Dynamic Switching State Systems for Visual Tracking

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Stefan Becker

aus Lebach

Tag der mündlichen Prüfung:	24.04.2020
Erster Gutachter:	Prof. Dr.-Ing. Jürgen Beyerer
Zweiter Gutachter:	Prof. Ph.D. Brendan T. Morris

Abstract

Estimating the motion state of objects is a central component of most visual tracking pipelines. Therefore, object observations provided by an appearance model, representing the object in image space, serve as input for the actual filtering and the prediction into future frames. Under real-life conditions, the dynamics of tracked objects are subject to change over time. Especially in such maneuver scenarios, current methods struggle to deal with the model mismatch due to varying system characteristics.

This thesis addresses the problem of how to capture the dynamics of maneuvering objects in an efficient and reactive way. Towards this end, the perspective of recursive Bayesian filters and the perspective of deep learning approaches on state estimation are considered and their functional viewpoints are brought together.

The starting point of this thesis is the **interacting multiple-model** (IMM) filter, as the most common representative Bayesian formulation for dealing with model mismatches or rather maneuvering objects. For a model mismatch scenario, in which tracking is done directly in image space, a state de-coupling and a re-coupling scheme are introduced as modifications for an improved design compared to the standard IMM filter.

In order to deal with two maneuver types, switching noise levels and switching dynamics, **recurrent neural network** (RNN)-based approaches are proposed as alternatives to IMM filtering. The approaches maintain the functionality of an IMM filter while reducing the amount of required filter tuning. With a focus on applications in the surveillance and intelligent vehicle domains, the effectiveness of RNN-based solutions is demonstrated for the exemplary tasks of *path prediction* and *intention prediction*, reflecting the most

common prototypical maneuver types. The presented RNN-based network yields performance comparable to other existing relevant methods on a public benchmark. The suggested modifications help to achieve a robust prediction performance with regard to switching noise levels. For sudden motion changes, a proposed RNN-based IMM surrogate can capture the change in the dynamical behavior mare reliably than the Bayesian filter counterparts. The abilities of the RNN-IMM are evaluated in extensive experiments on real-world and synthetic datasets, reflecting prototypical maneuver situations of pedestrians in the application domain of intelligent vehicles.

Kurzfassung

Die Schätzung des Bewegungszustands von Objekten ist eine zentrale Komponente für die video-basierte Objektverfolgung. Dabei werden Objektbeobachtungen, die von einem Erscheinungsmodell geliefert werden und das Objekt im Bildraum repräsentieren, als Eingabe für die Filterung und die Vorhersage in zukünftige Frames verwendet. Unter realen Bedingungen variiert die Dynamik des verfolgten Objektes über die Zeit. Besonders in solchen Manöversituationen haben aktuelle Methoden wegen Modellfehlanpassungen aufgrund der variierenden Systemeigenschaften Schwierigkeiten den Bewegungszustand des Objektes zu schätzen.

Diese Arbeit befasst sich mit dem Problem der effizienten und reaktiven Erfassung der Dynamik von manövrierenden Objekten. Zu diesem Zweck werden die Perspektive rekursiver Bayes'scher Filter und die Perspektive tiefer lernender Ansätze zur Zustandsschätzung betrachtet und ihre funktionalen Sichtweisen zusammengeführt.

Ausgangspunkt dieser Arbeit ist das **interacting multiple-model** (IMM)-Filter, als einer der am häufigsten verwendete Ansätze basierend auf einer Bayes'sche Formulierung zum Umgang mit Modellfehlanpassungen bzw. manövrierenden Objekten. Für ein Modellfehlanpassungsszenario, bei dem die Objektverfolgung direkt im Bildraum erfolgt, werden eine Zustandsentkopplung und ein Rückkopplungsschema als Modifikationen für ein verbessertes Design im Vergleich zum Standard-IMM-Filter eingeführt. Zum besseren Umgang mit den zwei Manövertypen von variierenden Rauschpegeln und

variierenden Objektdynamiken werden **recurrent neural network** (RNN)-basierte Ansätze als Alternative zum IMM-Filter vorgestellt. Die Ansätze bilden die Funktionalität eines IMM-Filters ab und reduzieren gleichzeitig den Umfang der erforderlichen Filterabstimmung.

Mit dem Schwerpunkt auf Anwendungen in den Bereichen Videoüberwachung und intelligente Fahrzeuge wird die Wirksamkeit der vorgestellten RNN-basierten Ansätze exemplarisch für Aufgabenstellungen der *Pfadvorhersage* und der *Intentionsvorhersage* demonstriert. Die ausgewählten Anwendungen spiegeln prototypische Manöversituationen wieder. Ein vorgestelltes RNN-basiertes Netzwerk erzielt eine Leistung vergleichbar mit relevanten Methoden auf dem aktuellen Stand der Technik auf einem öffentlichen Benchmark. Die vorgeschlagenen Modifikationen tragen dazu bei eine robuste Vorhersageleistung in Bezug auf die Rauschpegel zu erreichen. Bei plötzlichen Bewegungsänderungen kann ein vorgeschlagenes RNN-basiertes IMM-Surrogat die Änderung im dynamischen Verhalten zuverlässiger erfassen als die Bayes'sche Filter Pendants. Die Fähigkeiten des RNN-IMM werden in umfangreichen Experimenten auf realen und synthetischen Datensätzen, die prototypische Manöversituationen von Fußgängern im Anwendungsbereich intelligenter Fahrzeuge widerspiegeln, evaluiert.

Acknowledgements

This thesis is the result of my work in the department *Object Recognition* at the *Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB*. I was in the fortunate position of receiving much individual support in a variety of ways.

First of all, I would like to thank my advisor Prof. Dr.-Ing. Jürgen Beyerer for his guidance and feedback, which were invaluable to complete this thesis. I am grateful to Prof. Ph.D. Brendan T. Morris, Prof. Dr. Bernhard Beckert, Prof. Ph.D. Mehdi B. Tahoori, and Prof. Dr. Peter Sanders for agreeing to be part of my examination committee. In particular, I would like to thank Ph.D. Brendan T. Morris for his interest in my work and for being in the committee as a second advisor.

Special thanks go to my supervisors Dr. Wolfgang Hübner and Dr. Michael Arens at *Fraunhofer IOSB* for providing conditions, feedback, and freedom to prepare this thesis.

I want to acknowledge my colleagues of the department *Object Recognition* for their constant assistance and in particular, my colleagues of the *Video Content Analysis* group. The support and the conversations were essential for solving various technical challenges.

Lastly, I want to thank my family and friends for their advice and their encouragement throughout all phases of this work.

Contents

Notation

This chapter introduces the notation and symbols which are used in this thesis.

General notation

Scalars	italic Roman and Greek lowercase letters	x, α
Sets	calligraphic Roman uppercase letters	$\mathcal{D}$
Vectors	bold Roman lowercase letters	$\mathbf{t}$
Matrices	bold Roman uppercase letters	$\mathbf{R}$
State spaces	bold calligraphic Roman uppercase letters	$\boldsymbol{\mathcal{X}}$

In multidimensional sets of elements related to time series, the first superscript index denotes time.

Distributions

$\mathcal{N}$	Gaussian distribution
$\mathcal{B}in$	Binomial distribution

Numbers, indexing and conventions

$\mathbb{N}$	natural numbers
$\mathbb{R}$	real numbers
k, t	discrete points in time
i, j, ℓ, q	indexing for objects, observations and points
$\lceil \cdot \rceil$	ceil operator, the least integer greater than or equal to the value.

State modeling and probabilities

$\mathcal{X}$	(dynamical) state-space
$\mathcal{H}$	(recurrent) state-space
$\mathcal{Z}$	observation space
$\mathcal{Y}$	target space
$f(\cdot)$	dynamical model
$h(\cdot)_{obs}$	observation model
$\mathbf{F}$	system matrix of the Kalman Filter
$\mathbf{G}$	noise gain matrix of the Kalman Filter
$\mathbf{H}$	observation matrix of the Kalman Filter
$\mathbf{K}$	Kalman gain
$\mathbb{E}[\cdot]$	expectation value
$\mathbf{x}^k$	(dynamical) state vector at time k

$\mathbf{h}^k$	(recurrent) state vector at time k
$\mathbf{z}^k$	observation vector at time k
$\mathbf{y}^k$	target vector at time k
m^k	dynamical mode at time k
$\mathbf{v}^k$	process noise at time k
$\mathbf{w}^k$	observation noise at time k
$\mathbf{Q}^k$	process noise covariance matrix at time k
$\mathbf{R}^k$	observation noise covariance matrix at time k
$\mathbf{P}$	covariance matrix
$\mathbf{P}_{\mathbf{xx}}^k$	(dynamical) state covariance matrix
$\mathbf{P}_{\mathbf{zz}}^k$	observation covariance matrix
$\mathbf{P}_{\mathbf{xx}}^{k,-}$	prior probability
$\mathbf{P}_{\mathbf{xx}}^{k,+}$	posterior probability
$p(\mathbf{x}^k)$	probability density function (pdf)
$P(m^k)$	probability mass function (pmf)
$p(\mathbf{x}^{k+1}\|\mathbf{x}^k,...)$	transition density
$P(m^{k+1}\|m^k,...)$	transition probability

1 Introduction

One fundamental ability essential for intelligent autonomous systems to see, understand, and react to the environment is to track objects of interest in image sequences. Its applications cover a broad range from intelligent vehicles to robot navigation and smart video surveillance. For example, the ability to anticipate the actions of pedestrians in a scene and to predict their future positions is a safety issue for autonomous vehicles and other vision-based active safety systems.

Figure 1.1: Scenes captured from an approaching vehicle, the most important question being whether *the pedestrian is going to cross the street.* Traditionally, such questions are tackled with adaptive recursive Bayesian filters [Sch13].

Despite enormous advances in extracting observations of objects from images due to deep learning, the actual filtering and the prediction into future frames are mainly restricted to the application of recursive Bayesian filters. The problem-specific choice of their design parameters, such as connecting

the object motion uncertainty and its predictability to physical system parameters or the observation uncertainty, requires not only well-suited physical models, but also a large amount of engineering. Especially in situations where the tracked objects perform a maneuver, this has proved a challenging task. A maneuver is any motion characteristic that an object is performing other than the dynamical model used by the filter. An illustrative example for a variation in the dynamics, which poses significant challenges for the filters to adapt, is to determine if a pedestrian is going to cross the street (see figure 1.1). Such situations additionally require the choice of various adequate dynamical models, including associated transition modeling.

Towards this end, the overarching research question of this thesis is how to capture the dynamics of maneuvering objects in image sequences effectively. Maneuvering objects can be defined by either being subject to random perturbations i.e., different noise levels or subject to sudden motion changes.

The dynamical model acts as one component of a larger vision system whose tasks mainly consist of providing additional information for further processing steps, supporting appearance models by bridging detection failures, and forecasting the behavior by predicting future states.

1.1 Problem Statement

Given a short sequence of observations $\mathcal{Z}$ generated by an appearance model of a visual tracker, we are interested in estimating the state of a maneuvering object. In the following, systems where the discrete-time version of the motion or dynamical model can be formalized as follows are considered:

$$\mathcal{Y} = f_\theta\left(\mathcal{Z}^{0:k}, \mathcal{C}^{0:k}\right) + \epsilon, \tag{1.1}$$

The aim is to estimate the expected conditioned states of the object

$$\mathbb{E}[\mathcal{Y} | \mathcal{Z}^{0:k}, \mathcal{C}^{0:k}]. \tag{1.2}$$

Here, $\mathcal{Y}$ describes the states or state distributions of a tracked object, $\mathcal{C}$ describes additional contextual cues extracted from the observed image sequences and $\boldsymbol{\epsilon}$ describes an additional error term. In Bayesian filtering, models of this type are called state-space models or dynamical systems, whereas in deep learning, they are referred to as recurrent neural networks. This thesis includes discussions on both formulations and their connection by maximum likelihood inference. In order to effectively capture the dynamics of maneuvering objects and to reduce the amount of engineering, a comparable deep learning solution to adaptive recursive Bayesian filtering is introduced. The research questions in this context are answered along with the prototypical types of maneuvers, abrupt change of motions and random perturbations. The main application areas throughout this thesis are intelligent vehicles and automated surveillance systems.

1.2 Contributions

The starting point of this thesis are adaptive filters and their most common representative, the **interacting multiple-model** (IMM) filter [Blo88]. Based on a Bayesian formulation, the IMM filter is designed for capturing motion uncertainties and modeling complex object dynamics in situations where the object undergoes sudden changes. The IMM filter can be used to combine several dynamical models and offers a good compromise between performance and complexity. This thesis contributes to an improved design of a basic IMM filter as a module in a visual tracking pipeline by introducing both a state de-coupling and a re-coupling scheme as modifications:

- Firstly, when relying solely on visual cues, the benefit of a suggested de-coupling of the state estimate of an IMM filter is demonstrated [Bec16].
- Secondly, a state re-coupling scheme is introduced which helps to better deal with the corresponding observation uncertainties of such a tracking pipeline [Bec18a].

Although the IMM filter has some drawbacks, it is still a core element for many state-of-the-art applications. In order to reduce the amount of required engineering and to learn an improved dynamical model structure, a contribution of this thesis is the transfer of the IMM functionality into a comparable deep learning architecture. Since adaptive filters and in particular the IMM filter are designed to deal with maneuvering objects, in the following two major maneuver types are considered separately:

- Switching noise levels: The effectiveness of deep neural networks for predicting future pedestrian states is evaluated. A proposed network achieves state-of-the-art performance on publicly available datasets [Bec18c]. The results can be accessed on the *TrajNet* website (http://trajnet.stanford.edu/, last accessed 19.12.2019). The ranking of different predictors combines the final displacement error and the average displacement error for predicting the next 12 states of pedestrian trajectories pooled over the datasets. The proposed network is an RNN-encoder with a dense layer on top for projecting into the observation space. Although being simple at core, the network can achieve a performance comparable to more elaborated models in terms of considering more cues than solely position information.
- Switching behavior: The connections to the IMM filter are explored and an IMM filter surrogate is presented (RNN-IMM). Similar to an IMM filter solution, the presented RNN-IMM assigns a probability value to different dynamical modes and, based on them, generates a multi-modal distribution over future object states as output[Bec19b, Bec19a]. The switching behavior is thoroughly analyzed for prototypical, critical maneuver situations, such as a *bending in* maneuver of pedestrians. The presented RNN-IMM solution reduces not only the amount of explicit modeling of filter parameters but enables an improved maneuver onset and maneuver termination behavior.

In order to provide a learned reference trajectory for pooled object trajectory data, this thesis contributes by introducing an alignment network.

- Alignment network: The application of hard-coded normalization strategies on pooled trajectories shifts the variation along the trajectory. Hence, the arbitrarily chosen references hinder applying clustering approaches. The proposed network learns a freely adjustable prototype as a reference trajectory. Firstly, the resulting prototype reflects the minimum variance of the input trajectories, which allows deducing the dominating dynamical behavior. Secondly, with a fixed reference, the conditions for clustering approaches and out-of-distribution detections are improved.

Overall, this thesis is motivated by uniting the interconnected Bayesian and deep learning perspectives on maneuver prediction. In response to the research question on how to effectively capture changing object dynamics in image sequences, a transfer to an IMM filter comparable neural networks is introduced.

