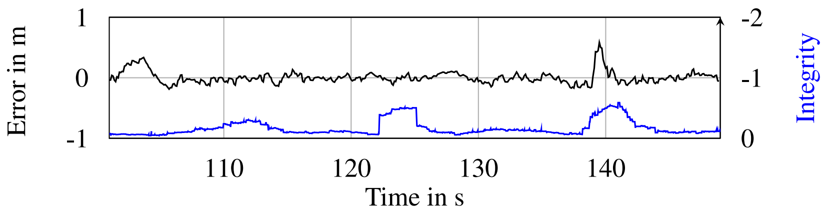
Figure 5.11: **Uncertainty Estimation Time-Series for Dataset A**. While the yaw angle uncertainty estimation is robust, the lateral uncertainty estimation of the map matching algorithm is susceptible to map matches on wrong lanes.

Fig. 5.12 shows the lateral or UTM error and the integrity value for three scenarios with varying complexity. While the integrity is high for the low-dynamic dataset A recorded on the OTC test track, the integrity noticeably decreased for dataset O, which was recorded on a five-lane environment of the A3. For the cloverleaf

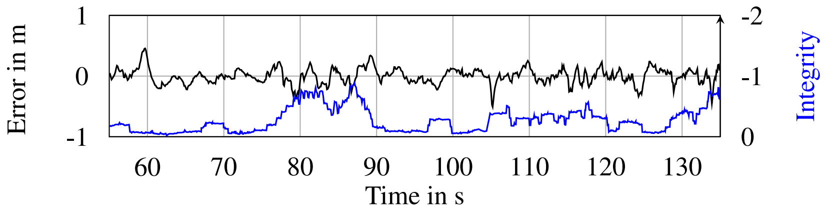
	E	T	E	T
Dataset A	22.72 cm	10.04 cm	0.24 deg	0.18 deg
Dataset B	16.19 cm	8.58 cm	0.23 deg	0.23 deg
Dataset C	16.96 cm	7.00 cm	0.23 deg	0.23 deg
Dataset D	16.53 cm	7.71 cm	0.20 deg	0.23 deg
Dataset E	13.16 cm	8.73 cm	0.21 deg	0.40 deg
Dataset F	15.27 cm	12.76 cm	0.34 deg	0.45 deg
Dataset G	24.66 cm	7.10 cm	0.24 deg	0.30 deg
Dataset H	18.02 cm	5.87 cm	0.19 deg	0.33 deg
Dataset I	26.74 cm	12.35 cm	0.34 deg	0.37 deg
Dataset J	11.96 cm	11.65 cm	0.21 deg	0.36 deg
Dataset K	19.64 cm	10.55 cm	0.27 deg	0.29 deg
Dataset L	24.95 cm	8.38 cm	0.20 deg	0.29 deg
Dataset M	44.08 cm	11.18 cm	0.52 deg	0.34 deg
Dataset N	59.71 cm	20.64 cm	0.42 deg	0.35 deg
Dataset O	44.12 cm	12.69 cm	0.24 deg	0.18 deg
Dataset P	69.45 cm	68.85 cm	0.76 deg	0.56 deg

Table 5.1: **Covariance Estimation Results**. The table depicts the lateral position (left) and orientation (right) uncertainty estimations (E) of the map matching algorithm and the computed STD of the optimized pose error towards the ADMA (E). For dataset P the UTM position error is displayed.

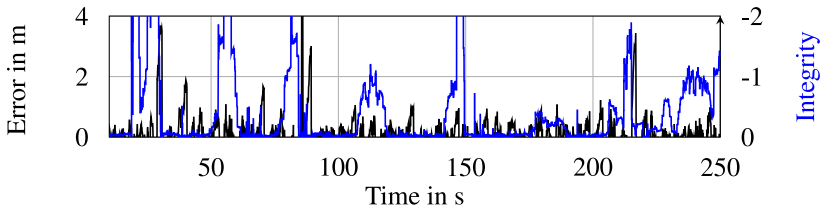
intersection in dataset P the localization module is temporarily disturbed and the integrity is significantly decreased. The undergrade crossings of the intersections at approximately 27 s, 83 s, 150 s, and 210 s can be clearly seen in the integrity time plot. Fig. 5.13 shows the integrity UTM-heatmap for these datasets. While the integrity value is high for dataset A and dataset O, it is significantly decreased for the undergrade crossings in dataset P. From Fig. 5.7f and 5.7g it can be seen that lane markers are hard to observe. Additionally, GNSS signals are highly perturbed due to multipath scattering on the undergrade's surface. Ultimately, these disruptive factors lead to solver divergence at the first undergrade passing. However, the integrity monitoring system is able to detect the inconsistency between map matches and odometry estimates, denoted by blue pose estimates.



(a) Dataset A on the OTC test track



(b) Dataset O on a five-lane environment of the A3.



(c) Dataset P on the cloverleaf highway intersection Aschaffenburger Cross

Figure 5.12: **Integrity Monitoring Time-Series**. Intensified complexity of the scenario can be directly seen in the integrity value. For dataset P the total UTM error is shown instead of the lateral error. Note that the vertical axis of the integrity has a negative sign to be consistent with the relationship of low residuals and high integrity.

5.3.4 Runtime Evaluation

Guaranteeing hard real-time capabilities and estimating a maximal upper computation time for the presented algorithm is out of the scope of this thesis due the

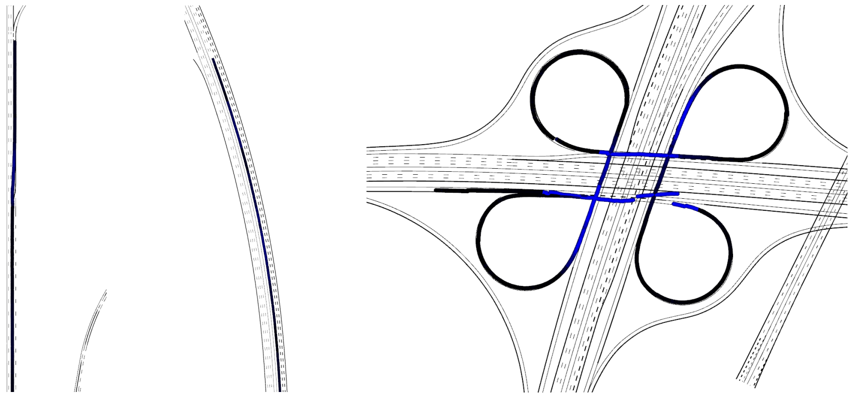

(a) Dataset A (b) Dataset O (c) Dataset P

Figure 5.13: **Integrity Monitoring UTM-Heatmap**. The integrity value is encoded in the pose estimate's color. Black denotes a high integrity value of 0.00 and blue denotes a low integrity value of -1.00 and below. It can be seen that the pose estimation has a low integrity in the cloverleaf intersection during undergrade crossings.

complexity of the utilized libraries running on Ubuntu[17]. However, as the approach was a key module of the automated driving functions, presented publicly during the two-day Ko-HAF final event, computational time is of utmost importance.
To quantify the computational burden of the algorithms, both the CPU usage of the respective ROS node and the optimization solver time were constantly monitored. Fig. 5.14 shows the solver time and CPU usage. The test vehicle was equipped with a superior computation unit. Details on the computing unit's hardware are included in Appx. A.3. The solver is triggered at 30 Hz and solves the optimization problem whenever new measurements are available. Pose estimates are provided at 50 Hz using priorly optimized pose estimates and odometry estimates.
The dataset has a negligible influence on the computational burden of the presented approach. Thus, an extensive evaluation and discussion of the solver time and CPU usage on all test sets is dispensed. Instead, a sample time-series plot of a reference set is used to quantify the computational time. Fig. 5.14 shows the initialization phase of the pose graph optimization. During the first few seconds the solver time an CPU usage is relatively small. As the amount of available pose estimates from the camera and GNSS receiver increases, the solver time increases likewise. The

[17] The used Ubuntu version cannot guarantee hard real-time performance

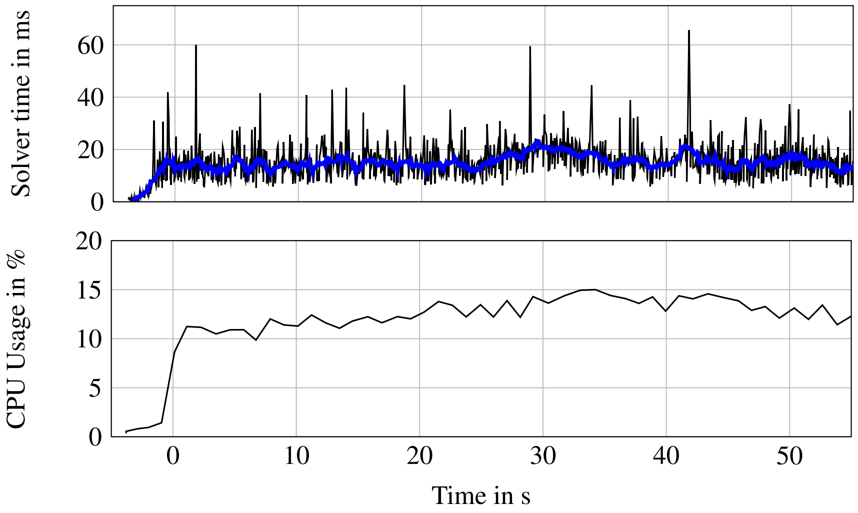


Figure 5.14: **Runtime Evaluation**. Top: The black line shows the raw solver time in ms and the blue line the mean solver time [window size = 21] of all three pose graph optimizations. Bottom: CPU Usage of the whole pose graph optimization ROS-node at 100 % playback speed is shown. 12.5 % is equal to one of eight logical CPU cores of an i7-4810MQ.