1.3 Outline

The thesis is structured as follows: Chapter 2 introduces the theoretical background in order to unite the functional views of deep learning and Bayesian filtering on object tracking. Furthermore, current state-of-the-art is surveyed for selected exemplary applications. In chapter 3, the problem of maneuvering object tracking is considered from the Bayesian filtering perspective, resulting in improved IMM filter designs. Chapter 4 presents an alternative deep learning-based solution in order to reduce the amount of hand-tuning of the filters and to provide an effective solution for the switching state problem. Conclusions are drawn in chapter 5.

2 Perspectives on State Estimation from Visual Observations

In this chapter, the perspectives of deep learning and recursive Bayesian filtering on (visual) object tracking are united. Based on the united functional view, the contributions of this thesis are positioned with respect to existing literature and to specific applications.

2.1 What is Visual Tracking?

Vision-based or visual tracking is defined as the process of using image observations and a predictive dynamical model to consistently estimate the state(s) of one or more object(s) over the discrete-time steps corresponding to video frames [Mag11]. Thereby recursive Bayesian filtering acts mostly as a top-down process for state estimation, which involves incorporating prior information about the scene or object to connect the object dynamics to physical systems [Bla03]. This tracking pipeline with top-down filtering is often referred to as detection-by-tracking [And08] and without top-down filtering as tracking-by-detection. A block diagram for a single object visual tracking pipeline is visualized in figure 2.1.

Vision-based tracking of a single object is formulated as the estimation of a time series $\mathcal{Z} = \{\mathbf{z}^k : k \in \mathbb{N}\}$ over a set of discrete-time instances k, based on the information $\mathcal{I} = \{\mathbf{I}^k : k \in \mathbb{N}\}$ from the set of images. The vector-valued time series $\mathcal{Z}$ is considered as the states of the object and is mainly referred to as the trajectory of the object. However, for recursive Bayesian filters or

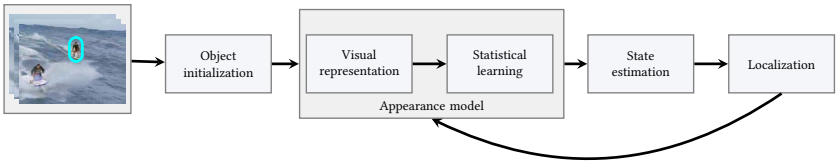

Figure 2.1: Block diagram of a visual tracking pipeline which shows the main components of a tracking cycle.

dynamical systems [1] the term state $\mathcal{X} = \{\mathbf{x}^k : k \in \mathbb{N}\}$ refers to a collection of variables such as position, velocity, orientation, which are indirectly observed through noisy observations.

Since the focus of this thesis is neither on building an appearance model for object detection nor the necessary feature extraction, but on the top-down state estimation, it is crucial to distinguish between the term states clearly. The term observation $\mathbf{z}^k$ will be used to describe the object representation (e.g., bounding box, centroid, blobs) in the image generated by an appearance model of a detector or a visual tracker. Thus, appearance modeling basically boils down to representing object pixel intensities. The resulting associated image region is the observation serving as input for the dynamical state estimation. Pedestrian detections in the form of enclosing bounding-boxes is an illustrative example of an observation space $\mathcal{Z}$. Observation, detection, and object state interchangeably refer to the shape approximation in the image in contrast to the dynamical state $\mathbf{x}^k$, which fully describes a dynamical system.

In figure 2.2, commonly used object representations for describing the location and an approximation of the object shape are depicted. For the goal of tracking an object in the 2D image space, the minimal form of $\mathbf{z}^k$ is the center position of the object in $\mathbf{I}^k$.

[1] The terms state-space models and dynamical systems are used interchangeably in this thesis. Whereas the term state-space models originates from probabilistic modeling, the term dynamical systems originates from signal processing. Bayesian filtering refers to the Bayesian way of formulating optimal filtering for dynamical systems.

Figure 2.2: Examples of object states for different visual tracking tasks.

Deep-tracking based approaches are mostly tailored to image processing tasks such as classification and detection, thus detection-by-tracking with a Bayesian filter is still very common [Kre17]. The tasks of the Bayesian filter within the overall pipeline are:

- The support of the appearance model to bridge detection failures or occlusion situations.
- Provide additional information for subsequent processing stages.
- Enhance the detection robustness.
- Estimate indirectly observables.
- Forecast the behavior of the object.

Within the pipeline, the tasks of the Bayesian filter can be explicitly associated with different types of inference problems: prediction, filtering, and smoothing. Because inference is a very general problem for machine learning models, the consideration of the filter functionality as an inference problem helps to unite the functional viewpoints. Furthermore, the types of required computations are neatly separated in order to reason from sequential data correctly [Moh15].

2.2 One Problem - Two Functional Views

Recursive Bayesian filtering refers to the Bayesian way of formulating the estimation of the hidden (dynamical) states using probability theory. Hence, the hidden dynamical states and the observations are assumed to be random variables. The dynamical state itself is represented by means of a **probability density function** (pdf)[1] $p(\mathbf{x}^k)$ at time step k. It is assumed that the **transition density**[2] $p(\mathbf{x}^{k+1}|\mathbf{x}^k)$ is the same for all time instances and behaves according to a known system transition function. This function is referred to as the dynamical model (see equation 1.1). Other commonly used terms include, among others, motion model, process model, and plant model. For recursive Bayesian filtering, the dynamical model can be written as

$$\mathbf{x}^{k+1} = f^k\left(\mathbf{x}^k, \mathbf{v}^k\right). \tag{2.1}$$

Here, $f^k(\cdot)$ is a non-linear function and $\mathbf{v}^k$ the process noise. In the remainder of this thesis, only discrete-time models are considered because the observations are a set of discrete-time instants. The time steps are related through $t^{k+1} = t^k + \Delta T$, where ΔT is the sampling time. The dynamical state $\mathbf{x}$ is assumed to be an unobserved Markov process, and $\mathbf{z}$ are the observations of a **hidden Markov model** (HMM). The observation or measurement model maps the hidden dynamical state into the observation space and is given by

$$\mathbf{z}^k = h^k_{obs}\left(\mathbf{x}^k, \mathbf{w}^k\right), \tag{2.2}$$

with the non-linear observation function $h^k_{obs}(\cdot)$ and observation noise $\mathbf{w}^k$. A graphical model, which expresses the conditional dependence structure of such an HMM, is depicted in figure 2.3 (see for example [Kol09]). The structure of the model complies with a directed acyclic graph representing the factorization of the joint probability.

[1] In case of a discrete state-space **probability mass function** (pmf) $P(\mathbf{x}^k)$.
[2] In case of a discrete state-space **transition probability** $P(\mathbf{x}^{k+1}|\mathbf{x}^k)$.

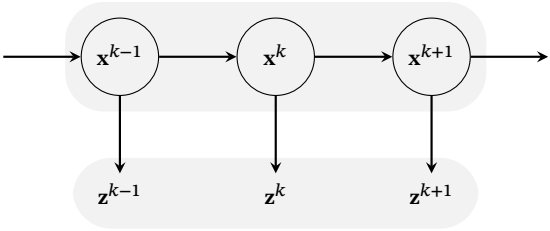

Figure 2.3: A graphical model specifying the conditional relations for a dynamical system.

As aforementioned, the conditional density $p(\mathbf{x}^{k+1}|\mathbf{x}^k)$, which depends on the dynamical model, is assumed to be stationary. This is equivalent to assuming that the parameters of the transition function are shared across time steps [Moh15]. Thus, it is possible to directly connect to **recurrent neural networks** (RNNs) [Goo16, Rum88] and the loss function of RNNs using maximum likelihood estimation. Under the Markov assumption, the probability of an observed sequence $\mathcal{Z}$ according to a **dynamical system** (DS) as depicted in figure 2.3 can be calculated by marginalizing over $\mathbf{x}^k$:

$$\begin{aligned} p(\mathbf{z}^0,\dots,\mathbf{z}^K) &= \prod_k \int p(\mathbf{z}^k,\mathbf{x}^k)\,\mathrm{d}\mathbf{x}^k \text{ with} \\ p(\mathbf{z}^k,\mathbf{x}^k) &= p(\mathbf{z}^k|\mathbf{x}^k)p(\mathbf{x}^k|\mathbf{x}^{k-1}) \end{aligned} \tag{2.3}$$

Using the negative log-likelihood, the following loss function can be obtained:

$$\mathcal{L}(\Theta)_{DS} = -\sum_k \log \int p(\mathbf{x}^k|\mathbf{x}^{k-1})p(\mathbf{z}^k|\mathbf{x}^k)\,\mathrm{d}\mathbf{x}^k \tag{2.4}$$

For deterministic transition dynamics,

$$p_\Theta(\mathbf{x}^k|\mathbf{x}^{k-1}) = \delta(\mathbf{x}^k - f_\Theta(\mathbf{x}^{k-1},\mathbf{z}^{k-1})), \tag{2.5}$$

the loss function can be reformulated to

$$\begin{aligned}\mathcal{L}(\Theta)_{DS} &= -\sum_k \log p(\mathbf{z}^k | f_\Theta(\mathbf{x}^{k-1}, \mathbf{z}^{k-1})) \\ &= -\sum_k \log p(\mathbf{z}^k | f_\Theta(\mathbf{x}^{k-1})).\end{aligned} \tag{2.6}$$

Next, the loss function is recovered from the perspective of RNNs. According to equation 2.4, the goal is to capture the probability of the observed sequence $\mathcal{Z}$. RNNs are extensions of multi-layer feed-forward networks, where hidden units $\mathcal{H} = \{\mathbf{h}^k : k \in \mathbb{N}\}$ are used to encode an internal hidden state space [Goo16]. In extension to multi-layer networks, the parameters are shared across different parts of a model. Here, the parameter of the transition function are shared across time steps, resulting in a neural network where the activation of the hidden layers are fed back into the network along with the input. Figure 2.4 depicts the unfolded computational graph of an RNN, where the hidden state sequence is used to compute the output vector sequence $\mathcal{O} = \{\mathbf{o}^k : k \in \mathbb{N}\}$.

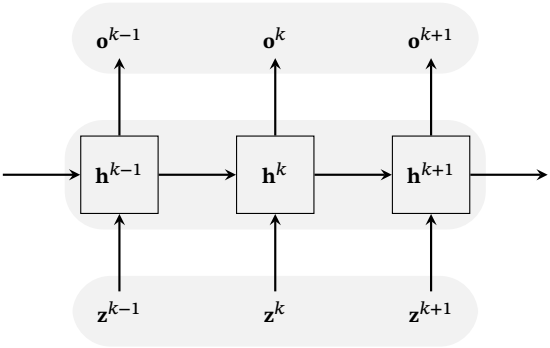

Figure 2.4: Recurrent neural network seen as an unfolded computational graph. Each node is associated with a particular time instance.

The unfolded model structure corresponds, similarly to recursive Bayesian filtering, to a directed acyclic computational graph. Thus, the recurrent network processes information by incorporating it into the state $\mathbf{h}$ that is passed forward through time by the transition function. The hidden state for one time step is given by

$$\mathbf{h}^{k+1} = f_\Theta\left(\mathbf{h}^k, \mathbf{z}^{k+1}\right). \tag{2.7}$$

For a basic RNN [Elm90] the transition function is given by

$$\mathbf{h}^{k+1} = \phi\left(\mathbf{W}_{hh}\mathbf{h}^k + \mathbf{W}_{hz}\mathbf{z}^{k+1} + \mathbf{b}_h\right), \tag{2.8}$$

where $\mathbf{W}_{hh}$ and $\mathbf{W}_{hz}$ represents the weights, $\mathbf{b}_h$ the biases of a recurrent layer, and $\phi(.)$ an activation function. Based on ideas of graph unrolling and parameter sharing, a wide variety of recurrent neural networks can be designed [Goo16]. For the moment, we stick with an RNN as depicted in figure 2.4 that generates an output at each time step and uses a hidden-to-hidden recurrent connection as described above. The depicted RNN does not specify what form the output and loss function take. Thus, the output $\mathbf{o}^k$ can be used to parameterize a predictive distribution $p(\mathbf{z}^{k+1}|\mathbf{o}^k)$ over possible next observations $\mathbf{z}^{k+1}$. In order to match $\mathbf{z}$, the form of $p(\mathbf{z}^{k+1}|\mathbf{o}^k)$ must be chosen carefully. The problem of finding a good predictive distribution can be very challenging and is usually referred to as probability density modeling [Gra13a]. Given a hidden state, the output is computed as follows

$$\mathbf{o}^k = o\left(\mathbf{W}_{ho}\mathbf{h}^k + \mathbf{b}_o\right), \tag{2.9}$$

where $o(\cdot)$ is the output layer function, $\mathbf{W}_{ho}$ denotes a weight matrix and $\mathbf{b}_o$ denotes a bias vector. The complete network defines a function, parameterized by the weight matrices, from observations $\mathbf{z}^{0:k}$ to the output vector $\mathbf{o}^k$. Equation 2.7 can be considered as the RNN equivalent of the dynamical model, and equation 2.9 can be considered as the RNN equivalent of the observation model. The probability of an observed sequence $\mathcal{Z}$ as estimated by an RNN is

given by

$$p(\mathbf{z}^0, \dots, \mathbf{z}^K) = \prod_k p(\mathbf{z}^{k+1}|\mathbf{o}^k). \tag{2.10}$$

The corresponding loss function of an RNNs using maximum likelihood estimation can be defined as:

$$\mathcal{L}(\Theta)_{RNN} = -\sum_k \log p(\mathbf{z}^{k+1}|\mathbf{o}^k), \text{ or rather} \tag{2.11}$$

$$= -\sum_k \log p(\mathbf{z}^{k+1}|f_\Theta\left(\mathbf{h}^{k-1}, \mathbf{z}^k\right)). \tag{2.12}$$

Due to the deterministic nature of RNNs, the computation of the predictive distributions is realized by the feed-forward operations in the unfolded network. By applying **backpropagation through time** (BPTT) [Wil95] to the computational graph, the partial derivatives of the loss with respect to the network weights can efficiently be calculated, and the network can be trained using stochastic gradient descent.

When comparing the loss functions of equation 2.12 and 2.6, it becomes evident that the RNN loss corresponds to maximum likelihood estimation with deterministic dynamics. According to Bayesian filtering, the result for the associated inference problems are given in form of a conditional probability density that represents the dynamical state estimate [Hub15]. The estimation tasks depend on the relation between the time steps k and K. If $k < K$, the estimation problem is referred to as *prediction* (inferring the future), for $k = K$ the estimation is referred to as *filtering*, *update*, or *correction* respectively (inferring the present), and if $k > K$, it is referred to as *smoothing* (inferring the past). Prediction and filtering are typically performed on-line, while smoothing is an off-line estimation task, as it improves past state estimates given additional information. For Bayesian filters, the conditional densities are calculated recursively under the assumption that the dynamical state is a Markov process. For the tracking pipeline described in section 2.1 and for

many other practical applications, prediction and filtering are performed alternatingly, which is commonly referred to as *prediction-update* cycle. Based on the above drawn connections between the Bayesian perspective and the RNN perspective, for both on-line estimation tasks of recursive Bayesian filters, there exists an RNN counterpart, where *prediction* and *update* are realized by feed-forward operations in the unfolded network. For applying Bayesian filtering, strict assumptions such as Gaussian transitions are required to solve the inference problem, but those assumptions are commonly violated in a real environment. Before the seminal Kalman filter [Kal60] is introduced as a basic dynamical state estimator, this thesis is positioned with respect to the existing literature for two selected tasks, where the role of a top-down state estimator as part of a vision-based tracking is a crucial component. While extracting the observation from images is specific to computer vision, inference is very general, and the field of machine learning and pattern recognition is entered. In order to narrow down the large number of existing approaches originating from different communities, these approaches are categorized with respect to the applied motion model, the level of contextual information used, and the time horizon under consideration. The following discussion uses mainly the Bayesian perspective for positioning the contributions compared to related work.