time plot is shifted, so that $t = 0$ denotes the time when enough pose estimates are present so that the solver drops old poses. At this point, solver time and CPU usage stagnate.
The solver time for all three pose graph optimizations has a mean of 14.48 ms and STD of 6.68 ms. The maximal computation time is 65.69 ms. The solver is run on a separate thread. Thus, frequent odometry pose updates can still be provided at 50 Hz, but their accuracy is inferior with increasing time since the last successful solver iteration. The CPU usage, of the respective ROS-node process, has a mean of 11.54 % and STD of 3.58 %. The maximal value of the CPU usage is 15.00 %. This includes the measurement handling, map matching, covariance estimation, integrity monitoring, and other algorithms that are necessary to solve the pose graph problem. 100 % represents eight logical or four physical CPU cores of the Dell M4800.

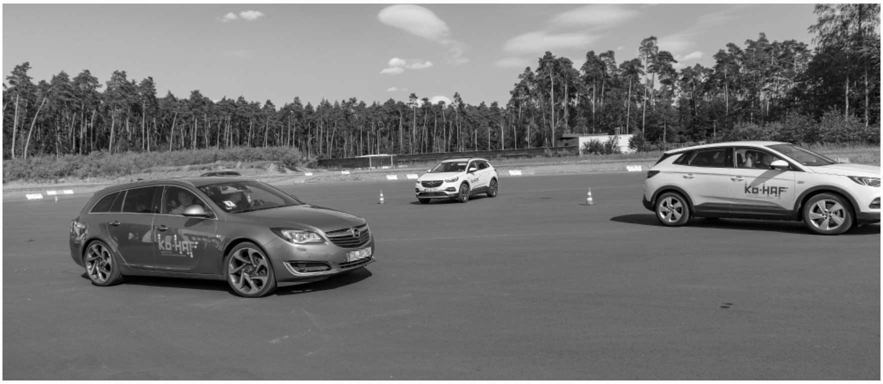


Figure 5.15: **Ko-HAF Final Demonstration**. The image shows the test vehicle (gray) and two additional vehicles (white), to simulate traffic, after picking up participants at the Ko-HAF final event.

6 Summary, Conclusion and Outlook

A novel approach for fail-safe vehicle pose estimation in sparse lane-level highway maps through pose graph optimization was presented. The proposed set of algorithms rely on a monochrome front and rear camera, a low-cost GNSS receiver and in-series vehicle dynamic sensors. A graph with pose estimates as vertices and odometry estimates as graph edges is used to represent the optimization problem. To ensure constant computation time only the latest pose and odometry estimates are included in the graph.
The GNSS residual is calculated using the Mahalanobis distance to SPP pose estimates from pseudo- and deltaranges. The map matching residual is computed by aligning light rays from the lane marker detection module with the digital map and returning one or several, if the matching is ambiguous, pose estimates. Aligning light rays with lane-level maps instead of projecting these rays on a static ground plane and aligning the resulting point clouds, is especially advantageous under high pitch and roll angles of the vehicle body towards the chassis. Several odometry estimates are checked for consistency and fused in advance to derive a robust odometry estimate that is used to link pose estimates in the pose graph.
To increase the robustness of the optimization solver towards measurement outliers and map matching ambiguities, a set of loss functions are applied on the residuals. Multiple pose graphs, each with a subset of sensors, are solved to provide map matching initialization as well as map-relative and lane-relative pose estimates.
To weight residuals in the optimization problem, the pose and odometry estimators' covariance matrices are computed by comparing pose deltas and odometry estimates, without the necessity of a high-precision reference station or extensive calibration procedures. The optimization residuals and the estimators covariance matrices are processed in an integrity module to compute the consistency of the pose estimation result.

The presented algorithms were investigated on a test vehicle, equipped with an additional front and rear monochrome camera, a computing unit and a low-cost GNSS receiver. Furthermore, the vehicle was equipped with a high-precision aided INS for validation purposes. The algorithms were investigated quantitatively on the OTC test track, where a DGPS reference system is available, and qualitatively on public highways around Frankfurt, Germany.

On the test track a lateral accuracy STD of below 13 cm was achieved, even for highly dynamic scenarios, such as full brake application and evasive vehicle maneuvering at velocities up to 140 km/h. Although the pose estimation algorithms provide accurate estimates in real-world scenarios with multiple lanes, it was found that the estimators accuracy is highly degraded in demanding scenarios, such as undergrade crossings of cloverleaf intersections, were both lane marker detections and GNSS pose estimates are highly corrupted. However, the integrity monitoring system is able to robustly detect the inconsistency in perturbed situations and thus a fail-safe pose estimation can be guaranteed.
The optimization problem can be solved with a mean computation time of 6.68 ms on a Dell M4800, beside other tasks such as image processing. The presented approach was demonstrated during the public Ko-HAF final event.

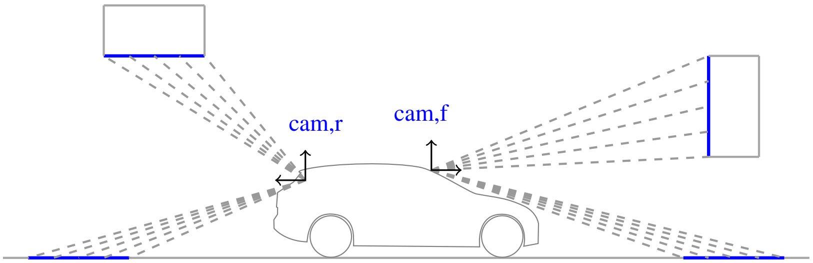


Figure 6.1: **Extending the Approach to 3D Structures**. Edges and corners of 3D infrastructure, such as tunnels, buildings and guardrails, can be efficiently represented in digital maps by parametric curves. Using these features, the presented approach can be further robustified and improved.

The presented pose graph optimization technique was found superior compared to an EKF and PF implementation [Int3], as parameter constraints, outlier rejections through loss functions and ambiguities selection can be embedded in the optimization problem directly. Although it is possible to intercept outliers in EKFs or PFs by applying a χ^2-Test on the innovation step size and resolve ambiguities through iteratively solving the EKF or PF equations throughout a specified time interval, while gating various ambiguous solutions, the optimization approach is much more intuitive and less error-prone.
It is advised to extend the sensor setup as long tunnels and undergrade crossings in cloverleaf intersections can temporarily disturb both camera and GNSS pose estimates which can lead to solver divergence. Pose estimates based on Lidar, ultrasound or radar sensors are suggested, as these sensors are readily available in

in-series vehicles. The presented approach can be flexibly extended to additional pose and odometry estimators. The integrity monitoring system can be used to weight the pose estimates from different sensors to guarantee robust and fail-safe pose estimates. Furthermore, the utilized dense HD maps with lane markers can be enhanced to infrastructure edges, represented by splines that aid both the visual and lidar localization techniques as depicted in Fig. 6.1.
To delimit the computational complexity of the optimization problem, the odometry estimates were priorly fused. If computational time is negligible or if few odometry estimates are available and their integrity cannot be assured, raw odometry estimates can be incorporated in the pose graph optimization directly to monitor their integrity as well.
Multipath scattering was rejected using an RAIM algorithm. However, it is beneficial to directly include pseudo- and deltarange measurements in the pose graph optimization directly to dissolve ambiguities, or to support the pose estimation in concealed areas with less than four satellites as tightly-coupled aided INSs do.

A Appendix

A.1 Sensor Positions

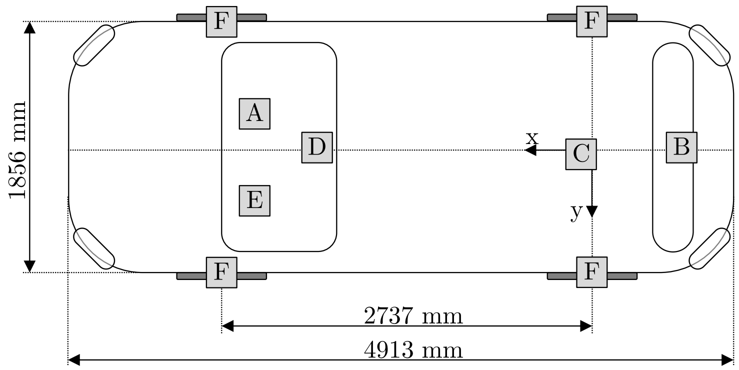


Figure A.1: **Schematic Overview of Sensor Positions.** Front camera (A), rear camera (B), ADMA (C), IMU (D), SAS (E), and WSS (F). The absolute position values are listed in Tab.A.1

	A	B	C	D
x	2492 mm	−637 mm	110 mm	2384 mm
y	−266 mm	−35 mm	−40 mm	−11 mm

Table A.1: **Sensor Positions**. Front camera (A), rear camera (B), ADMA (C), and IMU (D) in vehicle coordinates (DIN ISO 8855)

A.2 In-Series Sensors

Steering Angle Sensor

- Data: Angle and angle velocity of the steering column
- Rate: 100 Hz
- Range: ± 740 deg
- Resolution: 0.10 deg
- Error (maximum): ± 2.00 deg offset, ± 4.00 deg scale

Wheel Speed Sensor

- Data: Cyclic pulse counter
- Rate: 100 Hz
- Range: −512 to 511 ticks
- Resolution: 48 ticks per wheel rotation

2D Accelerometer

- Data: Lateral and longitudinal acceleration
- Rate: 100 Hz
- Range: ± 14.28 m/s
- Resolution: 0.05 m/s
- Error (maximum): ± 1 m/s offset, ± 5.50 % scale

1D Gyroscope

- Data: Yaw rate
- Rate: 100 Hz
- Range: ± 100 deg /s
- Resolution: 0.10 deg /s
- Error (maximum): ± 2.50 deg offset, ± 4 % scale

A.3 Computation Units

Self-configured vehicle computer

- CPU: 2 x Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60 GHz
  Physical cores: 8 (2 logical cores per physical)
  Clock: 1199.96 MHz
  Cache size : 20 480 kB
- GPU: GeForce GTX 980
  Core clock: 1126 MHz
  Memory: 4096 MB
  Memory clock: 1752.50 MHz
- RAM: 8 x Samsung M393A1G40DB0-CPB 8GB
  Size: 8192 MB
  Speed: 2133 MHz
  Type: DDR4

Dell M4800 workstation, Dell Technologies Inc.

- CPU: Intel(R) Core(TM) i7-4810MQ CPU @ 2.80 GHz
  Physical cores: 4 (2 logical cores per physical)
  Clock: 1223.12 MHz
  Cache size : 6144 kB
- GPU: Nvidia Quadro K2100M
  Core clock: 654 MHz
  Memory: 2048 MB
  Memory clock: 1500 MHz
- RAM: 4 x Samsung M471B1G73QH0-YK0
  Size: 8192 MB
  Speed: 1600 MHz
  Type: DDR3

A.4 Highway Test Track

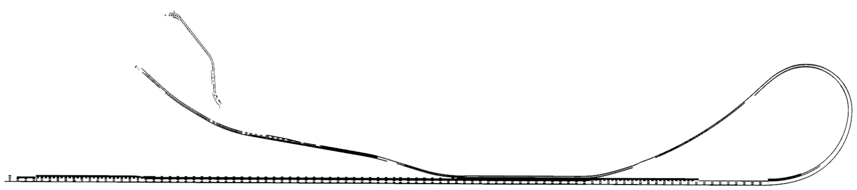
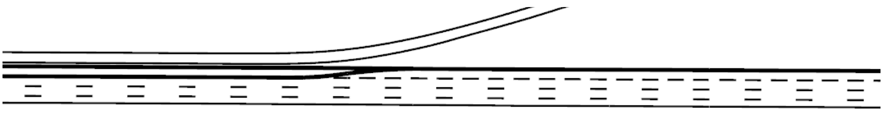
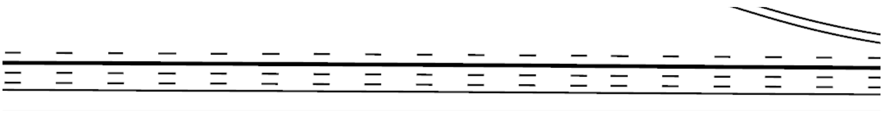


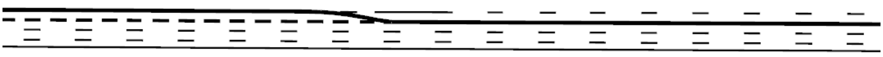
Figure A.2: **Opel Testcenter - Highway Test Track**. Displayed in Rviz.



(a) Access



(b) Part of the middle section

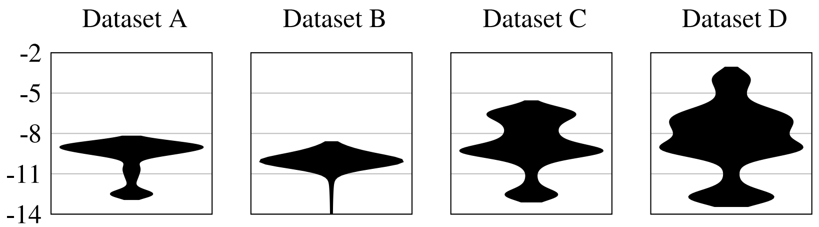


(c) Exit

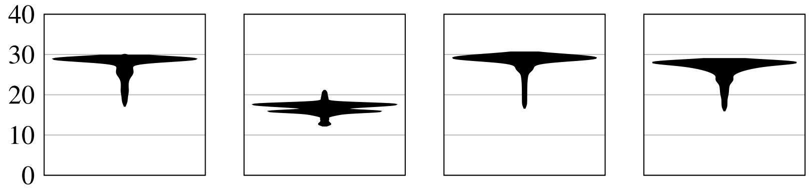
Figure A.3: **Opel Testcenter - Highway Test Track (Detailed)**. Displayed are three parts of the OTC test track on the Opel Test Center in Rviz.

A.5 Datasets

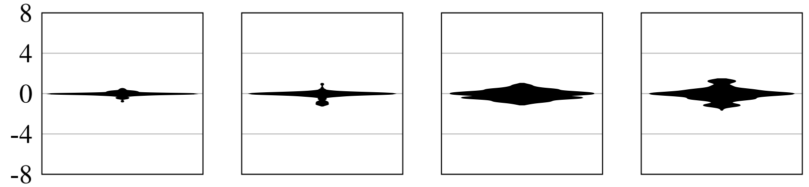
Dataset Group A



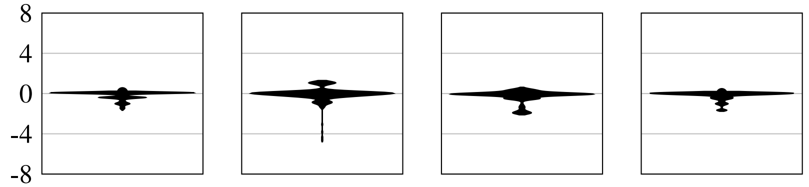
(a) Lateral position in m



(b) Absolute velocity in m/s



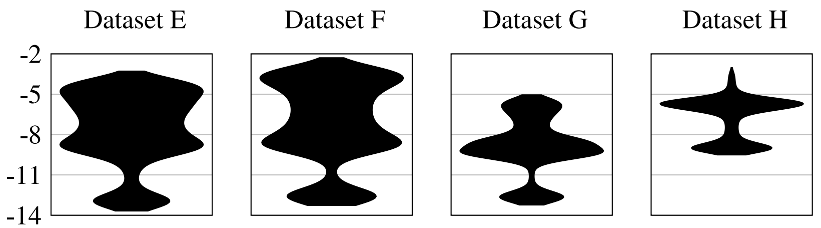
(c) Lateral acceleration in m/s^2



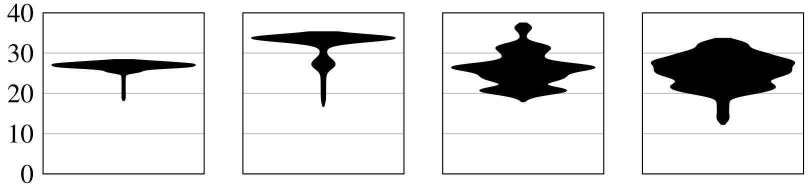
(d) Longitudinal acceleration in m/s^2

Figure A.4: **Dataset Group AViolin Plots**.