2.3 Related Work

The two selected tasks, in which higher-level processing strongly relies on the state estimator, are *path prediction* and *intention prediction*. Whereby *path prediction* is mainly tackled as a pure *prediction* problem for *intention prediction*, both *prediction* and *filtering* is mostly done jointly (see for example [Gav99]).

2.3.1 Path Prediction

In tasks such as *path prediction*, the term *agent* often denotes the dynamic objects of interest such as robots, pedestrians, cyclists, cars, or other human-driven vehicles. The target agent is the dynamic object for which the motion

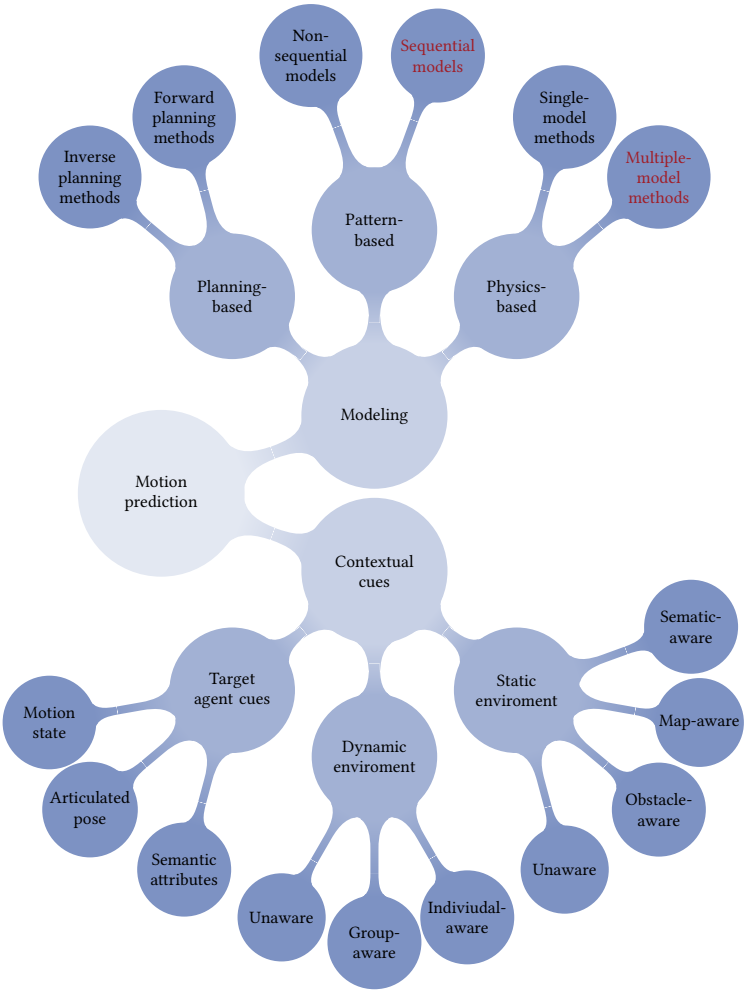

Figure 2.5: Overview of the taxonomy of categories according to Rudenko et al. [Rud19]. The categorization of this thesis within the taxonomy is highlighted in red.

prediction is done and corresponds to our tracked object. The term *path* is here restrictively used for a sequence of positions, and the term *trajectory* can include additional information for describing the movement of the object. In

our case, the term trajectory corresponds to a sequence of object states (see section 2.1) [1] However, the focus here is on *path prediction*, but the prediction of video frames, actions, articulated motion, or human activities often rely on the same motion prediction methods. As explained, there is a cross-disciplinary interest and a fast-growing body of work for motion prediction. In order to categorize the different prediction methods, we built on the taxonomy introduced by Rudenko et al. [Rud19]. In accordance with this taxonomy, motion prediction is categorized with respect to the modeling approach and the type of contextual cues. In figure 2.5, the categories of the taxonomy introduced by Rudenko et al. are visualized.

Categorization from other related surveys may differ slightly, but are similar at their core. In order to name a few, there are surveys from application domains such as service robots [Kru13, Las17], intelligent vehicles [Ras19, Rid18, Bro16, Lef14], and computer vision [Hir18, Mur17, Mor08].

Most relevant for this thesis is the survey of Hirakawa et al. [Hir18]. They survey path prediction methods for vision-based systems, where all the considered methods are realized on top of computer vision tasks, such as pedestrian detection. This corresponds precisely to our distinction between appearance modeling to generate observations as input data and the top-down state estimator. Hirakawa et al. categorize motion modeling approaches mainly into Bayesian models, energy minimization methods, deep learning methods, and inverse reinforcement learning methods. In addition, the approaches are categorized depending on whether they explicitly use object features or environmental features extracted from a video. Rasouli and Kotsos [Ras19] survey pedestrian behavior in the application domain of intelligent vehicles and use the terms pedestrian factors and environmental factors to distinguish with respect to the awareness of a specific factor. With the taxonomy of Rudenko et al., this distinction is addressed by contextual cues. The categories utilized by Kruse et al. [Kru13], Lasota et al. [Las17] and Lefèvre et al. [Lef14] are also based on motion modeling and correspondingly included in the taxonomy.

[1] In robotics, the term *path* is used for describing a space curve without a notion of time and the term *trajectory* is used for a path with a notion of time.

The taxonomy depicted in figure 2.5 enables to distinguish along the different modeling approaches and along an increasing level of contextual awareness. Using the motion modeling approach as a classification criterion, prediction approaches are divided in *physics*-based methods, *pattern*-based methods and *planning*-based methods. The second criterion asks what contextual cues are exploited, leading to a classification between *target agent* cues, *dynamic environment* cues, and *static environment* cues.

In addition to this taxonomy, we distinguish if some contextual cues are used in an additional processing step in order to associate the object's dynamical state with the physical world. From the perspective of a Bayesian filter with a dynamical model, it is important if a reasonable observation model can be applied. In particular, *path prediction* is mostly done on ground level, which implicitly requires additional assumptions about the environment or additional sensors (LIDAR, stereo camera system) or approaches like structure-from-motion (SfM) [Sze10] to reconstruct a $3D$ scene. For example, an intelligent vehicle is accompanied by many additional sensors, which allow the actual prediction of dynamic objects being done in an ego-motion compensated vehicle centered coordination system. Thus, even when the environmental cues are not used as input for the motion prediction itself, the overall vision-based system is aware of its environment. Thus, the system implicitly relies on more contextual cues. However, there exist several scenarios where this mapping is unknown, includes substantially higher expense, or is an overall unsolved problem. An example is general object tracking. In such a case, the object is directly tracked in image space on randomly selected videos. In several domains, this implicit knowledge of the environment is presumed but not always given. An example from the domain of visual surveillance is video recordings without extrinsic or intrinsic calibration of the cameras. The condition of being able to rely on mapping to the physical world or not is referred to as explicit or implicit contextual cues in the remainder of this thesis.

In addition to the categories proposed by Rudenko et al., the relevant time horizon helps to further differentiate between prediction methods. Motion prediction methods can be roughly categorized into short-term prediction

with relevant time horizons of $0.5 - 2$ seconds, and into long-term prediction with relevant time horizons of $5 - 20$ seconds. In figure 2.6, a mapping between preferred motion modeling approaches and the prediction time horizon is visualized.

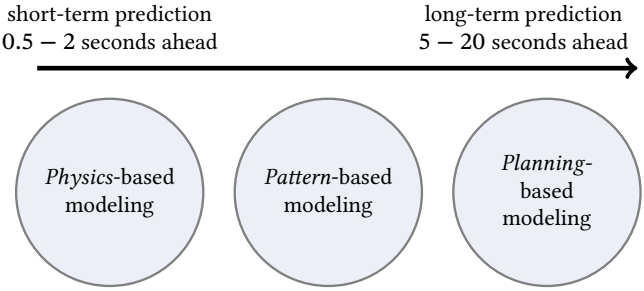

Figure 2.6: Categorization of the relevant time horizon for different motion prediction approaches.

Depending on an increasing time horizon, a shift in the preferred motion modeling category is visible. Due to the context of maneuvering objects, it is clear that a quick reaction to a change in motion is required, and only short-term prediction is considered. Nevertheless, the category of *planning*-based methods is kept for a better overall view on motion prediction.

Physics-based methods: *Physics*-based methods define an explicit transition function, the dynamical model, which is based on Newton's law of motion as part of a recursive Bayesian filter. Individual dynamical models differ according to the type of motion they describe. Different motion types include maneuvering or non-maneuvering motions, the complexity of object dynamics, and the noise model.

As already described, prediction is done by inferring from observed cues. In the prediction taxonomy, these models are subdivided into *single-model* approaches and *multiple-model* approaches that involve several modes of dynamics. In situations where the object's behavior changes abruptly, *multiple-model* approaches are utilized. In order to model the motion of maneuvering objects, a fusion of different prototypical motion models is done. A more detailed description of different fusion strategies and the technical background of *multiple-model* approaches is given in chapter 3. In short, *multiple-model* methods include an adaptive set of dynamical models and a fusion strategy to select individual models [Poo17, Koo16, Koo19, Sch13, Aga12]. Examples of *single-model* methods include the approaches of Yamaguchi et al. [Yam11], Pelligrini et al. [Pel09], Zernetsch et al. [Zer16], and Elganar et al. [Eln01].

Physics-based methods are commonly considered for short-term predictions. In contrast to *pattern*-based methods, they can readily be applied to unknown environments without the need for training data. They provide fast and efficient inference including explicit handling of prediction uncertainty. Drawbacks are the limited expressive power and the large amount of engineering required to design a filter [Bar02]. Physics-based approaches are due to their generalization ability and their fast inference still the most popular approaches for applications with a short prediction time horizon, such as collision avoidance [Rud19].

Pattern-based methods: Instead of using an explicit motion model, *pattern*-based methods learn generalized transitions and trajectories from training data. This is done by using different function approximators such as HMM or neural networks. Depending on the type of function approximator, two main categories are distinguished. *Sequential* methods typically learn conditional models under the assumption that the dynamical state is conditionally dependent on the history of past states. As shown in section 2.2, this function approximator can be realized with HMMs and RNNs. In most cases, the function approximator is realized as a regression problem. For neural networks, the corresponding loss function is that of a feed-forward network, with an appropriate distance function for the path being predicted, such as the squared loss. Under certain assumptions, such as discrete or Gaussian

transitions with random dynamical states, HMM and Kalman filters, respectively, allow learning a function approximator. More recent approaches use variational inference or particle **Markov-chain-Monte-Carlo** (MCMC) for large-scale dynamical systems [Bar12]. In order to predict a sequence of state transitions, consecutive one-step predictions are made to concatenation into paths of arbitrary length.

Examples of *sequential pattern*-based methods are the approaches of Vemula et al. [Vem18], Keller et al. [Kel14], Goldhammer et al. [Gol14], Alahi et al. [Ale17, Ala16], Kucner et al. [Kuc17], Zhang et al. [Zha19], and Xue et al. [Xue19]. A more elaborate description of the technical background for *sequential pattern*-based methods will be given in chapter 4.

Non-sequential methods aim to learn a set of motion patterns or directly model the distribution over full trajectories without temporal factorization of the dynamics. Commonly, non-sequential approaches are based on clustering in order to identify sets of long-term motion patters in the observed trajectories. Clustering is an unsupervised machine learning technique for identifying structure in unlabeled data [Bis06]. For generating useful clusters, the clustering approaches address issues such as the definition of a distance or similarity measure, update methodology, and cluster validation [Mor08]. In order to name a few *non-sequential* approaches which intend to model the distribution of object trajectories, there are the approaches from Xiao et al. [Xia15], Luber et al. [Lub12], and Trautman et al. [Tra10]

In summary, *pattern*-based methods can deal with comparatively large prediction horizons and are suited for scenarios with complex unknown dynamics. On the downside, this requires training samples from specific scenes that can not easily be pooled together. A further issue is the generalization capability. *Pattern*-based methods tend to be used in non-safety critical applications in a spatially constrained environment [Rud19]. In the scope of this thesis, some of the standard recursive filter functionality is transferred to such a *pattern*-based learning approach and particularly used in a time-critical scenario.

Planning-based methods: As unique characteristics, *planning*-based methods assume a criterion for optimal motion in the environment. By solving

a sequential decision-making problem, the optimal path of an object is computed. Most approaches differ in the type of objective functions that minimizes the total cost of a sequence of actions or rather motions. Thus, *planning*-based methods explicitly reason about the goal of a long-term motion and compute policies or path hypotheses to enable to reach those goals. In order to estimate an optimal path, these methods rely on Markov decision processes (MDP) [Mur12], reinforcement learning, rapidly-exploring random trees (RRT) [Kar11], *potential field* or *shortest-path* algorithm such as Dijkstra and A* (see for example Thrun et al. [Thr05]). Using the motion prediction taxonomy, planning-based approaches can be classified into two subcategories of *forward planning* methods and *inverse planning* methods. The distinction depends on the choice of the reward function. *Forward planning* methods rely on a pre-defined reward function and *inverse planning* methods aim to learn the reward function by applying statistical learning techniques on the trajectory data.

Examples from the category of *forward planning* methods include the approaches of Rudenko et al. [Rud17], Vasquez [Vas16], Rösmann [Rös17], Karasev et al. [Kar16a], and from the category of *inverse planning* methods the approaches of Kitani et al. [Kit12], Rehder et al [Reh18], Ziebart et al. [Zie09] are included.

In summary, *planning*-based approaches are considered if it is possible to define goals for the objects explicitly and a model or map of the environment is available. If these conditions are met, the *planning*-based approaches tend to generate better long-term predictions than the *physics*-based techniques and tend to generalize better than the *pattern*-based to unseen environments. However, in dynamic environments, re-computation of the reward function is required and mostly, this is time-consuming. Thus, for short-term prediction and fast changing object dynamics, these approaches are not well-suited. The assets and drawbacks of the introduced motion modeling approaches are summarized in table 2.1.

Contextual Cues: Besides using the modeling approach to categorize the prediction approach, the amount of exploited contextual cues help to further distinguish between single approaches. The contextual cues for describing

Table 2.1: Summary of the assets and drawbacks of the different motion modeling approaches.

Motion modeling	**Assets**	**Drawbacks**
Physics-based approaches	+ Simple, efficient, work well under mild conditions in particular for short-term prediction horizons. + Explainable, data efficient and generalize well with respect to unseen environments. + Possible to incorporate dynamic contextual cues to models but lead to complex algorithms.	– No reasoning over global environment. – Capture only pre-defined motion dynamics. – Large amount of engineering required.
Pattern-based approaches	+ Learning from actual motion of objects. + Reduced modeling required. + Ability to capture complex dynamics. + Long-term predictions. + Capture theoretically all contextual cues present in the training data. + Fast inference.	– Require large amount of training data. – Limited generalization to new environments. – Low explainability.
Planning-based approaches	+ Generalization to new environments. + Explicitly reasoning on executed actions intended on goals and map awareness. + Long-term predictions.	– Mandatory pre-requirement of goals (e.g. as semantic annotations). – Re-computation of the reward function required in dynamic environments. – Strong dependency on the discretization of action and state-spaces. – Re-computation of the reward function is time consuming.

the overall contextual awareness of an approach are briefly explained here. In the application domain of intelligent vehicles, Rasouli et al. [Ras19] presented a very detailed survey of factors of pedestrian behavior such as demographics and environmental conditions. However, their categorization in pedestrian factors and environmental factors, include major and sub-factors which can be directly mapped to the categories of Rudenko et al. According to Rudenko et al. and using their terminology, the categorization of the prediction problem along the contextual cues is done based on three criteria. These classification criteria are defined as the contextual cues from the object itself

(*object* or *target agent* cues), cues from a dynamic environment or cues from a static environment.

Although it is sufficient to not further differentiate the motion state cues as part of the object cues in respect of a taxonomy for prediction, it is crucial for keeping the strict separation of unobserved and observed dynamical states for the scope of this thesis. Instead of combined contextual cues as input vector representing the observed environment for a general formulation of the prediction function, the provided input by the tracking system is always considered separately.

Instead of using

$$\mathcal{Y} = f_\theta \left(\mathcal{C}^{0:k} \right) + \boldsymbol{\epsilon} \tag{2.13}$$

to formalize the prediction problem, the following equation is used (see equation 1.1)

$$\mathcal{Y} = f_\theta \left(\mathcal{Z}^{0:k}, \mathcal{C}^{0:k} \right) + \boldsymbol{\epsilon},$$

where $\mathcal{Y}$ describes for a *path prediction* problem the future locations (or distribution over the locations). As before, $\mathcal{Z}$ are the observations generated by the tracking system (appearance model of the visual tracker), $\mathcal{C}$ are additional contextual cues extracted from the observed image sequences, and $\boldsymbol{\epsilon}$ describes an additional error term.

Thus, the scope of this thesis is to replace the f_θ of a dynamical system in combination with a recursive Bayesian filter with a learning-based solution. By utilizing deep learning-based approaches, it is clear that the proposed solutions fall not univocally into a single class of taxonomy of Rudenko et al. The starting point is *physics*-based *multiple-model* approaches, which are still the dominant approaches to capture maneuvers.

With respect to the object dynamics, every object is considered separately. Thus these approaches are unaware of other objects. For tracking systems with Bayesian filters, this aspect is tackled by data-association solutions like

multi-hypotheses tracking. As mentioned earlier, for detection-by-tracking the contextual cues are often implicitly used to allow the mapping to the physical system under specific assumptions, but the scene context is not included in the modeling approach for the prediction. Thus, basic *physics*-based *multiple-model* are unaware of other static environment cues. However, for all the modeling approaches exist context-aware predictors, but due to the fact that *learning*-based approaches are best suited to integrate this kind of information, we will indicate for the appropriate situations how to handle additional context cues, but keep the focus on providing stable solutions for the case the tracked object is unaware of additional clues.

2.3.2 Intention Prediction

Although many approaches can relatively reliably predict the location of objects a few seconds ahead, they still struggle to predict when the object will stop. Towards this end, *intention prediction* is selected as an exemplary task to evaluate different aspects of the proposed solution with respect to the switching dynamics of objects. Intention prediction is an expression mainly used in the domain of intelligent vehicles as part of overall pedestrian behavior analysis of vision-based active safety systems. The pedestrian intention can be estimated jointly with *path prediction*, as proposed by [Gav99], but also as pure classification task of a pedestrian action. Due to this close relation between intention and path prediction, approaches for *intention prediction* can be categorized with the same taxonomy as before. Furthermore, we look at the problem from the Bayesian filter perspective and retain accordingly the modeling basis relying on the observed trajectory.

However, the estimation of the pedestrians' intention with respect to their impending motion can basically be tackled with all of the approaches introduced in section 2.3.1 and mixtures of them. An essential difference is the short time-window for the prediction and the decision to be made due to the speed of the vehicle. Since *physic*-based methods are efficient for short prediction horizons and generalize well to unseen environments, the number

of approaches in recent literature for *intention prediction* relying on *physics*-based approaches is significantly larger than for *path* prediction. The *multiple-model* approaches help to better deal with motion model uncertainties. The integration of context-awareness for the predictors lead to complex learning algorithm. For inference, the combination with Bayesian filter is kept.

In reviews on *intention prediction* or pedestrian behavior prediction [Rid18, Ras19], the *prediction-update* cycle of recursive filter is used to categorize all approaches originated from tracking as *dynamics*-based prediction. Thereby, the distinction between a *physics*-based and *pattern*-based approaches is lost, but the Kalman filter, independent of a learned or selected physical motion model, can be set as baseline approach. A large variety of physics-based models describing the motion of dynamic objects in ground, marine, airborne object tracking, is presented in the work of Li et al. [Li03]. Popular examples of motion models include the **constant velocity** (CV) model, **constant acceleration** (CA) model, and **constant turn** (CT) model. Since the publication of the seminal Kalman article [Kal60], as special case of Bayesian filtering, many extensions have been proposed. For example non-linear extensions, such as the **extended Kalman filter** (EKF), the **unscented Kalman filter** (UKF), or non-Gaussian extensions, such as **particle filter** (PF) [Bar02, Gri18].

In addition to the before mentioned *physics*-based *single-model* methods, the following approaches use, inter alia, a Kalman filter approach for prediction of pedestrian positions. Bertozzi et al. [Ber04] (EKF), Meuter et al. [Meu08] (UKF), and Møgelmose [Møg15] (PF) use Kalman filtering with a CV model. In the work of Binelli et al. [Bin05] and Elnagar et al. [Eln01], a Kalman filter is combined with a CA model. For tracking other road users, such as bikes and vehicles, variants of the CT model are often utilized (see for examples [Bar08, Bat09]). Zernetsch et al. [Zer16] incorporated additional *object* cues in form of the resistance forces from inclination and rolling to extend the cyclist dynamical model. In [Sch13], Schneider and Gavrila conducted a comparative study on using Kalman filters with different dynamical models for pedestrian *path prediction*. An alternative approach relying also on an HMM, but with discrete hidden states representing intention classes, was introduced in the

work of Wakim et al. [Wak04]. They classify the four pedestrian behaviors of *standing*, *walking*, *jogging*, and *running*.

By combining such intention class prediction jointly with *path prediction*, we end up with *multiple-model* approaches. For longer time-horizons the intention of the object motion is dominated by its goals. This again illustrates the gentle transitions between the *modeling* methods. For example the approaches of Kitani et al. [Kit12], Tamura et al. [Tam12], and Ziebart [Zie09] propose algorithms to learn a dynamical model yielding goal-directed behavior of pedestrians using maximum entropy inverse optimal control. Under the assumption that pedestrians make near-optimal decisions with stochastic policies, probability distributions over trajectories are predicted.

The primary approach of *multiple-model* methods are referred to as multiple-model methods and hybrid dynamical state methods [Hof04], that augment the discrete motion or intention state with the continuous dynamical state. Following the description of Li and Jilkov [Li10] of *multiple-model* methods, they consist of the following elements. Firstly, an adaptive dynamical model set. Secondly, methods to deal with discrete value uncertainties, such as a Markov or a semi-Markov assumption. Thirdly, a recursive estimation scheme to deal with the continuous dynamical states conditioned on the dynamical model. Fourthly, a strategy to estimate the overall best by fusion or selection of individual filters. The combination of an HMM and a (linear) dynamical system is called **jump Markov linear system** (JMLS) [Mur12]. Other common expressions include **switching state-space model** (SSSM) or **switching linear dynamical system** (SLDS). For predicting cyclist intentions, Pool et al. [Poo17] presented a mixture of five linear dynamical models and included the static environmental cues by excluding single motion prediction not complying with the road topology.

Instead of a JMLS, Karasev et al. [Kar16a] rely on a jump-Markov decision process [Mur12] to model pedestrian motion. The pedestrian dynamics is described with a soft Markov decision process, and the pedestrian goals are the hidden discrete states. Environmental cues are included with engineered reward function terms for surface types (e.g., sidewalk, crosswalk, road, grass).

As stated before, the **interacting multiple-model** (IMM) filter is the most common inference technique applied for tracking problems [Maz98] with maneuvering objects. For example [Lin16] used an IMM filter to track pedestrians in the application domain of service robots. Madrigal et al. [Mad13] proposed an IMM filter solution in a surveillance scenario. In [Kae04], Kaempchen et al. tracked maneuvering vehicles with an IMM filter. In the particular context of *intention prediction* of pedestrians from a vehicle perspective, Schneider and Gavrila [Sch13] proposed an IMM filter with several basic motion models. Köhler et al. combined an IMM filter for pedestrian tracking with a **support-vector-machine** (SVM) to classify the intention to cross based on motion contour image [Bob96] in a surveillance scenario.

In order to include *contextual* cues several approaches added a **dynamic Bayesian network** (DBN) [Kol09] or **conditional random fields** (CRFs) [Laf01] on top of a SLDS (see for example [Has15a, Has15b, Koo19, Bon14, Koo14, Sch15]). Specifically, this means that for inference the IMM filter is applied to predict future object positions and the additional hidden state influences the transition probability between single dynamical models. Hashimoto et al. [Has15b] used an DBN to consider the behavior of other pedestrians. In [Has15a], they included the information of pedestrians being part of a group. In accordance with Quintero et al. [Qui15] and Keller et al. [Kel11], Hashimoto et al. reported that it is harder to recognize the decision of a pedestrian to stop than the decision to cross a street.

Kooij et al. [Koo14] presented a DBN to model the latent factors of head poses extracted by a head pose detector to account for inattentive pedestrians. Together with spatial cues captured by the distance of the pedestrian to the road curbside, the change in pedestrian dynamics is controlled. In [Sch15], Schulz and Stiefelhagen proposed an intention recognition system based on latent-dynamic CRF to integrate the pedestrian head orientation for controlling the motion model switches. For the estimation of vehicle trajectories with an IMM filter, Kuhn et al. [Kuh15] presented a DBN to embed the context of possible routes by a pre-defined environment geometry.

There is a recognizable trend of integrating more and more *contextual* cues of possible causes of intention changes to better anticipate instead of reacting to changes in dynamics. This, however, does not change the fact that quick reaction to a change in dynamics is crucial for the overall system. Even though contextual cues can for example be incorporated with DBNs or other modeling schemes, the current predominant machine learning paradigms are neural networks. In particular, RNNs are the standard approach for modeling sequential data. As explained in section 2.3.1, there exist an increasing amount of RNN-based approaches for *path prediction*. Our goal is to preserve benefits from traditional *multiple-model* methods to deal with maneuvering object, but get rid of the tedious tuning of the filters. Nevertheless, the before listed approaches [Ale17, Ala16, Zha19, Xue19] in the category of *sequential pattern*-based models are closely related to this work. The technical background of the RNN-based variants is explained in chapter 4.

At this point, we limit ourselves to several approaches applied in an *intention prediction* setting and refer to surveys such as [Rud19, Rid18, Hir18, Ras19, Hir18] for further reading. Not relying on the neural networks nor a *multiple-model* approach, but categorized as *pattern-based* methods are the works from [Qui15] and Keller et al.[Kel11]. In [Kel11] probabilistic hierarchical trajectory matching is used to match an observed pedestrian track with a database of tracklets or rather trajectory sections. Extrapolated future location from the best fitting sections are then combined with dynamic features extracted using dense optical flow inside the pedestrian bounding boxes, or as in [Qui15] extracted from full-body articulated poses. In both works, these body motion dynamics are learned using **Gaussian processes with dynamic model** (GPDM) with an HMM to switch between the behavior classes of *crossing* and *stopping*. Quintero et al. [Qui19] included the behavior classes *starting* and *standing*. A pure intention classification approach was proposed in the work of Völz et al. [Völ15] using a SVM to infer pedestrian crossing from extracted tracks from LIDAR sensor. Alternatively, they proposed an regression forest [Völ16], and later presented an RNN-based solution [Völ19]. Goldhammer et al. [Gol15, Gol14] introduced a neural network-based approach. Trajectories from a pedestrian head tracking system in static traffic are approximated

with a least-spare polynomial fit for a fixed input window. The resulting polynomial coefficients are used as input for an **multi-layer perceptron** (MLP) [Goo16] to predict future paths.

Representatively the RNN-based approach of [Sal18] is mentioned due to the fact that they use an RNN regression model to predict pedestrian path without the maneuver context, but distinguish for their analysis between corresponding trajectories classes such as *crossing*. In the context of vehicle maneuvers, such as *lane changing*, Deo and Trivedi [Deo18] presented an RNN-based model to compute maneuver-dependent vehicle trajectories. Together with our proposed RNN-based pedestrian *path prediction* network [Bec18c], this model serves as basis for the presented RNN-based IMM filter surrogate [Bec19b, Bec19a].

2.4 Summary

In this chapter, the contributions of this thesis were positioned with respect to related literature for the selected application of *path* and *intention prediction*. The focus of the thesis is state estimation of maneuvering objects as part of a visual tracking pipeline realized as detection-by-tracking approach. For a high level of abstraction, the processing pipeline contains the following elements. The object observations provided by an appearance model based on extracted image feature, describing the object in image space. These observations serve as input for a Bayesian filter or the proposed RNN-based alternatives. We shift from a *physic*-based modeling to a *pattern*-based modeling of the dynamics. Both modeling approaches predict a parametric distribution over the object states and jointly capture maneuver probabilities for subsequent processing stages.

3 The Bayesian Perspective

In this chapter, Bayesian filtering solutions including the IMM filter for dealing with maneuvering objects are explored. After an introduction of the technical background, design modifications compared to a basic IMM filter are introduced. Some of the results presented in this chapter have been published in our previous work [Bec16, Bec18a].

3.1 Background

As described in section 2.2, dynamical state estimation, also known as Bayesian filtering, is a general probabilistic approach for recursively estimating an unknown probability density function over time using incoming observations and a dynamical model. In order to calculate these densities in a recursive fashion, the assumption of the dynamical state $\mathbf{x}^k$ being a Markov process is implied. The *prediction-update* cycles consist of alternating estimates of the conditional probability density from an initial state density $p(\mathbf{x}^0)$ at time step $k = 0$.