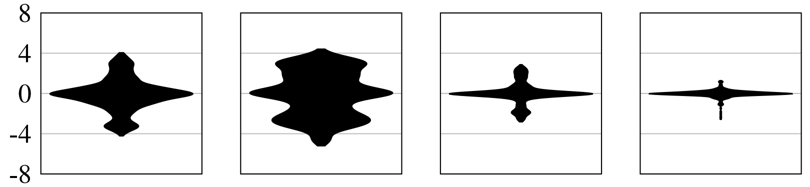
Dataset Group B



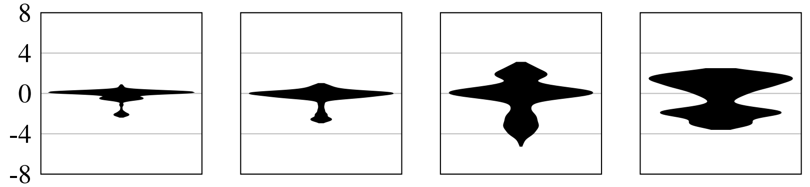
(a) Lateral position in m



(b) Absolute velocity in m/s



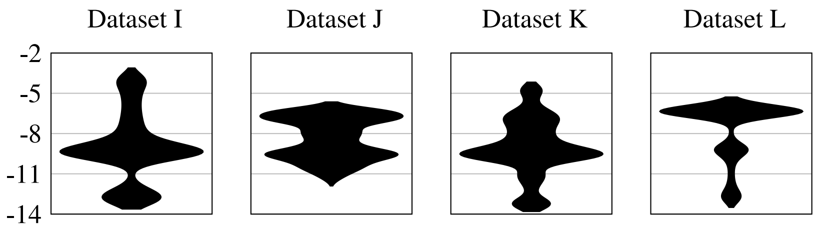
(c) Lateral acceleration in m/s^2



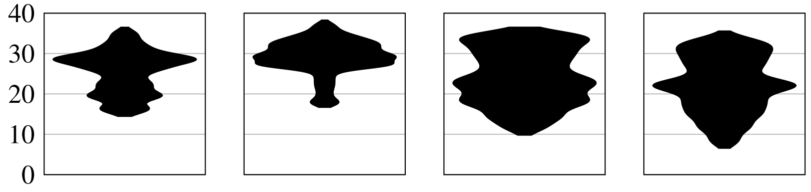
(d) Longitudinal acceleration in m/s^2

Figure A.5: **Dataset Group BViolin Plots**.

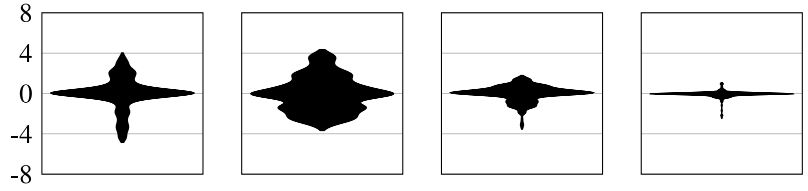
Dataset Group C



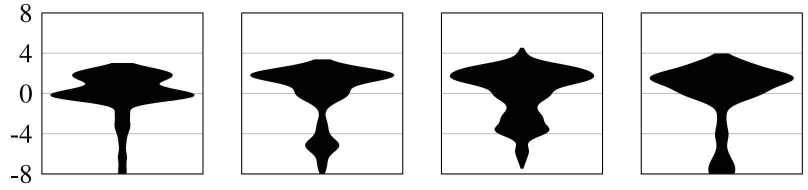
(a) Lateral position in m



(b) Absolute velocity in m/s



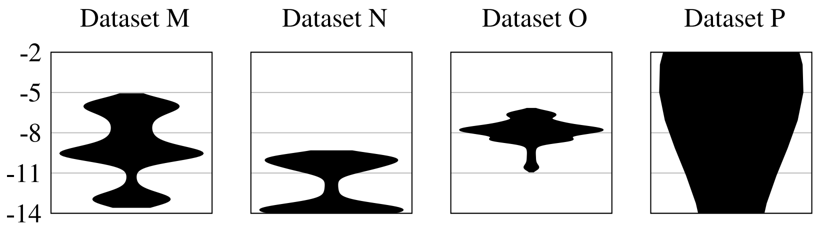
(c) Lateral acceleration in m/s^2



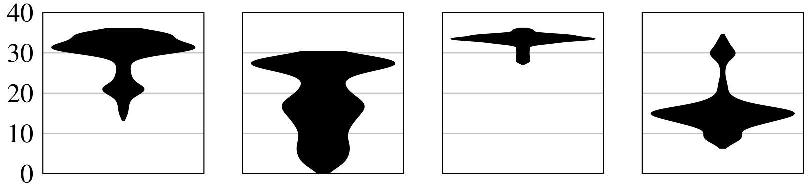
(d) Longitudinal acceleration in m/s^2

Figure A.6: **Dataset Group CViolin Plots**.

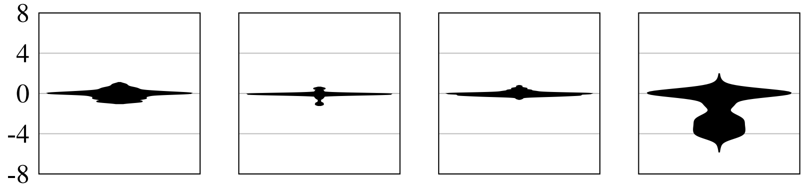
Dataset Group D



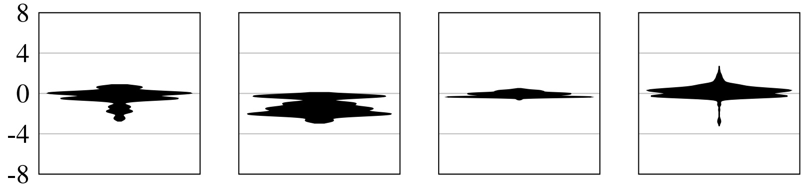
(a) Lateral position in m



(b) Absolute velocity in m/s



(c) Lateral acceleration in m/s^2

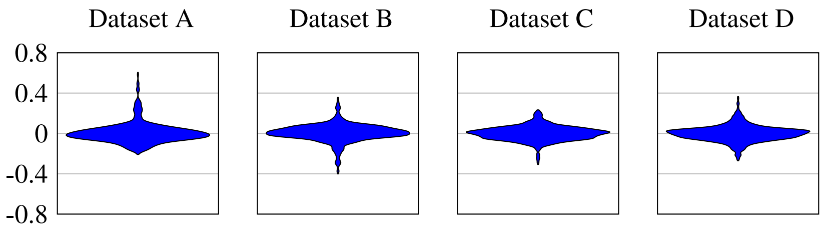


(d) Longitudinal acceleration in m/s^2

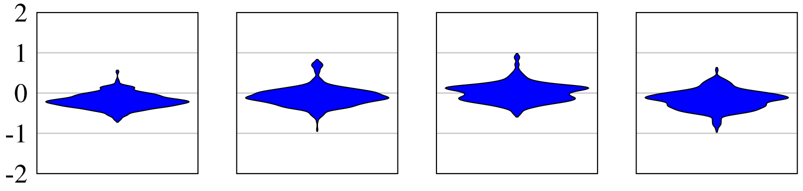
Figure A.7: **Dataset Group DViolin Plots**.

A.6 Pose Estimation Results

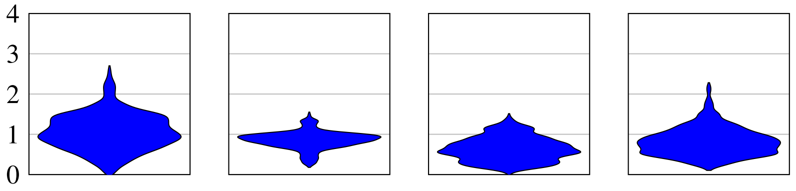
Dataset Group A Results



(a) Lateral position error in m



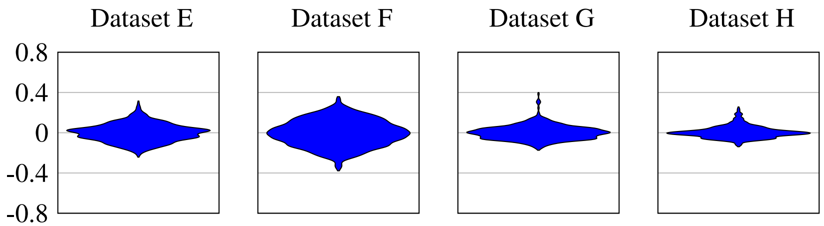
(b) UTM orientation error in deg



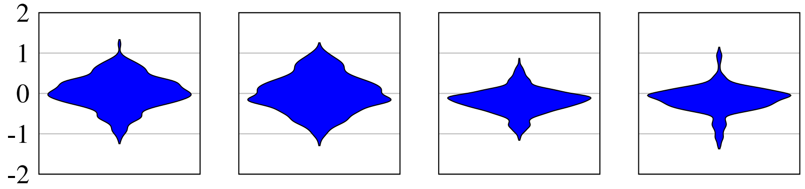
(c) UTM position error in m

Figure A.8: **Dataset Group A Violin Plots - Filter Results**. .