In the *prediction* step, the predictive distribution of the dynamical state is computed. Then, the observation model $h_{obs}^k\left(\mathbf{x}^k, \mathbf{w}^k\right)$ allows to predict the expected observation. Thus, given the conditional density $p^+(\mathbf{x}^k) \triangleq p(\mathbf{x}^k|\tilde{\mathbf{z}}^{0:k})$ the density of the predicted state at time step $k+1$ is calculated according to the *Chapman-Kolmogorov* equation [Hub15]

$$p^-(\mathbf{x}^{k+1}) \triangleq p(\mathbf{x}^{k+1}|\tilde{\mathbf{z}}^{0:k}) = \int p(\mathbf{x}^{k+1}|\mathbf{x}^k)p^+(\mathbf{x}^k)\,\mathrm{d}\mathbf{x}^k. \tag{3.1}$$

Here, $\tilde{\mathbf{z}}^k$ is an actual observation, a realization of $\mathbf{z}^k$. $p(\mathbf{x}^{k+1}|\mathbf{x}^k)$ is the transition density that depends on the dynamical model $\mathbf{x}^{k+1} = f^k\left(\mathbf{x}^k, \mathbf{v}^k\right)$ and $\mathbf{v}^k$ the process noise. That way, a prior estimate of the current state $\mathbf{x}^{k,-}$ is obtained. In the *update* step, a newly available observation $\tilde{\mathbf{z}}^k$ is incorporated into the predictive density $p^-(\mathbf{x}^k)$. The posterior density of the dynamical state can be derived with Bayes' rule and is given by

$$p^+(\mathbf{x}^k) \triangleq p(\mathbf{x}^k|\tilde{\mathbf{z}}^{0:k}) = \eta^k p(\tilde{\mathbf{z}}^k|\mathbf{x}^k)p^-(\mathbf{x}^k) \quad , \tag{3.2}$$
$$\text{with} \quad \eta^k \triangleq \left(\int p(\tilde{\mathbf{z}}^k|\mathbf{x}^k)p^-(\mathbf{x}^k)\,\mathrm{d}\mathbf{x}^k\right)^{-1} .$$

The term η^k represents the normalization constant, the term $p(\tilde{\mathbf{z}}^k|\mathbf{x}^k)$ is called the *likelihood function* of $\mathbf{x}^k$ for a given observation $\tilde{\mathbf{z}}^k$ and depends on the observation model $\mathbf{z}^k = h^k_{obs}\left(\mathbf{x}^k, \mathbf{w}^k\right)$ and the observation noise $\mathbf{w}^k$. The posterior $p^+(\mathbf{x}^k)$ is the probability distribution over the $\mathbf{x}^k$ at time step k, conditioned on all past observations $\mathbf{z}^{0:k}$ and is also referred to as *belief* or *state of knowledge* [Thr05].

Bayesian filtering provides an optimal solution for equation 3.1 and 3.2 and can be considered a statistical inversion problem [Hub15]. In figure 3.1, the *prediction-update* cycle of a Bayesian filter is visualized. In general, a closed-form solution of the filtering equations is not possible due to the integrals and multiplications of density functions involved. Thus, simplifying assumptions are required.

Under the assumption of linear dynamical and observation models affected by Gaussian noise, the seminal **Kalman filter** (KF) [Kal60] is optimal and has a closed-form solution. For the case of $\mathbf{w}^k$ and $\mathbf{v}^k$ being uncorrelated, the KF is an optimal dynamical state estimator in the sense of the least square errors and Bayesian filtering [Gri18]. Thus, the (linear) Kalman filter is a method for exact on-line inference in a linear DS and linear DSs are commonly used to describe basic dynamical models, such as a CV model.

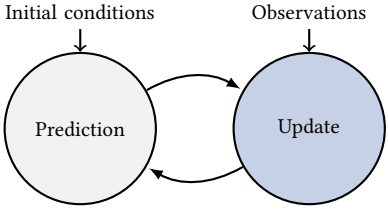

Figure 3.1: Visualization of the *prediction-update* cycle of a Bayesian filter. The filter recursively estimates the unknown system state $\mathbf{x}^k$ from the observations $\mathbf{z}^k$ and estimated state $\hat{\mathbf{x}}^{k-1}$ using the dynamical model and the observation model.

3.1.1 Kalman Filter

For the **Kalman filter**, the dynamical model from equation 2.1 and the observation model from 2.2 is restricted to linear equations. Accordingly, the dynamical model can be described by equation

$$\mathbf{x}^{k+1} = \mathbf{F}^k \mathbf{x}^k + \mathbf{G}^k \mathbf{v}^k \tag{3.3}$$

and the observation model

$$\mathbf{z}^k = \mathbf{H}^k \mathbf{x}^k + \mathbf{w}^k. \tag{3.4}$$

Hereby, $\mathbf{F}^k \in \mathbb{R}^{n_x \times n_x}$ is the system matrix of the Kalman filter and $\mathbf{H}^k \in \mathbb{R}^{n_z \times n_x}$ the observation matrix. The noise processes $\mathbf{v}^k \in \mathbb{R}^{n_v}$ and $\mathbf{w}^k \in \mathbb{R}^{n_z}$ are assumed to be white Gaussian noise process with known covariance matrices $\mathbf{Q}^k$ and $\mathbf{R}^k$. Further, it is assumed that $\mathbf{v}^k$ and $\mathbf{w}^k$ are uncorrelated. $\mathbf{G}^k \in \mathbb{R}^{n_x \times n_v}$ is the noise gain, over which the system noise enters the dynamical system:

$$\mathbf{P}_{\mathbf{vv}} \triangleq \mathrm{Cov}(\mathbf{v}^i \mathbf{v}^k) = \mathbb{E}[\mathbf{v}^i {\mathbf{v}^k}^\mathsf{T}] = \begin{cases} \mathbf{Q}^k & \text{for} \quad i = k \\ 0 & \text{for} \quad i \neq k \end{cases}, \tag{3.5}$$

$$\mathbf{P}_{\mathbf{ww}} \triangleq \mathrm{Cov}(\mathbf{w}^i \mathbf{w}^k) = \mathbb{E}[\mathbf{w}^i {\mathbf{w}^k}^\mathsf{T}] = \begin{cases} \mathbf{R}^k & \text{for} \quad i = k \\ 0 & \text{for} \quad i \neq k \end{cases}. \tag{3.6}$$

Hence,

$$\mathbf{v}^k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^k) \quad , \tag{3.7}$$

$$\mathbf{w}^k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}^k) \quad , \tag{3.8}$$

with

$$\mathbf{P}_{\mathbf{vw}} \triangleq \mathrm{Cov}(\mathbf{v}^i \mathbf{w}^k) = \mathbb{E}[\mathbf{w}^i {\mathbf{w}^k}^\mathsf{T}] = \mathbf{0} \quad \forall i, k, \tag{3.9}$$

$$\mathbf{P}_{\mathbf{x}^0\mathbf{v}} \triangleq \mathrm{Cov}(\mathbf{x}^0 \mathbf{v}^k) = \mathbb{E}[\mathbf{x}^0 {\mathbf{v}^k}^\mathsf{T}] = \mathbf{0} \quad \forall k, \tag{3.10}$$

$$\mathbf{P}_{\mathbf{x}^0\mathbf{w}} \triangleq \mathrm{Cov}(\mathbf{x}^0 \mathbf{w}^k) = \mathbb{E}[\mathbf{x}^0 {\mathbf{w}^k}^\mathsf{T}] = \mathbf{0} \quad \forall k. \tag{3.11}$$

Given that a Gaussian distribution can be represented by the two first moments, the state estimate boils down to calculating the mean vector $\hat{\mathbf{x}}^k$ and covariance matrix $\mathbf{P}_{\mathbf{xx}}^k$ of the true state $\mathbf{x}^k$. The dynamical state representation is given by

$$\mathbf{x}^k \sim \mathcal{N}(\hat{\mathbf{x}}^k, \mathbf{P}_{\mathbf{xx}}^k), \tag{3.12}$$

with

$$\begin{aligned} \mathbf{P}_{\mathbf{xx}}^k \triangleq \mathrm{Cov}(\mathbf{x}^k \mathbf{x}^k) &= \mathbb{E}[(\mathbf{x}^k - \mathbb{E}[\mathbf{x}]^k)(\mathbf{x}^k - \mathbb{E}[\mathbf{x}]^k)^\mathsf{T}] \\ &= \mathbb{E}[(\mathbf{x}^k - \hat{\mathbf{x}}^k)(\mathbf{x}^k - \hat{\mathbf{x}}^k)^\mathsf{T}]. \end{aligned} \tag{3.13}$$

$$\tag{3.14}$$

The prior estimates, which are obtained during the *prediction* step and do not account for the current observation, are denoted by $\hat{\mathbf{x}}^{k,-}$ and $\mathbf{P}_{\mathbf{xx}}^{k,-}$:

$$\hat{\mathbf{x}}^{k,-} = \mathbb{E}[\mathbf{x}^k | \tilde{\mathbf{z}}^{0:k-1}], \tag{3.15}$$

$$\mathbf{P}_{\mathbf{xx}}^{k,-} = \mathbb{E}[(\mathbf{x}^k - \hat{\mathbf{x}}^{k,-})(\mathbf{x}^k - \hat{\mathbf{x}}^{k,-})^\mathsf{T}]. \tag{3.16}$$

In the *update* step, the current observation is incorporated to obtain the posterior estimate:

$$\hat{\mathbf{x}}^{k,+} = \mathbb{E}[\mathbf{x}^k | \tilde{\mathbf{z}}^{0:k}], \tag{3.17}$$

$$\mathbf{P}_{\mathbf{xx}}^{k,+} = \mathbb{E}[(\mathbf{x}^k - \hat{\mathbf{x}}^{k,+})(\mathbf{x}^k - \hat{\mathbf{x}}^{k,+})^\mathsf{T}]. \tag{3.18}$$

At time $k = 0$, the Kalman filter is initialized with the prior distribution for the state $\mathcal{N}(\hat{\mathbf{x}}^0, \mathbf{P}_{\mathbf{xx}}^0)$. The *prediction* step of a KF can be derived by using the transition function from equation 3.3 in the expectation computation:

$$\begin{aligned}
\hat{\mathbf{x}}^{k,-} &= \mathbb{E}[\mathbf{x}^k | \tilde{\mathbf{z}}^{0:k-1}] \\
&= \mathbb{E}[\mathbf{F}\mathbf{x}^{k-1} + \mathbf{G}\mathbf{v}^{k-1} | \tilde{\mathbf{z}}^{0:k-1}] \\
&= \mathbb{E}[\mathbf{F}\mathbf{x}^{k-1} | \tilde{\mathbf{z}}^{0:k-1}] + \mathbb{E}[\mathbf{G}\mathbf{v}^{k-1} | \tilde{\mathbf{z}}^{0:k-1}] \\
&= \mathbf{F}\,\mathbb{E}[\mathbf{x}^{k-1} | \tilde{\mathbf{z}}^{0:k-1}] + \mathbf{0} \\
&= \mathbf{F}\hat{\mathbf{x}}^{k-1,+}.
\end{aligned} \tag{3.19}$$

The associated covariance matrix results from equation 3.19 and 3.3:

$$\begin{aligned}
\mathbf{P}_{\mathbf{xx}}^{k,-} &= \mathbb{E}[(\mathbf{x}^k - \hat{\mathbf{x}}^{k,-})(\mathbf{x}^k - \hat{\mathbf{x}}^{k,-})^\mathsf{T}] \\
&= \mathbb{E}[(\mathbf{F}\mathbf{x}^{k-1} + \mathbf{G}\mathbf{v}^{k-1} - \mathbf{F}\hat{\mathbf{x}}^{k-1,+})(\mathbf{F}\mathbf{x}^{k-1} + \mathbf{G}\mathbf{v}^{k-1} - \mathbf{F}\hat{\mathbf{x}}^{k-1,+})^\mathsf{T}] \\
&= \mathbb{E}[(\mathbf{F}(\mathbf{x}^{k-1} - \hat{\mathbf{x}}^{k-1,+}) + \mathbf{G}\mathbf{v}^{k-1})((\mathbf{x}^{k-1} - \hat{\mathbf{x}}^{k-1,+})^\mathsf{T}\mathbf{F}^\mathsf{T} + {\mathbf{v}^{k-1}}^\mathsf{T}\mathbf{G}^\mathsf{T})] \\
&= \mathbb{E}[\mathbf{F}(\mathbf{x}^{k-1} - \hat{\mathbf{x}}^{k-1,+})(\mathbf{x}^{k-1} - \hat{\mathbf{x}}^{k-1,+})^\mathsf{T}\mathbf{F}^\mathsf{T}] \\
&\quad + \mathbb{E}[\mathbf{F}(\mathbf{x}^{k-1} - \hat{\mathbf{x}}^{k-1,+}){\mathbf{v}^{k-1}}^\mathsf{T}\mathbf{G}^\mathsf{T}] \\
&\quad + \mathbb{E}[\mathbf{G}\mathbf{v}^{k-1}(\mathbf{x}^{k-1} - \hat{\mathbf{x}}^{k-1,+})^\mathsf{T}\mathbf{F}^\mathsf{T}] + \mathbb{E}[\mathbf{G}\mathbf{v}^{k-1}{\mathbf{v}^{k-1}}^\mathsf{T}\mathbf{G}^\mathsf{T}] \\
&= \mathbf{F}\,\mathbb{E}[(\mathbf{x}^{k-1} - \hat{\mathbf{x}}^{k-1,+})(\mathbf{x}^{k-1} - \hat{\mathbf{x}}^{k-1,+})^\mathsf{T}]\mathbf{F}^\mathsf{T} + \mathbf{0} + \mathbf{0}
\end{aligned}$$

$$
\begin{aligned}
&+ \mathbf{G}\,\mathbb{E}[\mathbf{v}^{k-1}{\mathbf{v}^{k-1}}^{\mathsf{T}}]\mathbf{G}^{\mathsf{T}} \\
&= \mathbf{F}\mathbf{P}_{\mathbf{xx}}^{k-1,+}\mathbf{F}^{\mathsf{T}} + \mathbf{G}\mathbf{Q}^{k-1}\mathbf{G}^{\mathsf{T}}.
\end{aligned} \tag{3.20}
$$

In equation 3.20, $\mathbf{Q}^k$ reflects the uncertainty in the dynamical model. The uncertainty $\mathbf{P}_{\mathbf{xx}}^{k,-}$ of the dynamical state increases for every one-step prediction in accordance with this equation.

In order to calculate the corresponding posterior estimates for the *update* step on the basis of the current observation, the yet unknown quantities of $\hat{\mathbf{z}}^k$, $\mathbf{P}_{\mathbf{xz}}^k$, $\mathbf{P}_{\mathbf{zz}}^k$, and $\mathbf{P}_{\mathbf{xz}}^k$ must first be determined. The expected value of the observation can be obtained under consideration of the observation model

$$
\begin{aligned}
\hat{\mathbf{z}}^k &= \mathbb{E}[\tilde{\mathbf{z}}^k] \\
&= \mathbb{E}[\mathbf{H}\mathbf{x}^k + \mathbf{w}^k] \\
&= \mathbf{H}^k\,\mathbb{E}[\mathbf{x}^k] + \mathbb{E}[\mathbf{w}^k] \\
&= \mathbf{H}\hat{\mathbf{x}}^{k,-}.
\end{aligned} \tag{3.21}
$$

The difference between the actually obtained observation $\tilde{\mathbf{z}}^k$ and predicted $\hat{\mathbf{z}}^k$ is called *innovation* or *residuum*.

$$
\begin{aligned}
\mathbf{s}^k &\triangleq \tilde{\mathbf{z}}^k - \hat{\mathbf{z}}^k \\
&= \tilde{\mathbf{z}}^k - \mathbf{H}\hat{\mathbf{x}}^{k,-},
\end{aligned} \tag{3.22}
$$

where the innovation or residual covariance $\mathbf{P}_{\mathbf{zz}}^k \triangleq \mathbf{S}^k$ is given by:

$$
\mathbf{P}_{\mathbf{zz}}^k = \mathbb{E}[(\tilde{\mathbf{z}}^k - \hat{\mathbf{z}}^k)(\tilde{\mathbf{z}}^k - \hat{\mathbf{z}}^k)^{\mathsf{T}}]. \tag{3.23}
$$

By inserting the observation model and equation 3.23 we obtain

$$\begin{aligned}
\mathbf{P}_{\mathbf{zz}}^{k} &= \mathbb{E}[(\mathbf{H}^{k}\mathbf{x}^{k} + \mathbf{w}^{k} - \mathbf{H}\hat{\mathbf{x}}^{k})(\mathbf{H}^{k}\mathbf{x}^{k} + \mathbf{w}^{k} - \mathbf{H}\hat{\mathbf{x}}^{k})^{\mathsf{T}}] \\
&= \mathbb{E}[\mathbf{H}^{k}(\mathbf{x}^{k} - \hat{\mathbf{x}}^{k})(\mathbf{x}^{k} - \hat{\mathbf{x}}^{k})^{\mathsf{T}}\mathbf{H}^{k^{\mathsf{T}}}] + \mathbb{E}[\mathbf{H}^{k}(\mathbf{x}^{k} - \hat{\mathbf{x}}^{k})\mathbf{w}^{k^{\mathsf{T}}}] \\
&\quad + \mathbb{E}[(\mathbf{x}^{k} - \hat{\mathbf{x}}^{k})^{\mathsf{T}}\mathbf{H}^{k^{\mathsf{T}}}] + \mathbb{E}[\mathbf{w}^{k}\mathbf{w}^{k^{\mathsf{T}}}] \\
&= \mathbf{H}^{k}\,\mathbb{E}[(\mathbf{x}^{k} - \hat{\mathbf{x}}^{k})(\mathbf{x}^{k} - \hat{\mathbf{x}}^{k})^{\mathsf{T}}]\mathbf{H}^{k^{\mathsf{T}}} + \mathbb{E}[\mathbf{w}^{k}\mathbf{w}^{k^{\mathsf{T}}}] \\
\mathbf{S}^{k} &= \mathbf{H}^{k}\mathbf{P}_{\mathbf{xx}}^{k,-}\mathbf{H}^{k^{\mathsf{T}}} + \mathbf{R}^{k}.
\end{aligned} \tag{3.24}$$

The matrix $\mathbf{P}_{\mathbf{xz}}^{k}$ can analogously be determined

$$\begin{aligned}
\mathbf{P}_{\mathbf{xz}}^{k} &= \mathbb{E}[(\mathbf{x}^{k} - \hat{\mathbf{x}}^{k})(\tilde{\mathbf{z}}^{k} - \hat{\mathbf{z}}^{k})^{\mathsf{T}}] \\
&= \mathbb{E}[(\mathbf{x}^{k} - \hat{\mathbf{x}}^{k})(\mathbf{H}^{k}\mathbf{x}^{k} + \mathbf{w}^{k} - \mathbf{H}\hat{\mathbf{x}}^{k})^{\mathsf{T}}] \\
&= \mathbb{E}[(\mathbf{x}^{k} - \hat{\mathbf{x}}^{k})(\mathbf{x}^{k} - \hat{\mathbf{x}}^{k})^{\mathsf{T}}\mathbf{H}^{k^{\mathsf{T}}}] + \mathbb{E}[(\mathbf{x}^{k} - \hat{\mathbf{x}}^{k})\mathbf{w}^{k^{\mathsf{T}}}] \\
&= \mathbb{E}[(\mathbf{x}^{k} - \hat{\mathbf{x}}^{k})(\mathbf{x}^{k} - \hat{\mathbf{x}}^{k})^{\mathsf{T}}]\mathbf{H}^{k^{\mathsf{T}}} \\
&= \mathbf{P}_{\mathbf{xx}}^{k,-}\mathbf{H}^{k^{\mathsf{T}}}.
\end{aligned} \tag{3.25}$$

For calculating the observation update, the posterior density (see equation 3.2) is conditioned on the current observation. For a given observation $\tilde{\mathbf{z}}^{k}$, the resulting observation update of the Kalman filter calculates the Gaussian posterior density $\mathcal{N}(\mathbf{x}^{k}; \hat{\mathbf{x}}^{k,+}, \mathbf{P}_{\mathbf{xx}}^{k,+})$ of the dynamical state $\mathbf{x}^{k}$ with mean vector and covariance matrix according to

$$\hat{\mathbf{x}}^{k,+} = \hat{\mathbf{x}}^{k,-} + \mathbf{P}_{\mathbf{xz}}^{k}\mathbf{P}_{\mathbf{xz}}^{k\,-1}(\tilde{\mathbf{z}}^{k} - \hat{\mathbf{z}}^{k}), \tag{3.26}$$

$$\mathbf{P}_{\mathbf{xx}}^{k,+} = \mathbf{P}_{\mathbf{xx}}^{k,-} + \mathbf{P}_{\mathbf{xz}}^{k}\mathbf{P}_{\mathbf{zz}}^{k\,-1}\mathbf{P}_{\mathbf{zx}}^{k}. \tag{3.27}$$

The expressions 3.26 and 3.27 can be derived by conditioning the joint Gaussian distribution of $\mathbf{x}$ and $\mathbf{z}$ on $\mathbf{z}$ (see for example [Hub15]). Substituting equations 3.21, 3.24, 3.25 together with

$$\mathbf{P}_{\mathbf{zx}}^{k} = \mathbf{P}_{\mathbf{zx}}^{k\,\mathsf{T}} = \mathbf{H}^{k}\mathbf{P}_{\mathbf{xx}}^{k,-} \tag{3.28}$$

in equation 3.26, we obtain the desired expression for calculating the posterior mean

$$\hat{\mathbf{x}}^{k,+} = \hat{\mathbf{x}}^{k,-} + \mathbf{P}_{\mathbf{xx}}^{k,-}\mathbf{H}^{k\mathsf{T}}(\mathbf{H}^k\mathbf{P}_{\mathbf{xx}}^{k,-}\mathbf{H}^{k\mathsf{T}} + \mathbf{R}^k)^{-1}(\tilde{\mathbf{z}}^k - \mathbf{H}^k\hat{\mathbf{x}}^{k,-}). \tag{3.29}$$

It is common to use the abbreviation

$$\mathbf{K}^k = \mathbf{P}_{\mathbf{xx}}^{k,-}\mathbf{H}^{k\mathsf{T}}(\mathbf{H}^k\mathbf{P}_{\mathbf{xx}}^{k,-}\mathbf{H}^{k\mathsf{T}} + \mathbf{R}^k)^{-1}. \tag{3.30}$$

The matrix $\mathbf{K}^k$ is the so-called Kalman *gain*. Using $\mathbf{K}^k$ the *update* step is given by:

$$\hat{\mathbf{x}}^{k,+} = \hat{\mathbf{x}}^{k,-} + \mathbf{K}^k(\tilde{\mathbf{z}}^k - \mathbf{H}^k\hat{\mathbf{x}}^{k,-}), \tag{3.31}$$

$$\mathbf{P}_{\mathbf{xx}}^{k,+} = \mathbf{P}_{\mathbf{xx}}^{k,-} - \mathbf{K}^k\mathbf{H}^k\mathbf{P}_{\mathbf{xx}}^{k,-} = \left(\mathbf{I} - \mathbf{K}^k\mathbf{H}^k\right)\mathbf{P}_{\mathbf{xx}}^{k,-}. \tag{3.32}$$

The KF provides equations for propagating Gaussian distributions through a linear system, resulting in a maintained Gaussian distribution. In case of non-linearities in at least the dynamical model or the observation model, this does not apply anymore. Further, the property that the joint density of $\mathbf{x}$ and $\mathbf{z}$ is also Gaussian is only satisfied for linear models. Most non-linear filtering approaches utilize the construction of a joint Gaussian of $\mathbf{x}$ and $\mathbf{z}$ and conditioning on $\mathbf{z}$ to derive the Kalman filter. Thus, these approaches aim to find an approximation for non-linear filtering problems. Some popular example filters are the EKF and the iterative EKF (IEKF), which approximate the non-linear function by using the Taylor series expansion around the mean of the Gaussian distribution. Other approaches, such as the UKF, approximate the distribution by means of a set of points that can be propagated through the non-linear functions and serve to determine the new distribution parameters. A generalization of this approach leads to the family of PFs.

These modification concepts of the KF can be transferred to concepts presented in the following. Although linear models are considered, the same techniques, that are explained in the sequel, can be used by linearization. A more elaborate description of non-linear filtering extension of the KF can be found in [Bar02, Sär13, Thr05].

3.1.2 Maneuvering Objects

In the absence of the problem of data association, maneuvering object tracking faces two interrelated main challenges: object motion mode uncertainty and non-linearity. Mode refers to true object motion or a pattern of behavior, and the dynamical model is a mathematical - usually simplified - description of the object motion with a certain accuracy level. Estimation is based on models, approximations of the modes, which precisely describe the truth. *Multiple-model* approaches are generally considered to be the mainstream approach to maneuvering object tracking under motion mode uncertainty [Li10].

In filtering, *multiple-model* approaches are included in the group of *adaptive* filtering (see for example [Bar02]). In general, a well functioning filter depends on an adequate choice of $\mathbf{Q}^k$ and $\mathbf{R}^k$. The concept of an *adaptive* filter considers every filter which adapts itself when it detects dynamics that the dynamical model cannot account for. In object tracking, maneuvers are defined as model mismatch problems and in addition to *multiple-model* approaches, so-called maneuver detection based methods are also considered as *adaptive* filters. Some example methods are *adjustable level process noise* [Zar09], *variable state dimension*, and *input estimation*. But these methods are generally considered to be too slow to compensate maneuvers [Bar02].

As described in section 2.3.2, *multiple-model* approaches are the preferred choice when the object motion is poorly described by a single model. These approaches assume that the system behaves in accordance with one of a finite number of dynamical models. The models can differ in noise levels or in their structure. Such systems are also referred to as *hybrid systems* or *hybrid dynamical state* methods since they augment the discrete motion state with the continuous dynamical state $\boldsymbol{\xi}^k = (\mathbf{x}^k, m^k)$. *Multiple-model* approaches can be further sub-divided into *static* and *switching multiple-model* approaches. Static approaches converge quickly to only the most probable model without recovering [Lab14]. Thus, a reinitialization of mismatched filters is required. This is accomplished by using the estimate from other models. Since the necessary modifications are a rigorously built-in part of *switching multiple-model* approaches, static approaches are not further considered.

From the broad set of proposed *switching multiple-model* approaches, there is no clear-cut best algorithm, but the **interacting multiple-model** (IMM) filter is considered to be the best compromise of low computational complexity and good tracking performance [Pit05]. In particular for the task of *intention* prediction the IMM filter is the primary approach [Sch13, Koo19, Bon14, Sch15, Has15a, Has15b].

The prediction problem for maneuvering objects with such multiple-model systems can be described as

$$y = f_{m^k}\left(z^{0:k}, c^{0:k}\right) + \epsilon,$$

where $m^k \in \mathcal{M} = \{m_1, \ldots, m_M\}$ denotes the mode or dynamical model at time k that is in effect during the sampling period ending at k. The dynamical model of the linear Kalman filter from equation 3.3 thus yields

$$\mathbf{x}^{k+1} = \mathbf{F}^k(m^k)\mathbf{x}^k + \mathbf{G}^k(m^k)\mathbf{v}^k(m^k), \tag{3.33}$$

and the adapted observation model is given by

$$\mathbf{z}^k = \mathbf{H}^k(m^k)\mathbf{x}^k + \mathbf{w}^k(m^k). \tag{3.34}$$

In case equation 3.33 and 3.34 correspond to linear dynamical systems, such systems are referred to as JMLS [Mur12]. Among other, the expressions SSSM and SLDS are also common. For the dynamical model and the observation model, the transition matrices are formulated depending on m^k. The mode at time k is assumed to be among the set of possible M models

$$m^k \in \{m_j\}_{j=1}^{M}. \tag{3.35}$$

The sequence of dynamical models through time k (qth mode history) is denoted as

$$\mathcal{M}^{k,q} = \left\{m_i^{1,q}, \ldots, m_i^{k,q}\right\} \quad q = 1, \ldots, M^k, \tag{3.36}$$

where $(\cdot)_i^{\kappa,q}$ is the model index i at time κ from history q and $1 \leq (\cdot)_i^{\kappa,q} \leq M$. Note that the number of histories increases exponentially with time. The switching between the motion models is assumed to be a Markov process with known transition probabilities. Thus a Markov-chain consisting of a **transition probability matrix** (TPM) with

$$p_{ij} \triangleq P(m^k = m_j | m^{k-1} = m_i). \tag{3.37}$$

The event that model j is in effect at time k is denoted as

$$m_j^k \triangleq \{m^k = m_j\} \tag{3.38}$$

and the conditional probability of the qth sequence of models

$$\alpha^{k,q} = P(\mathcal{M}^{k,q} | \mathbf{z}^{0:k}). \tag{3.39}$$

The qth sequence of models through time can be written as

$$\mathcal{M}^{k,q} = \left\{\mathcal{M}^{k,l}, m_j^k\right\}, \tag{3.40}$$

where l denotes the parent sequence with the last element m_j. Due to the Markov property,

$$P(m_j^k | \mathcal{M}^{k,q}) = P(m_j^k | m_i^{k-1}) \triangleq p_{ij}, \tag{3.41}$$

where the index i corresponds to the last model in the parent sequence l. An optimal estimator for such a system calculates the following expectation

$$\mathbb{E}[\mathbf{x}^k | \tilde{\mathbf{z}}^{0:k}] = \sum_{q=1}^{M^k} \mathbb{E}[\mathbf{x}^k | \mathcal{M}^{k,q}, \tilde{\mathbf{z}}^{0:k}] P(\mathcal{M}^{k,q} | \tilde{\mathbf{z}}^{0:k}). \tag{3.42}$$

Thus, the conditional pdf of the dynamical state $\mathbf{x}^k$ at time k using the theorem of total probability with respect to mutually exclusive and exhaustive set of events (equation 3.36), is a Gaussian mixture with an exponentially increasing

number of terms [Bar02]:

$$p(\mathbf{x}^k|\tilde{\mathbf{z}}^{0:k}) = \sum_{q=1}^{M^k} p(\mathbf{x}^k|\mathcal{M}^{k,q},\tilde{\mathbf{z}}^{0:k})P(\mathcal{M}^{k,q}|\tilde{\mathbf{z}}^{0:k}). \tag{3.43}$$

By using Bayes' rule, the probability of a sequence of models can be obtained as

$$\begin{aligned}
\alpha^{k,q} &= P(\mathcal{M}^{k,q}|\tilde{\mathbf{z}}^{0:k}) \\
&= P(\mathcal{M}^{k,q}|\tilde{\mathbf{z}}^{k},\tilde{\mathbf{z}}^{0:k-1}) \\
&= \eta p(\tilde{\mathbf{z}}^{k}|\mathcal{M}^{k,q},\tilde{\mathbf{z}}^{0:k-1})P(\mathcal{M}^{k,q}|\tilde{\mathbf{z}}^{0:k-1}) \\
&= \eta p(\tilde{\mathbf{z}}^{k}|\mathcal{M}^{k,q},\tilde{\mathbf{z}}^{0:k-1})P(m_j^k,\mathcal{M}^{k-1,l}|\tilde{\mathbf{z}}^{0:k-1}) \\
&= \eta p(\tilde{\mathbf{z}}^{k}|\mathcal{M}^{k,q},\tilde{\mathbf{z}}^{0:k-1})P(m_j^k|\mathcal{M}^{k-1,l},\tilde{\mathbf{z}}^{0:k-1})\alpha^{k-1,l} \\
&= \eta p(\tilde{\mathbf{z}}^{k}|\mathcal{M}^{k,q},\tilde{\mathbf{z}}^{0:k-1})P(m_j^k|\mathcal{M}^{k-1,l})\alpha^{k-1,l},
\end{aligned} \tag{3.44}$$

where η is the normalization constant. Since the current mode only depends on the previous, it follows:

$$\alpha^{k,q} = \eta p(\tilde{\mathbf{z}}^{k}|\mathcal{M}^{k,q},\tilde{\mathbf{z}}^{0:k-1})P(m_j^k|m_i^k)\alpha^{k-1,l} \tag{3.45}$$

$$= \eta p(\tilde{\mathbf{z}}^{k}|\mathcal{M}^{k,q},\mathbf{z}^{0:k-1})p_{ij}\alpha^{k-1,l}, \tag{3.46}$$

where $i = l^{k-1}$ is the index of the last model m^{k-1} of the parent sequence l. Equation 3.46 shows that even if the model sequence is Markov, a conditioning on the entire past history is required. In order to prevent a combinatorial explosion and to apply *multiple-models* in practice, approximations of the optimal solution are required. The different *multiple-model* approaches vary in the way how they approximate equation 3.43. As explained in section 2.3, they consist of the following elements [Li10]. Firstly, the adaptive set of selected dynamical models. Secondly, methods to deal with discrete value uncertainties, such as a Markov or a semi-Markov assumption. Thirdly, a recursive estimation scheme to deal with the continuous dynamical states conditioned

on the dynamical model. Fourthly, a strategy to estimate the overall best filter by fusion or selection of individual filters.