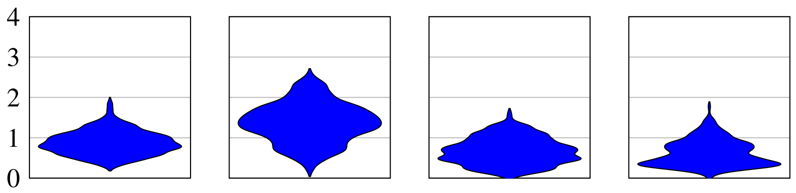
Dataset Group B Results



(a) Lateral position error in m



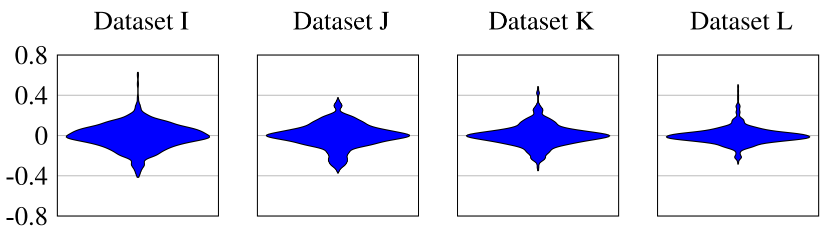
(b) UTM orientation error in deg



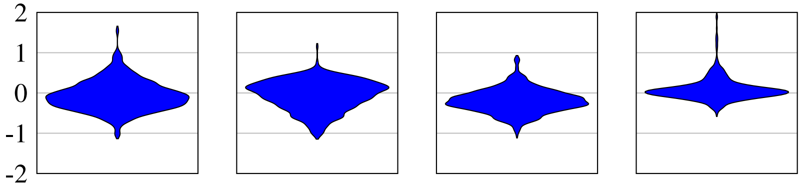
(c) UTM position error in m

Figure A.9: **Dataset Group B Violin Plots - Filter Results**. .

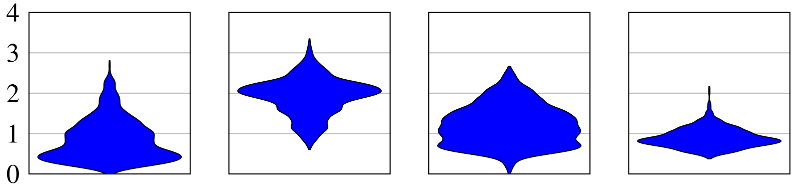
Dataset Group C Results



(a) Lateral position error in m



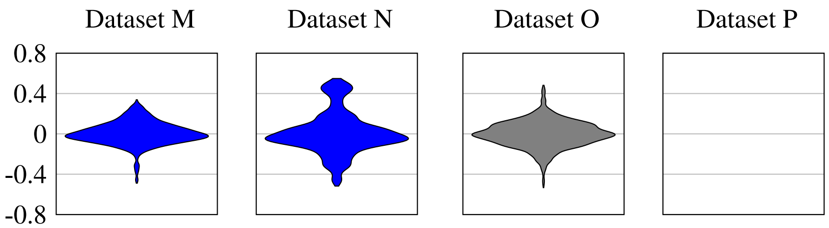
(b) UTM orientation error in deg



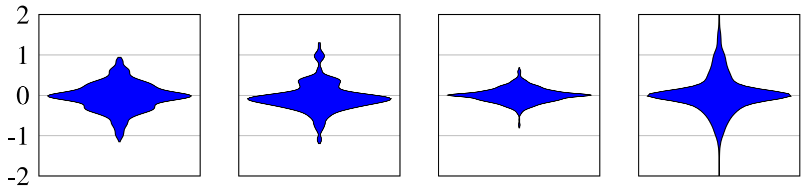
(c) UTM position error in m

Figure A.10: **Dataset Group C Violin Plots - Filter Results**. .

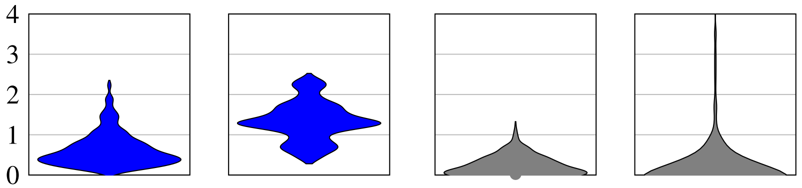
Dataset Group D Results



(a) Lateral position error in m



(b) UTM orientation error in deg



(c) UTM position error in m

Figure A.11: **Dataset Group D Violin Plots - Filter Results**. The errors in Dataset O and Dataset P are computed in absence of DGPS correction data since the sets were recorded on the public highway track. To compensate map or GNSS offset errors, the ADMA pose estimates were priorly aligned with the optimized poses. These error plots are colored gray. For Dataset P the lateral error cannot be computed precisely as the cloverleaf intersection is represented by a multitude of reference lines..

A.7 Pose Estimation Results (Tables)

	Dataset A	Dataset B	Dataset C	Dataset D
Lateral Position STD	10.04 cm	8.58 cm	7.00 cm	7.71 cm
Lateral Position Max.	58.37 cm	34.09 cm	21.91 cm	34.95 cm
Orientation STD	0.18 deg	0.23 deg	0.23 deg	0.23 deg
Orientation Max.	0.53 deg	0.79 deg	0.94 deg	0.59 deg

Table A.2: **Dataset Group A Results**.

	Dataset E	Dataset F	Dataset G	Dataset H
Lateral Position STD	8.73 cm	12.76 cm	7.10 cm	5.87 cm
Lateral Position Max.	29.74 cm	33.30 cm	38.49 cm	24.47 cm
Orientation STD	0.40 deg	0.45 deg	0.30 deg	0.33 deg
Orientation Max.	1.25 deg	1.16 deg	0.81 deg	1.08 deg

Table A.3: **Dataset Group B Results**.

	Dataset I	Dataset J	Dataset K	Dataset L
Lateral Position STD	12.35 cm	11.65 cm	10.55 cm	8.38 cm
Lateral Position Max.	60.23 cm	35.15 cm	46.40 cm	48.71 cm
Orientation STD	0.37 deg	0.36 deg	0.29 deg	0.29 deg
Orientation Max.	1.58 deg	1.15 deg	0.87 deg	2.15 deg

Table A.4: **Dataset Group C Results**.

	Dataset M	Dataset N	Dataset O	Dataset P
Lateral Position STD	11.18 cm	20.64 cm	12.69 cm	-
Lateral Position Max.	31.59 cm	50.75 cm	45.64 cm	-
Orientation STD	0.34 deg	0.35 deg	0.18 deg	0.56 deg
Orientation Max.	0.87 deg	1.23 deg	0.65 deg	2.88 deg

Table A.5: **Dataset Group D Results**.

List of Figures

List of Tables

Bibliography

[1] ABRAMOV, Alexey ; BAYER, Christopher ; HELLER, Claudio ; LOY, Claudia: Multi-lane perception using feature fusion based on GraphSLAM. In: *International Conference on Intelligent Robots and Systems (IROS)* IEEE, Daejeon, South Korea, 2016. – ISBN 978–1–5090–3762–9, S. 3108–3115

[2] ABUHASHIM, Tariq S. ; ABDEL-HAFEZ, Mamoun F. ; AL-JARRAH, Mohammad A.: Building a robust integrity monitoring algorithm for a low cost GPS-aided-INS system. In: *Springer, International Journal of Control, Automation and Systems* 8 (2010), Nr. 5, S. 1108–1122. – ISSN 2005–4092

[3] AEBERHARD, Michael ; RAUCH, Simon ; BAHRAM, Mohammad ; TANZMEISTER, Georg ; THOMAS, Julian ; PILAT, Yves ; HOMM, Florian ; HUBER, Werner ; KAEMPCHEN, Nico: Experience, Results and Lessons Learned from Automated Driving on Germany's Highways. In: *IEEE, Intelligent Transportation Systems Magazine* 7 (2015), Nr. 1, S. 42–57. – ISSN 1941–1197

[4] AGOGINO, Alice ; CHAO, Susan ; GOEBEL, Kai ; ALAG, Satnam ; CAMMON, Bradly ; WANG, Jiangxin: Intelligent Diagnosis Based On Validated And Fused Data For Relilability And Safety Enhancement Of Automated Vehicles In An IVHS. In: *California Partners for Advanced Transit and Highways (PATH)* (1998). – ISSN 1055–1425

[5] ANDERSON, Theodore W. ; MATHÉMATICIEN, Etats-Unis: *An introduction to multivariate statistical analysis*. Bd. 2. John Wiley & Sons, New York, USA, 1958. – ISBN 978–0–471–36091–9

[6] ARORA, Sanjeev ; BARAK, Boaz: *Computational complexity: a modern approach*. Bd. 1. Cambridge University Press, New York, USA, 2009. – ISBN 978–0–52142–4–264

[7] ASHBY, Neil: The sagnac effect in the global positioning system. In: *Relativity in Rotating Frames* Bd. 135. Springer, Dordrecht, Germany, 2004. – ISBN 978–9–04816–5–148, S. 11–28

[8] AUFRERE, Romuald ; CHAPUIS, Roland ; CHAUSSE, Frederic: Real Time Vision Based Road Lane Detection and Tracking. In: *IAPR Workshop on Machine Vision Applications* University of Tokio, Japan, 2000, S. 75–78

[9] BAR-SHALOM, Yaakov: *Tracking and data association*. Bd. 179. Academic Press Professional Inc., San Diego, USA, 1987. – ISBN 978–0–12079–7–608

[10] BAR-SHALOM, Yaakov ; LI, X R. ; KIRUBARAJAN, Thiagalingam: *Estimation with applications to tracking and navigation: theory algorithms and software*. Bd. 1. John Wiley & Sons, New York, USA, 2001. – ISBN 978–0–47141–6–555

[11] BAR-SHALOM, Yaakov ; WILLETT, Peter K. ; TIAN, Xin: *Tracking and data fusion*. Bd. 1. YBS Publishing, Storrs, USA, 2011. – ISBN 978–0–96483–1–278

[12] BARSHAN, Billur ; DURRANT-WHYTE, Hugh F.: Inertial navigation systems for mobile robots. In: *IEEE, Transactions on Robotics and Automation* 11 (1995), Nr. 3, S. 328–342. – ISSN 2374–958X

[13] BENGLER, Klaus ; DIETMAYER, Klaus ; FARBER, Berthold ; MAURER, Markus ; STILLER, Christoph ; WINNER, Hermann: Three decades of driver assistance systems: Review and future perspectives. In: *IEEE, Intelligent Transportation Systems Magazine* 6 (2014), Nr. 4, S. 6–22. – ISSN 1941–1197

[14] BO, Pengbo ; LUO, Gongning ; WANG, Kuanquan: A graph-based method for fitting planar B-spline curves with intersections. In: *CDE, Journal of Computational Design and Engineering* 3 (2016), Nr. 1, S. 14–23. – ISSN 2288–5048

[15] BOLIC, Miodrag ; DJURIC, Petar M. ; HONG, Sangjin: Resampling algorithms for particle filters: A computational complexity perspective. In: *EURASIP, Journal on Advances in Signal Processing* 15 (2004), Nr. 15, S. 2267–2277. – ISSN 1687–6180

[16] BOYD, Stephen ; VANDENBERGHE, Lieven: *Convex optimization*. Bd. 1. Cambridge University Press, Campbridge, UK, 2004. – ISBN 978–0–52183–3–783

[17] BUEHLER, Martin ; IAGNEMMA, Karl ; SINGH, Sanjiv: *The DARPA urban challenge: autonomous vehicles in city traffic*. Bd. 56. Springer-Verlag, Berlin, Deutschland, 2009. – ISBN 978–3–642–03990–4

[18] BURKARD, Rainer E. ; ZIMMERMANN, Uwe T.: *Einführung in die mathematische Optimierung*. Bd. 1. Springer-Verlag, Berlin, Germany, 2013. – ISBN 978–3–642–28762–8

[19] CANNY, John: A computational approach to edge detection. In: *IEEE, Transactions on pattern analysis and machine intelligence* 8 (1986), Nr. 6, S. 679–698. – ISSN 1939–3539

[20] CARTER, G ; KNAPP, C: Time delay estimation. In: *International Conference on Acoustics, Speech, and Signal Processing* Bd. 1 IEEE, Philadelphia, USA, 1976. – ISBN 978–9–40095–361–1, S. 357–360

[21] CHAO, Chih-Hao ; CHU, Chun-Yuan ; WU, An-Yeu: Location-constrained particle filter human positioning and tracking system. In: *Workshop on Signal Processing Systems* IEEE, Washington, USA, 2008. – ISBN 978–1–42442–923–3, S. 73–76

[22] CHAO, Min-An ; CHU, Chun-Yuan ; CHAO, Chih-Hao ; WU, An-Yeu: Efficient parallelized particle filter design on CUDA. In: *Workshop on Signal Processing Systems (SIPS)* IEEE, San Francisco, USA, 2010. – ISBN 978–1–4244–8934–3, S. 299–304

[23] CHAUSSE, Frederic ; LANEURIT, Jean ; CHAPUIS, Roland: Vehicle localization on a digital map using particles filtering. In: *Intelligent Vehicles Symposium* IEEE, Las Vegas, USA, 2005. – ISBN 0–7803–8961–1, S. 243–248

[24] CHEESEMAN, P ; SMITH, R ; SELF, M: A stochastic map for uncertain spatial relationships. In: *4th International Symposium on Robotic Research* MIT Press Cambridge, Santa Clara, USA, 1987. – ISBN 0–262–02272–9, S. 467–474

[25] CHEN, Lingji ; ARAMBEL, Pablo O. ; MEHRA, Raman K.: Estimation under unknown correlation: covariance intersection revisited. In: *IEEE, Transactions on Automatic Control* 47 (2002), Nr. 11, S. 1879–1882. – ISSN 1558–2523

[26] CLIFFORD, M.A.: Preliminary sketch of biquaternions. In: *Oxford Academic, Proceedings of the London Mathematical Society* S1-4 (1871), S. 381–395. – ISSN 1460–244X

[27] COPPERSMITH, Don ; WINOGRAD, Shmuel: Matrix multiplication via arithmetic progressions. In: *Elsevier, Journal of Symbolic Computation* 9 (1990), Nr. 3, S. 225–403. – ISSN 0747–7171

[28] DAWOOD, Maya ; CAPPELLE, Cindy ; EL NAJJAR, Maan E. ; KHALIL, Mohamad ; POMORSKI, Denis: Vehicle geo-localization based on IMM-UKF data fusion using a GPS receiver, a video camera and a 3D city model. In: *IEEE, Intelligent Vehicles Symposium* IEEE, Baden-Baden, Germany, 2011. – ISBN 978–1–4577–0891–6, S. 510–515

[29] DEMENTHON, Daniel F. ; DAVIS, Larry S.: Model-based object pose in 25 lines of code. In: *Kluwer Acadmic Publishers, International journal of computer vision* 15 (1995), Nr. 1-2, S. 123–141. – ISSN 1573–1405

[30] DEUSCH, Hendrik ; REUTER, Stephan ; NUSS, Dominik ; FRITZSCHE, Martin ; DIETMAYER, Klaus u. a.: Multi-sensor self-localization based on maximally stable extremal regions. In: *Intelligent Vehicles Symposium* IEEE, Dearborn, USA, 2014. – ISBN 978–1–4799–3638–0, S. 555–560

[31] DEUSCH, Hendrik ; WIEST, Jürgen ; REUTER, Stephan ; SZCZOT, Magdalena ; KONRAD, Marcus ; DIETMAYER, Klaus: A random finite set approach to multiple lane detection. In: *International Conference on Intelligent Transportation Systems* IEEE, Anchorage, USA, 2012. – ISBN 978–1–4673–3063–3, S. 270–275

[32] DICKMANNS, Ernst D. ; MYSLIWETZ, Birger ; CHRISTIANS, Thomas: An integrated spatio-temporal approach to automatic visual guidance of autonomous vehicles. In: *IEEE, Transactions on Systems, Man, and Cybernetics* 20 (1990), Nr. 6, S. 1273–1284. – ISSN 2168–2909

[33] DICKMANNS, Ernst D. ; BEHRINGER, Reinhold ; DICKMANNS, Dirk ; HILDEBRANDT, Thomas ; MAURER, Markus ; THOMANEK, Frank ; SCHIEHLEN, Joachim: The seeing passenger car 'VaMoRs-P'. In: *Proceedings of the Intelligent Vehicles Symposium* IEEE, Paris, France, 1994. – ISBN 0–7803–2135–9, S. 68–73

[34] DUPUIS, Marius ; STROBL, Martin ; GREZLIKOWSKI, Hans: OpenDRIVE 2010 and Beyond–Status and Future of the de facto Standard for the Description of Road Networks. In: *Proceedings of the Driving Simulation Conference* DSA, Paris, France, 2010. – ISBN 978–2–85782–685–9, S. 231–242

[35] FIKE, Jeffrey A. ; ALONSO, Juan J.: Automatic differentiation through the use of hyper-dual numbers for second derivatives. In: *Recent Advances in Algorithmic Differentiation* Bd. 87. Springer, Berlin, Germany, 2012. – ISBN 978–3–642–30023–3, S. 163–173

[36] GAO, J ; PETOVELLO, MG ; CANNON, ME: Development of precise GPS/INS/wheel speed sensor/yaw rate sensor integrated vehicular positioning system. In: *Proceedings of the National Technical Meeting of the Institute of Navigations* Institute of Navigation, Monterey, USA, 2006. – ISBN 978–1–60423–505–0, S. 780–792

[37] GEIGER, Andreas ; ZIEGLER, Julius ; STILLER, Christoph: Stereoscan: Dense 3d reconstruction in real-time. In: *Intelligent Vehicles Symposium* IEEE, Baden-Baden, Germany, 2011. – ISBN 978–1–4577–0891–6, S. 963–968

[38] GEORGY, Jacques ; KARAMAT, Tashfeen ; IQBAL, Umar ; NOURELDIN, Aboelmagd: Enhanced MEMS-IMU/odometer/GPS integration using mixture particle filter. In: *Springer-Verlag, GPS Solutions* 15 (2011), Nr. 3, S. 239–252. – ISSN 1521–1886

[39] Opel Testcenter, Dudenhofen, Germany. 49°59'29"N, 8°55'06"E, Eye alt 4.66 km. GeoBasis-DE 2009 Google. http://www.earth.google.com [December 01, 2017].

[40] Ko-HAF Testfield, Frankfurt, Germany. 50°05’16”N, 8°42’47”E, Eye alt 58.00 km. GeoBasis-DE 2009 Google. http://www.earth.google.com [December 01, 2017].