In [Pit05], Pitre et al. compared several *multiple-model* methods including generalized pseudo-Bayesian filter of first order (GPB1), and of second order (GPB2) [Cha78], IMM filter, B-best based *multiple-model* filter [Tug82], and Viterbi-based *multiple-model* algorithm [Ave91] for tracking applications and showed that the IMM filter offers the best compromise between good tracking performance and low computational complexity. GPB2 filter and IMM filter approximate equation 3.43 identically, but the GPB2 filter requires M^2 combined Kalman filters instead of the M combined Kalman filters of the IMM algorithm with similar tracking performance. A more detailed derivation of GPB filter can for example be found in [Bar02, Pit05]. Here, we focus on the IMM solution for the filtering problem.

The basic idea of IMM is to approximate equation 3.42 or rather equation 3.43 by

$$\mathbb{E}[\mathbf{x}^k|\tilde{\mathbf{z}}^{0:k}] = \sum_{j=1}^{M} \mathbb{E}[\mathbf{x}^k|m_j^k,\tilde{\mathbf{z}}^{0:k}]P(m_j^k|\tilde{\mathbf{z}}^{0:k}), \tag{3.47}$$

$$p(\mathbf{x}^k|\tilde{\mathbf{z}}^{0:k}) = \sum_{j=1}^{M} p(\mathbf{x}^k|m_j^k,\tilde{\mathbf{z}}^{0:k})P(m_j^k|\tilde{\mathbf{z}}^{0:k}). \tag{3.48}$$

Here, M basic filters run in parallel, and every filter is optimal for one specific state of the discrete Markov-chain, i.e., the dynamic model and the observation model fit to a specific mode with regards to equation 3.33 and equation 3.34. For the term $P(m_i^k|\mathbf{z}^{0:k})$, the posterior mode probability, the abbreviation α_i^k is used. The IMM algorithm consists of three major steps: *interaction* (mixing), *filtering*, and *combination*. The following derivation of these steps is oriented on [Wen11, Bar02].

IMM-Interaction

The first step of the IMM filter cycle is *interaction*. Here, the Markov-chain is propagated through time. Accordingly, the estimate of the motion mode

and the estimate of the dynamical state must be adapted. Thus, the following transition must be determined:

$$\alpha_j^{k-1} = \alpha_{jk-1}^{k-1} \rightarrow \alpha_{ik}^{k-1} \quad \forall i,j, \tag{3.49}$$

$$p(\mathbf{x}^{k-1}|m_j^{k-1},\tilde{\mathbf{z}}^{0:k-1}) \rightarrow p(\mathbf{x}^{k-1}|m_i^{k-1},\tilde{\mathbf{z}}^{0:k-1}) \quad \forall i,j. \tag{3.50}$$

The term $\alpha_{ik}^{k-1} \triangleq P(m_j^k|\tilde{\mathbf{z}}^{0:k-1})$ is the probability under the condition that all observation up to time step $k-1$ are available, and model i is in effect in time step k. By applying the *Chapman-Kolmogorov* equation (see equation 3.1) on the Markov-chain, the desired transition for equation 3.49 is given by

$$\alpha_{ik}^{k-1} = \sum_{j=1}^{M} \alpha_{ik|jk-1}^{k-1} \alpha_{jk-1}^{k-1} = \sum_{j=1}^{M} p_{ij} \alpha_{jk-1}^{k-1}. \tag{3.51}$$

Here, $\alpha_{ik|jk-1}^{k-1} \triangleq P(m_i^k|m_j^{k-1},\tilde{\mathbf{z}}^{0:k-1})$ is the probability under the condition that all observations up to time step $k-1$ are available, and model i is in effect in time step k for the case model j was in effect in time step $k-1$. When applying Bayes' rule, it follows

$$\alpha_{jk-1|ik}^{k-1} \alpha_{ik}^{k-1} = \alpha_{ik|jk-1}^{k-1} \alpha_{jk-1}^{k-1} = p_{ij} \alpha_{jk-1}^{k-1}, \tag{3.52}$$

and thus

$$\alpha_{jk-1|ik}^{k-1} = \frac{p_{ij}\alpha_{jk-1}^{k-1}}{\alpha_{ik}^{k-1}} = \frac{p_{ij}\alpha_{jk-1}^{k-1}}{\sum_{j=1}^{M} p_{ij}\alpha_{jk-1}^{k-1}} = \frac{p_{ij}\alpha_{jk-1}^{k-1}}{\bar{c}_i} \quad . \tag{3.53}$$

The desired pdf from equation 3.50 can now be described by

$$p(\mathbf{x}^{k-1}|m_i^{k-1},\tilde{\mathbf{z}}^{0:k-1}) = \sum_{j=1}^{M} p(\mathbf{x}^{k-1}|m_i^k, m_j^{k-1}, \tilde{\mathbf{z}}^{0:k-1}) \alpha_{jk-1|ik}^{k-1}. \tag{3.54}$$

The weighting probabilities $\alpha_{j|i}^{k} = \alpha_{j^{k-1}|i^{k}}^{k-1}$ are referred to as mixing or interacting probabilities. Using Bayes' rule, we get

$$\begin{aligned} &p(\mathbf{x}^{k-1}|m_i^k, m_j^{k-1}, \tilde{\mathbf{z}}^{0:k-1})P(m_i^k|m_j^{k-1}, \tilde{\mathbf{z}}^{0:k-1}) \\ &\quad = P(m_i^k|\mathbf{x}^{k-1}, m_j^{k-1}, \tilde{\mathbf{z}}^{0:k-1})p(\mathbf{x}^{k-1}|m_j^{k-1}, \tilde{\mathbf{z}}^{0:k-1}), \end{aligned} \tag{3.55}$$

and thus

$$\begin{aligned} &p(\mathbf{x}^{k-1}|m_i^k, m_j^{k-1}, \tilde{\mathbf{z}}^{0:k-1}) \\ &\quad = \frac{P(m_i^k|\mathbf{x}^{k-1}, m_j^{k-1}, \tilde{\mathbf{z}}^{0:k-1})}{P(m_i^k|m_j^{k-1}, \tilde{\mathbf{z}}^{0:k-1})} p(\mathbf{x}^{k-1}|m_j^{k-1}, \tilde{\mathbf{z}}^{0:k-1}). \end{aligned} \tag{3.56}$$

The probability of $m^k = m_i$ depends on $m^{k-1} = m_j$, but not on $\mathbf{x}^{k-1}$. Thus the terms of the fraction in equation 3.56 are equal. By using this simplified version of equation 3.56, we can also simplify 3.54, the desired pdf from equation 3.50, according to

$$p(\mathbf{x}^{k-1}|m_i^{k-1}, \tilde{\mathbf{z}}^{0:k-1}) = \sum_{j=1}^{M} p(\mathbf{x}^{k-1}|m_j^{k-1}, \tilde{\mathbf{z}}^{0:k-1})\alpha_{j|i}^{k-1}. \tag{3.57}$$

According to 3.57, the pdf of the dynamical state conditioned on model m_i is a weighted sum of M Gaussian distributions (mixture-of-Gaussian). The following step after *interaction*, is the *prediction* by using an elementary dynamical model. Thus, the sum of Gaussians from 3.57 has to be approximated by a single Gaussian distribution. The approximation by a single Gaussian can be done by moment matching (see for example [Bar02]). Thus, the mixed

initial condition (mixed mean and covariance) for each filter is computed as

$$\mathbf{x}_{0,i}^{k-1,+} = \sum_{j=1}^{M} \alpha_{j|i}^{k-1} \mathbf{x}_{j}^{k-1,+}, \tag{3.58}$$

$$\mathbf{P}_{0,i}^{k-1,+} = \sum_{j=1}^{M} \alpha_{j|i}^{k-1} \left(\mathbf{P}_{j}^{k-1,+} + (\hat{\mathbf{x}}_{j}^{k-1,+} - \mathbf{x}_{0,i}^{k-1,+})(\hat{\mathbf{x}}_{j}^{k-1,+} - \mathbf{x}_{0,i}^{k-1,+})^{\mathsf{T}} \right). \tag{3.59}$$

Here, $\hat{\mathbf{x}}_{j}^{k-1,+}$ and $\mathbf{P}_{j}^{k-1,+}$ are the updated mean and covariance for model j at time step $k-1$.

IMM-Filtering

In the *filtering* step, after initialization with $\mathbf{x}_{i}^{k-1,+}$ and $\mathbf{P}_{i}^{k-1,+}$, the Kalman filter equations (3.32, 3.31 and 3.19, 3.20) are applied for each individual filter. Correspondingly in the prediction step of the KF, the pdf of the models are propagated through time

$$p(\mathbf{x}^{k-1}|m_i^k, \tilde{\mathbf{z}}^{0:k-1}) \rightarrow p(\mathbf{x}^{k}|m_i^k, \tilde{\mathbf{z}}^{0:k-1}) \quad \forall i. \tag{3.60}$$

Thus, the prediction part is given by

$$\left[\mathbf{x}_{i}^{k,-}, \mathbf{P}_{i}^{k,-}\right] = KF_p \left[\mathbf{x}_{0,i}^{k-1,+}, \mathbf{P}_{0,i}^{k-1,+}, \mathbf{F}_{i}^{k-1}, \mathbf{Q}_{i}^{k-1}\right]. \tag{3.61}$$

Here the abbreviation $\mathbf{F}_{i}^{k}$ corresponds to $\mathbf{F}^{k}(m_i^k)$. This also applies accordingly to $\mathbf{Q}_{i}^{k}$, $\mathbf{H}_{i}^{k}$, and $\mathbf{R}_{i}^{k}$. The model probabilities are not affected here. In the *update* step of the IMM filter, the pdf of the filters are updated

$$p(\mathbf{x}^{k-1}|m_i^k, \tilde{\mathbf{z}}^{0:k}) \rightarrow p(\mathbf{x}^{k}|m_i^k, \tilde{\mathbf{z}}^{0:k}) \quad \forall i, \tag{3.62}$$

and thus

$$\left[\mathbf{x}_{i}^{k,+}, \mathbf{P}_{i}^{k,+}\right] = KF_u \left[\mathbf{x}_{i}^{k,-}, \mathbf{P}_{i}^{k,-}, \mathbf{H}_{i}^{k}, \mathbf{R}_{i}^{k}\right]. \tag{3.63}$$

In addition to the parameter of the pdf, the model probabilities have to be adapted

$$\alpha_{ik}^{k-1} \rightarrow \alpha_i^k \quad \forall i. \tag{3.64}$$

Using Bayes' rule yields

$$P(m_i^k|\tilde{\mathbf{z}}^k,\tilde{\mathbf{z}}^{0:k-1},\mathbf{x}_i^{k,-}) = \frac{p(\tilde{\mathbf{z}}^k|m_i^k,\tilde{\mathbf{z}}^{0:k-1},\mathbf{x}_i^{k,-})P(m_i^k|\tilde{\mathbf{z}}^{0:k-1},\mathbf{x}_i^{k,-})}{p(\tilde{\mathbf{z}}^k|\tilde{\mathbf{z}}^{0:k-1},\mathbf{x}_i^{k,-})}. \tag{3.65}$$

The likelihood Λ_i^k of the observation for each filter is computed

$$\begin{aligned}\Lambda_i^k &= p(\tilde{\mathbf{z}}^k|\mathbf{x}_i^{k,-}) \\ &= \frac{1}{\sqrt{(2\pi)^{n_z}\left|\mathbf{S}_i^k\right|}} \exp\left(-\frac{1}{2}{\mathbf{s}_i^k}^{\mathsf{T}}{\mathbf{S}_i^k}^{-1}\mathbf{s}_i^k\right),\end{aligned} \tag{3.66}$$

where $\mathbf{s}_i^k$ is the observation *innovation*, and $\mathbf{S}_i^k$ the *innovation* covariance (see equations 3.22 and 3.23) of the KF *update* step of model m_i.

Due to the fact that in $\mathbf{x}_i^{k,-}$ all observations till time step $k-1$ are incorporated, it is possible to write

$$p(\tilde{\mathbf{z}}^k|m_i^k,\tilde{\mathbf{z}}^{0:k-1},\mathbf{x}_i^{k,-}) = p(\tilde{\mathbf{z}}^k|\mathbf{x}_i^{k,-}) = \Lambda_i^k, \tag{3.67}$$

and we get for equation 3.65

$$\alpha_i^k = \frac{\Lambda_i^k \alpha_{ik}^{k-1}}{\sum_{i=1}^{M} \Lambda_i^k \alpha_{ik}^{k-1}}. \tag{3.68}$$

Note that α_{ik}^{k-1} is the propagated model probability, thus we can use the expression $\bar{c}_i$ from equation 3.53 to rewrite 3.68 the model probability update

equation according to

$$\alpha_i^k = \frac{1}{c}\Lambda_i^k \bar{c}_i. \tag{3.69}$$

Here, $c = \sum_{i=1}^{M} \Lambda_i^k \bar{c}_i$ is the normalization factor for equation 3.69. This expression is commonly used in literature ([Bar02, Lab14, Sär13]).

IMM-Combination

The final step in an IMM cycle is *combination*. The combination of the model conditioned estimates and covariances is done according to mode matching of the Gaussian mixture as follows:

$$\hat{\mathbf{x}}^{k,+} = \sum_{j=1}^{M} \alpha_j^k \mathbf{x}_j^{k,+}, \tag{3.70}$$

$$\hat{\mathbf{P}}^{k,+} = \sum_{j=1}^{M} \alpha_j^k \left(\mathbf{P}_j^{k,+} + (\mathbf{x}_j^{k,+} - \hat{\mathbf{x}}^{k,+})(\mathbf{x}_j^{k,+} - \hat{\mathbf{x}}^{k,+})^\mathsf{T} \right). \tag{3.71}$$

This completes a full IMM filter cycle. For a better overview the most important equations of the IMM algorithm are summarized in the following.