[41] GOULD, Nick ; ORBAN, Dominique ; TOINT, Philippe: Numerical methods for large-scale nonlinear optimization. In: *Cambridge University Press, Acta Numerica* 14 (2005), S. 299–361. – ISSN 0962–4929

[42] HAMERLY, Greg ; ELKAN, Charles: Learning the k in k-means. In: *International Conference on Neural Information Processing Systems* Whistler, British Columbia, Canada, 2003. – ISBN 978–0–26220–152–0, S. 281–288

[43] HAMILTON, William R.: *Elements of quaternions*. Bd. 3. Longmans, Green, & Company, London, UK, 1866. – ISBN 978–0–82840–219–4

[44] HAYKIN, Simon: *Kalman filtering and neural networks*. Bd. 47. John Wiley & Sons Inc., New York, USA, 2004. – ISBN 978–0–47136–998–1

[45] HAZLETT, Andrew C. ; CRASSIDIS, John L. ; FUGLEWICZ, Daniel P. ; MILLER, Patrick: Differential wheel speed sensor integration with GPS/INS for land vehicle navigation. In: *Guidance, Navigation, and Control Conference* AIAA, Portland, Oregon, 2013. – ISBN 978–1–62993–151–7, S. 6577–6596

[46] HELLMUND, Andre-Marcel ; WIRGES, Sascha ; TAS, Ömer S. ; BANDERA, Claudio ; SALSCHEIDER, Niels O.: Robot operating system: A modular software framework for automated driving. In: *19th International Conference on Intelligent Transportation Systems* IEEE, Rio de Janeiro, Brasil, 2016. – ISBN 978–1–5090–1889–5, S. 1564–1570

[47] HENDEBY, Gustaf ; KARLSSON, Rickard ; GUSTAFSSON, Fredrik: Particle filtering: the need for speed. In: *EURASIP, Journal on Advances in Signal processing* 2010 (2010), Nr. 181. – ISSN 1687–6180

[48] JEONG, Jinyong ; CHO, Younggun ; KIM, Ayoung: Road-SLAM: Road marking based SLAM with lane-level accuracy. In: *Intelligent Vehicles Symposium* IEEE, Los Angeles, USA, 2017. – ISBN 978–1–50904–804–5, S. 1736–1473

[49] JO, Kichun ; JO, Yongwoo ; SUHR, Jae K. ; JUNG, Ho G. ; SUNWOO, Myoungho: Precise Localization of an Autonomous Car Based on Probabilistic Noise Models of Road Surface Marker Features Using Multiple Cameras. In: *IEEE, Transactions on Intelligent Transportation Systems* 16 (2015), Nr. 6, S. 3377 – 3392. – ISSN 1558–0016

[50] JULIER, Simon ; UHLMANN, Jeffrey ; DURRANT-WHYTE, Hugh F.: A new method for the nonlinear transformation of means and covariances in filters and estimators. In: *IEEE, Transactions on automatic control* 45 (2000), Nr. 3, S. 477–482. – ISSN 1558–2523

[51] JULIER, Simon J. ; UHLMANN, Jeffrey K.: A non-divergent estimation algorithm in the presence of unknown correlations. In: *American Control Conference, 1997. Proceedings of the 1997* Bd. 4 IEEE, 1997, S. 2369–2373

[52] KALMAN, Rudolf E. u. a.: Contributions to the theory of optimal control. In: *Sociedad Matematica Mexicana, Boletin de la Sociedad Matematica Mexicana* 5 (1960), Nr. 2, S. 102–119

[53] KHALEGHI, Bahador ; KHAMIS, Alaa ; KARRAY, Fakhreddine O. ; RAZAVI, Saiedeh N.: Multisensor data fusion: A review of the state-of-the-art. In: *Elsevier, Information fusion* 14 (2013), Nr. 1, S. 28–44. – ISSN 1566–2535

[54] KUNZ, Felix ; NUSS, Dominik ; WIEST, Jürgen ; DEUSCH, Hendrik ; REUTER, Stephan ; GRITSCHNEDER, Franz ; SCHEEL, Alexander ; STÜBLER, Manuel ; BACH, Martin ; HATZELMANN, Patrick ; WILD, Cornelius ; DIETMAYER, Klaus: Autonomous driving at Ulm University: A modular, robust, and sensor-independent fusion approach. In: *Intelligent vehicles symposium* IEEE, Seoul, South Korea, 2015. – ISBN 978–1–46737–266–4, S. 666–673

[55] LANEURIT, J ; BLANC, C ; CHAPUIS, R ; TRASSOUDAINE, L: Multisensorial data fusion for global vehicle and obstacles absolute positioning. In: *Intelligent Vehicles Symposium* IEEE, Columbus, USA, 2003. – ISBN 0–780–37848–2, S. 138–143

[56] LANGLEY, Richard B. u. a.: Dilution of precision. In: *GPS world* 10 (1999), Nr. 5, S. 52–59. – ISSN 1048–5104

[57] LATEGAHN, Henning ; SCHREIBER, Markus ; ZIEGLER, Julius ; STILLER, Christoph: Urban localization with camera and inertial measurement unit. In: *Intelligent Vehicles Symposium* IEEE, Gold Coast, Australia, 2013. – ISBN 978–1–46732–755–8, S. 719–724

[58] LEVINSON, Jesse ; MONTEMERLO, Michael ; THRUN, Sebastian: Map-Based Precision Vehicle Localization in Urban Environments. In: *Proceedings of Robotics: Science and Systems* MIT Press, Atlanta, USA, 2007. – ISBN 978–0–262–52484–1

[59] LIU, Jun S. ; CHEN, Rong: Sequential Monte Carlo methods for dynamic systems. In: *JSTOR, Journal of the American statistical association* 93 (1998), Nr. 443, S. 1032–1044. – ISSN 1537–274X

[60] LONGUET-HIGGINS, H C.: A computer algorithm for reconstructing a scene from two projections. In: *Nature, International journal of science* 293 (1981), Nr. 5828, S. 133–135. – ISSN 1476–4687

[61] MAHALANOBIS, Prasanta C.: On the generalized distance in statistics. In: *Proceedings of the National Institute of Sciences of India* 2 (1936), Nr. 1, S. 49–55. – ISBN 0470–1399

[62] MARKLEY, F L. ; CHENG, Yang ; CRASSIDIS, John L. ; OSHMAN, Yaakov: Averaging quaternions. In: *AIAA, Journal of Guidance Control and Dynamics* 30 (2007), Nr. 4, S. 1193–1197. – ISSN 0731–5090

[63] MEINHOLD, Richard J. ; SINGPURWALLA, Nozer D.: Robustification of Kalman filter models. In: *ASA, Journal of the American Statistical Association* 84 (1989), Nr. 406, S. 479–486. – ISSN 1537–274X

[64] MELHI, Muhammed ; IPSON, Stanley S. ; BOOTH, William: A novel triangulation procedure for thinning hand-written text. In: *IAPR, Pattern Recognition Letters* 22 (2001), Nr. 10, S. 1059–1071. – ISSN 0167–8655

[65] NEUBECK, Alexander ; VAN GOOL, Luc: Efficient non-maximum suppression. In: *18th International Conference on Pattern Recognition* Bd. 3 IEEE, Hong Kong, China, 2006. – ISBN 0–769–52521–0, S. 850–855

[66] NISTÉR, David ; NARODITSKY, Oleg ; BERGEN, James: Visual odometry. In: *Computer Society Conference on Computer Vision and Pattern Recognition* IEEE, Washington, USA, 2004. – ISBN 0–769–52158–4, S. 652–659

[67] OLSON, Edwin ; LEONARD, John ; TELLER, Seth: Fast iterative alignment of pose graphs with poor initial estimates. In: *International Conference on Robotics and Automation* IEEE, Orlando, USA, 2006. – ISBN 0–780–39505–0, S. 2262–2269

[68] POMERLEAU, Dean: RALPH: Rapidly adapting lateral position handler. In: *Intelligent Vehicles Symposium* IEEE, Detroit, USA, 1995. – ISBN 0–780–32983–X, S. 506–511

[69] POMERLEAU, Dean ; JOCHEM, Todd: Rapidly adapting machine vision for automated vehicle steering. In: *IEEE, IEEE Expert* 11 (1996), Nr. 2, S. 19–27. – ISSN 2374–9407

[70] RAJAMANI, Rajesh: *Vehicle dynamics and control*. Bd. 2. Springer Science & Business Media, New York, USA, 2011. – ISBN 978–1–46141–433–9

[71] REINHARDT, Marc ; NOACK, Benjamin ; HANEBECK, Uwe D.: Closed-form optimization of covariance intersection for low-dimensional matrices. In: *15th International Conference on Information Fusion* IEEE, Singapore, Singapore, 2012. – ISBN 978–0–98244–385–9, S. 1891–1896

[72] RIDDERS, CJF: Accurate computation of F'(x) and F'(x) F"(x). In: *Elsevier, Advances in Engineering Software* 4 (1982), Nr. 2, S. 75–76. – ISSN 0965–9978

[73] SCHINDLER, Andreas: Vehicle self-localization with high-precision digital maps. In: *Intelligent Vehicles Symposium* IEEE, Gold Coast, Australia, 2013. – ISBN 978–1–47990–795–3, S. 141–146

[74] SCHNEIDER, Nick ; PIEWAK, Florian ; STILLER, Christoph ; FRANKE, Uwe: RegNet: Multimodal sensor registration using deep neural networks. In: *Intelligent Vehicles Symposium* IEEE, Los Angeles, USA, 2017. – ISBN 978–1–50904–804–5, S. 1803–1810

[75] SIMON, Dan ; CHIA, Tien L.: Kalman filtering with state equality constraints. In: *IEEE, Transactions on Aerospace and Electronic Systems* 38 (2002), Nr. 1, S. 128–136. – ISSN 1557–9603

[76] STATISTISCHES BUNDESAMT, Destatis: Verkehrsunfaelle. Fachserie 8 Reihe 7 (2018), August, S. 1–50

[77] STRASSEN, Volker: Gaussian elimination is not optimal. In: *Springer, Numerische mathematik* 13 (1969), Nr. 4, S. 354–356. – ISSN 0945–3245

[78] STRAUSS, Tobias ; ZIEGLER, Julius ; BECK, Johannes: Calibrating multiple cameras with non-overlapping views using coded checkerboard targets. In: *17th International Conference on Intelligent Transportation Systems* IEEE, Qingdao, China, 2014. – ISBN 978–1–47996–078–1, S. 2623–2628

[79] SUENDERHAUF, Niko: *Robust optimization for simultaneous localization and mapping*, Diss., 2012

[80] SUKKARIEH, Salah ; NEBOT, Eduardo M. ; DURRANT-WHYTE, Hugh F.: A high integrity IMU/GPS navigation loop for autonomous land vehicle applications. In: *IEEE, Transactions on Robotics and Automation* 15 (1999), Nr. 3, S. 572–578. – ISSN 2374–958X

[81] SÜNDERHAUF, Niko ; PROTZEL, Peter: Switchable constraints for robust pose graph SLAM. In: *International Conference on Intelligent Robots and Systems* IEEE, Vilamoura, Portugal, 2012. – ISBN 978–1–46731–736–8, S. 1879–1884

[82] THRUN, Sebastian ; BURGARD, Wolfram ; FOX, Dieter: *Probabilistic robotics*. MIT press, Cambridge, England, 2006. – ISBN 978–0–26220–162–9

[83] THRUN, Sebastian ; MONTEMERLO, Michael: The graph SLAM algorithm with applications to large-scale mapping of urban structures. In: *SAGE, The International Journal of Robotics Research* 25 (2006), Nr. 5-6, S. 403–429. – ISSN 0278–3649

[84] THRUN, Sebastian ; MONTEMERLO, Mike ; DAHLKAMP, Hendrik ; STAVENS, David ; ARON, Andrei ; DIEBEL, James ; FONG, Philip ; GALE,

John ; HALPENNY, Morgan ; HOFFMANN, Gabriel u. a.: Stanley: The robot that won the DARPA Grand Challenge. In: *Wiley Periodicals, Journal of field Robotics* 23 (2006), Nr. 9, S. 661–692. – ISSN 1556–4967

[85] TITTERTON, David ; WESTON, John L.: *Strapdown inertial navigation technology*. 2. The Institution of Engineers, Stevenage, United Kingdom, 2004. – ISBN 978–0–86341–358–2

[86] TOLEDO-MOREO, Rafael ; ZAMORA-IZQUIERDO, Miguel ; ÚBEDA-MIÑARRO, Benito ; GÓMEZ-SKARMETA, Antonio F. u. a.: High-integrity IMM-EKF-based road vehicle navigation with low-cost GPS/SBAS/INS. In: *IEEE, Transactions on Intelligent Transportation Systems* 8 (2007), Nr. 3, S. 491–511. – ISSN 1558–0016

[87] URMSON, Chris ; ANHALT, Joshua ; BAGNELL, Drew ; BAKER, Christopher ; BITTNER, Robert ; CLARK, MN ; DOLAN, John ; DUGGINS, Dave ; GALATALI, Tugrul ; GEYER, Chris u. a.: Autonomous driving in urban environments: Boss and the urban challenge. In: *Wiley Periodicals, Journal of Field Robotics* 25 (2008), Nr. 8, S. 425–466. – ISSN 1556–4967

[88] VELDKAMP, GR: On the use of dual numbers, vectors and matrices in instantaneous, spatial kinematics. In: *Elsevier, Mechanism and Machine Theory* 11 (1976), Nr. 2, S. 141–156. – ISSN 0094–114X

[89] VO, Ba-Ngu ; MA, Wing-Kin: The Gaussian mixture probability hypothesis density filter. In: *IEEE, Transactions on signal processing* 54 (2006), Nr. 11, S. 4091–4104. – ISSN 1941–0476

[90] WANG, Yue ; TEOH, Eam K. ; SHEN, Dinggang: Lane detection and tracking using B-Snake. In: *Elsevier, Image and Vision computing* 22 (2004), Nr. 4, S. 269–280. – ISSN 0262–8856

[91] WENDEL, Jan: *Integrierte Navigationssysteme: Sensordatenfusion, GPS und Inertiale Navigation*. Oldenbourg Wissenschaftsverlag GmbH, Munich, Germany, 2011. – ISBN 978–3–48658–160–7

[92] WERRIES, Adam ; DOLAN, John M.: Adaptive Kalman Filtering Methods for Low-Cost GPS/INS Localization for Autonomous Vehicles. In: *Carnegie Mellon University* (2016). `http://dx.doi.org/10.1184/R1/6551687.v1`. – DOI 10.1184/R1/6551687.v1

[93] WESTENBERGER, Antje ; HUCK, Tobias ; FRITZSCHE, Martin ; SCHWARZ, Tilo ; DIETMAYER, Klaus: Temporal synchronization in multi-sensor fusion for future driver assistance systems. In: *International Symposium on Precision Clock Synchronization for Measurement, Control and Communication* IEEE, Munich, Germany, 2011. – ISBN 978–1–61284–893–8, S. 93–98

[94] WRIGHT, Stephen ; NOCEDAL, Jorge: *Numerical optimization*. Bd. 35. Springer Science+Business Media, New York, USA, 1999. – ISBN 978–0387–30303–1

[95] ZHANG, Jason ; ZHANG, Kefei ; GRENFELL, Ron ; DEAKIN, Rod: Short note: On the relativistic Doppler effect for precise velocity determination using GPS. In: *Springer, Journal of Geodesy* 80 (2006), Nr. 2, S. 104–110. – ISSN 1432–1394

[96] ZHANG, Xiaohai: A Fast Numerical Method for the Optimal Data Fusion in the Presence of Unknown Correlations. In: *21th International Conference on Information Fusion* IEEE, Cambridge, UK, 2018. – ISBN 978–0–99645–276–2

[97] ZIEGLER, Julius ; BENDER, Philipp ; SCHREIBER, Markus ; LATEGAHN, Henning ; STRAUSS, Tobias ; STILLER, Christoph ; DANG, Thao ; FRANKE, Uwe ; APPENRODT, Nils ; KELLER, Christoph G. u. a.: Making Bertha drive - An autonomous journey on a historic route. In: *IEEE, Intelligent Transportation Systems Magazine* 6 (2014), Nr. 2, S. 8–20. – ISSN 1941–1197

Publications and Supervised Thesis

[Int1] Maximilian Harr, Klaus-Dieter Mueller, Andre-Marcel Hellmund, and Nikolas Wagner. Robust localization on highways using low-cost gnss, front/rear mono camera and digital maps. In *Automotive meets Electronics*, volume 8, pages 20–26. VDE Verlag, Dortmund, Germany, 2018.

[Int2] Maximilian Harr and Christoph Schaefer. Robust odometry estimation for automated driving and map data collection. In *Automatisiertes Fahren und vernetzte Mobilitaet (AutoReg), 2017 VDI VDE Conference*, volume 8, pages 91–102. VDI VDE, 2017.

[Int3] Maximilian Harr, Johannes Janosovits, Sascha Wirges, and Christoph Stiller. Fast and robust vehicle pose estimation by optimizing multiple pose graphs. In *International Conference on Information Fusion*, pages 1707–1714. IEEE, Cambridge, UK, 2018.

[Int4] Margit Krauss. High precision localisation using kalman filter. Master's thesis, Technische Universität Kaiserslautern, 2017.

[Int5] Christoph Schaefer. Fault detection and exclusion in redundant sensor systems for vehicle localization. Master's thesis, Technische Universität Darmstadt, 2017.

[Int6] Klaus-Dieter Mueller. Robust localization for autonomous driving on highways using particle filtering and digital maps. Master's thesis, Technische Universität Darmstadt, 2017.

[Int7] Malena Kellermann. Robust localization for autonomous driving on highways using pose graph optimization. Master's thesis, Technische Universität Darmstadt, 2017.

[Int8] Moritz Nakatenus. Digital map validation for hd-maps on highways. Master's thesis, Technische Universität Darmstadt, 2018.