- **Interaction**

  Mixing probabilities:

$$\alpha_{j|i}^{k-1} = \frac{p_{ij}\alpha_j^{k-1}}{\sum_{j=1}^{M} p_{ij}\alpha_j^{k-1}}.$$

Mixed mean and mixed covariance:

$$\mathbf{x}_{0,i}^{k-1,+} = \sum_{j=1}^{M} \alpha_{j|i}^{k-1} \mathbf{x}_{j}^{k-1,+},$$

$$\mathbf{P}_{0,i}^{k-1,+} = \sum_{j=1}^{M} \alpha_{j|i}^{k-1} \left(\mathbf{P}_{j}^{k-1,+} + (\hat{\mathbf{x}}_{j}^{k-1,+} - \mathbf{x}_{0,i}^{k-1,+})(\hat{\mathbf{x}}_{j}^{k-1,+} - \mathbf{x}_{0,i}^{k-1,+})^{\mathsf{T}} \right).$$

- **Filtering**

  Prediction:

$$\left[\mathbf{x}_{i}^{k,-}, \mathbf{P}_{i}^{k,-}\right] = KF_{p}\left[\mathbf{x}_{0,i}^{k-1,+}, \mathbf{P}_{0,i}^{k-1,+}, \mathbf{F}_{i}^{k-1}, \mathbf{Q}_{i}^{k-1}\right].$$

  Update:

$$\left[\mathbf{x}_{i}^{k,+}, \mathbf{P}_{i}^{k,+}\right] = KF_{u}\left[\mathbf{x}_{i}^{k,-}, \mathbf{P}_{i}^{k,-}, \mathbf{H}_{i}^{k}, \mathbf{R}_{i}^{k}\right].$$

  Model probability update:

$$\alpha_{i}^{k} = \frac{1}{c} \Lambda_{i}^{k} \bar{c}_{i},$$
$$\text{with} \quad c = \sum_{i=1}^{M} \Lambda_{i}^{k} \quad ; \quad \bar{c}_{i} = \sum_{j=1}^{M} p_{ij} \alpha_{j}^{k-1}.$$

- **Combination**

  Combined mean and combined covariance:

$$\hat{\mathbf{x}}^{k,+} = \sum_{j=1}^{M} \alpha_{j}^{k} \mathbf{x}_{j}^{k,+},$$

$$\hat{\mathbf{P}}^{k,+} = \sum_{j=1}^{M} \alpha_{j}^{k} \left(\mathbf{P}_{j}^{k,+} + (\mathbf{x}_{j}^{k,+} - \hat{\mathbf{x}}^{k,+})(\mathbf{x}_{j}^{k,+} - \hat{\mathbf{x}}^{k,+})^{\mathsf{T}} \right).$$

3.2 IMM Filter for Visual Tracking

In this chapter, we present our adapted designs for a basic IMM filter by both a state de-coupling and re-coupling scheme as modifications. All filters are applied as top-down state estimator in a visual tracking pipeline.

3.2.1 De-coupled IMM Filter

The dynamical models of recursive Bayesian filters rely on explicitly defined dynamic equations that follow physical models such as Newton's law of motion. In order to apply a *physics*-based dynamical model, not only a good physical model is required, but in addition, mapping between the observations to the $3D$ physical world. As described in section 2.3, the condition of being able to rely on mapping to the physical world can be ensured by utilizing *contextual* cues to better interpret the observed scene or by including more assumptions about the environment. For example, in order to perform *path prediction* on ground level additional sensors (LIDAR, stereo camera system) or approaches like structure-from-motion (SfM) reconstruct the 3D scene.

Although the aim of several approaches is to determine such a mapping function, there exist several scenarios where this mapping is unknown, involves substantially higher expense, or is an unsolved problem. Accordingly, without implicit contextual cues to estimate a mapping function, object tracking is performed directly in the 2D image space. Thus, the objects are solely tracked on directly mapped observations from the appearance model. A typical example is the tracking of objects without available external calibration. Figure 3.2 shows two examples where the state estimate solely relies on the enclosing bounding box of the object. In particular, the observation models are mapped linearly to the dynamical state of the object. As a consequence, the dynamical models are abused as general-purpose models to capture the object motion. In other words, the dynamical models are only rudimentary models of the true object motion mode with relatively large process noise levels.

As discussed earlier, an IMM filter is a good choice for dealing with motion uncertainties and reducing the effect of model mismatching. The effect of model

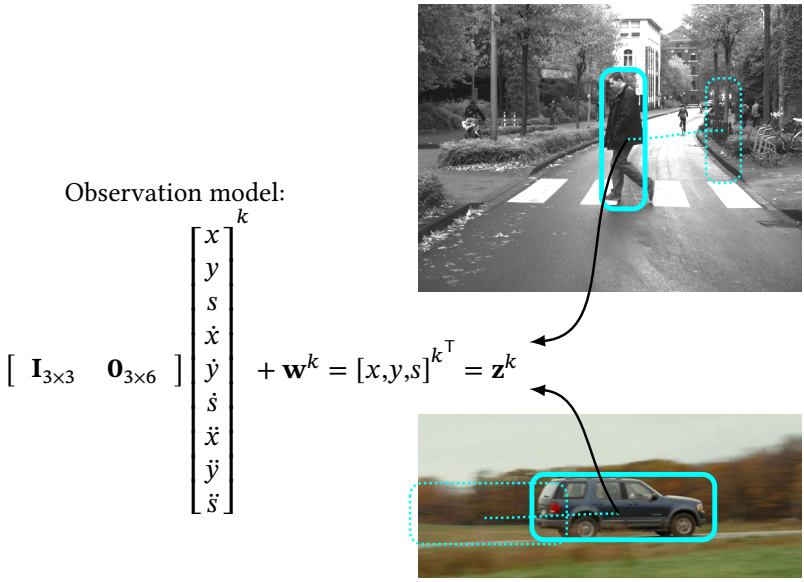

Figure 3.2: Two tracked objects where the dynamical state is directly estimated from observations provided by the output of a person detector or a visual tracker in terms of the center position (x,y) and the object scale (s). (Top) An example image from the *Daimler Mono Pedestrian dataset* [Enz09]. (Bottom) An example image from the sequence *car* of the *VOT*2014 dataset [Ceh16, Kri14].

mismatching is clearly present in a detection-by-tracking pipeline for tracking in image space. Although an IMM filter can describe more complex object dynamics by combining several basic dynamical models, there arise some fallacies when an IMM filter is restricted to this situation. In the following, these fallacies are analyzed by comparing a standard IMM filter setup to a proposed de-coupling of the dynamical states for the case of tracking objects only with directly mapped object observations. At first, a reference IMM setup for the desired scenario is introduced. For combining several dynamical models, the dynamical state of the object is described according to

$$\mathbf{x}^{k+1} = \mathbf{F}_i^k \mathbf{x}^k + \mathbf{v}_i^k, \tag{3.72}$$

and the observation model is given by

$$\mathbf{z}^k = \mathbf{H}_i^k \mathbf{x}^k + \mathbf{w}_i^k. \tag{3.73}$$

The observation noise $\mathbf{w}^k \in \mathbb{R}^{n_z}$ is assumed to be uncorrelated to the process noise and modeled as white Gaussian noise process $\mathbf{w}^k \sim \mathcal{N}(0,\mathbf{R}_w)$. For the goal of tracking an object on directly mapped observations, the $\mathbf{H}_i^k$ includes only binary values. As observations provided by an appearance model, the unified bounding box is used. Thus, $\mathbf{z}^k$ includes the center position (x,y) of the object in image $\mathbf{I}^k$, and the object scale s (see figure 3.2). Such information can be obtained from every object detector following the sliding window paradigm. Although common detectors differ in many aspects, the output of such a sliding window-based detector is a rectangular bounding box centered at the object location [Dol12, Enz09]. Alternatively, almost every visual tracker compared in the study of Cehovin et al. [Ceh16] uses the enclosing bounding box to represent the object state in the image. In order to choose an adaptive model set, the three most common general-purpose dynamical models are considered. These dynamical models are the **constant position** (CP), the **constant velocity** (CV), and the **constant acceleration** (CA). Despite being applicable as translational models for tracking in image space, these models are used for modeling the motion behavior of pedestrians for *intention* prediction with a single dynamical model [Møg15, Eln01, Bin05], or combined as *multiple-model* approach [Sch15, Gol14, Sch13]. Accordingly, the transition matrices $\mathbf{F}_i^k$ can then be defined as

$$\mathbf{F}_{CP}^k = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times6} \\ \mathbf{0}_{6\times3} & \mathbf{0}_{6\times6} \end{bmatrix} \tag{3.74}$$

for the CP model, and as

$$\mathbf{F}_{CV}^k = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{I}_{3\times3}\Delta T & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}. \tag{3.75}$$

for the CV model. In literature, several assumptions on how to model the acceleration process of an object are proposed. Here, in accordance with Li et al. [Li05] the following CA model has been chosen

$$\mathbf{F}_{CA}^{k} = \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{I}_{3\times3}\Delta T & \mathbf{I}_{3\times3}\frac{1}{2}\Delta T^2 \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{I}_{3\times3}\Delta T \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix}. \tag{3.76}$$

Hence, $\mathbf{F}_{i}^{k} \in \left\{\mathbf{F}_{CP}^{k}, \mathbf{F}_{CV}^{k}, \mathbf{F}_{CA}^{k}\right\}$ with $m_{i}^{k} \in \left\{m_{CP}^{k}, m_{CV}^{k}, m_{CA}^{k}\right\}$. In case these models are applied in image space, every single model is used together with a relatively high model uncertainty. With the above choice of the model set structure the dynamical state $\mathbf{x}^{k}$ of the reference IMM filter is given by

$$\mathbf{x}_{IMM\,1}^{k} = \left[x, y, s, \dot{x}, \dot{y}, \dot{s}, \ddot{x}, \ddot{y}, \ddot{s}\right]^{\mathsf{T}}. \tag{3.77}$$

In addition to a directly observed center position (x,y) and the object scale (s), the IMM filter uses the corresponding velocities $(\dot{x},\dot{y},\dot{s})$, and accelerations $(\ddot{x},\ddot{y},\ddot{s})$. For standard Kalman filtering, a de-coupling of the states is redundant. Due to the characteristics of an IMM filter, both choosing a wrong single motion model and carelessly extending the states can lead to a non-optimal performance. The IMM solution to avoid the combinatorial explosion of an optimal filtering behavior using *multiple-models* is done by conditioning all filters on the currently active model, and the final state estimate is obtained by merging the results of all base filters (see section 3.1.2). Thus, a poor estimate of the active model affects the weighting of the mixed inputs in the *interaction* step. Thereby, a combination of the location and the scale in a single state vector can result in errors in the calculation of the model probability, especially when combining the scale with the image position. For example, the scale change of an object can be constant while the object is moving. Thus, the best fitting model for the scale is CP, although this model is a poor fit for the image position. Therefore, we propose to de-couple the state estimate. In practice, this is done by using an additional IMM filter. Hence, the scale and the corresponding velocity, and acceleration are estimated independently from the position states and their derivatives. This first state separation step

leads to the following IMM configuration:

$$\mathbf{x}_{IMM\,2}^{k} = \left[\left[x,y,\dot{x},\dot{y},\ddot{x},\ddot{y}\right],\left[s,\dot{s},\ddot{s}\right]\right]^{\mathsf{T}} = \left[\mathbf{x}_{1}^{k\,\mathsf{T}},\mathbf{x}_{2}^{k\,\mathsf{T}}\right]^{\mathsf{T}}. \tag{3.78}$$

Thus, the state estimation problem can be written in terms of two de-coupled state sub-vectors

$$\begin{bmatrix} \mathbf{x}_{1}^{k+1} \\ \mathbf{x}_{2}^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{1,j}^{k} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{6\times6} & \mathbf{F}_{2,j}^{k} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{1}^{k} \\ \mathbf{x}_{2}^{k} \end{bmatrix} + \begin{bmatrix} \mathbf{G}_{1}^{k}\mathbf{v}_{1,j}^{k} \\ \mathbf{G}_{2}^{k}\mathbf{v}_{2,j}^{k} \end{bmatrix}. \tag{3.79}$$

A separation of the scale with an additional filter seems obvious, but when tracking with directly mapped image space data, a split into independent image coordinates may not immediately appear to be necessary. In order to show the benefit of such an IMM setup, we recommend the following IMM configuration:

$$\mathbf{x}_{IMM\,3}^{k} = \left[\left[x,\dot{x},\ddot{x}\right],\left[y,\dot{y},\ddot{y}\right],\left[s,\dot{s},\ddot{s}\right]\right]^{\mathsf{T}} = \left[\mathbf{x}_{1}^{k\,\mathsf{T}},\mathbf{x}_{2}^{k\,\mathsf{T}},\mathbf{x}_{3}^{k\,\mathsf{T}}\right]^{\mathsf{T}}. \tag{3.80}$$

The proposed de-coupled estimator results in

$$\begin{bmatrix} \mathbf{x}_{1}^{k+1} \\ \mathbf{x}_{2}^{k+1} \\ \mathbf{x}_{3}^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{1,j}^{k} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{F}_{2,j}^{k} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{F}_{3,j}^{k} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{1}^{k} \\ \mathbf{x}_{2}^{k} \\ \mathbf{x}_{3}^{k} \end{bmatrix} + \begin{bmatrix} \mathbf{G}_{1}^{k}\mathbf{v}_{1,j}^{k} \\ \mathbf{G}_{2}^{k}\mathbf{v}_{2,j}^{k} \\ \mathbf{G}_{3}^{k}\mathbf{v}_{3,j}^{k} \end{bmatrix}. \tag{3.81}$$

Here, three IMM filters are used to describe the x position, y position, s scale, and corresponding derivatives. Thus, every motion along the image axes is captured with a separate filter.

The strategy of de-coupling state estimates and basing the estimator on reduced-order filters has hitherto been mainly used in air traffic surveillance. In this field, usually, an aircraft's motion in the horizontal plane is independent of its vertical motion [Bar02]. For example, in the work of Yeddanapudi et al. [Yed97] and Wang et al. [Wan99], the aircraft motion in North and East direction is estimated with a separated filter to a second

filter for the vertical state (the altitude and the vertical velocity). Besides the explained benefits, especially for an IMM filter restricted to directly mapped image space observations, the computational complexity of a de-coupled system is also reduced compared to a system using only one state vector.

3.2.2 Evaluation: De-coupled IMM Filter

The above described IMM configurations are evaluated on the *VOT*2014 dataset [Ceh16, Kri14]. This dataset is a selection of 25 prototypical object tracking sequences. Although the dataset is originally designed to compare different visual trackers, it includes a variety of different object motions from different object categories. Some example sequences from the *VOT*2014 dataset are depicted in figure 3.3. Due to the fact that the object type, the capturing sensor, and the tracking scenario differ strongly, this dataset includes several situations in which the estimation of the mapping function to a 3D physical reference system is an unsolved problem or would require high expense.

Settings: The overall performance depends on a number of design parameters. The most critical design parameters are the model set structure, process and observation noises, initial state, and the jump structure given by the transition probabilities. Although, the basic IMM setup with three standard motion models (CP, CV, CA) is sub-optimal for some scenarios of the VOT2014 dataset, this combination is kept fixed. In practice, the TPM is often assumed to be known and is chosen a priori. As stated in Bar-Shalom [Bar02], an ad-hoc approach is to fill the diagonals with values close to 1. We set the diagonals to 0.99 and the remaining transition values to 0.005. The IMM filter is relatively insensitive to small changes in TPM. Since the CV model is the most widely used in tracking approaches, we set the initial model probability α_i^0 in favor of this model to 0.98 and to 0.01 for the other models. The observation noise is assumed to be an additive white noise. The process noise is modeled as the acceleration increment during a sampling interval (discrete Wiener process acceleration). In the experiments, the variances of

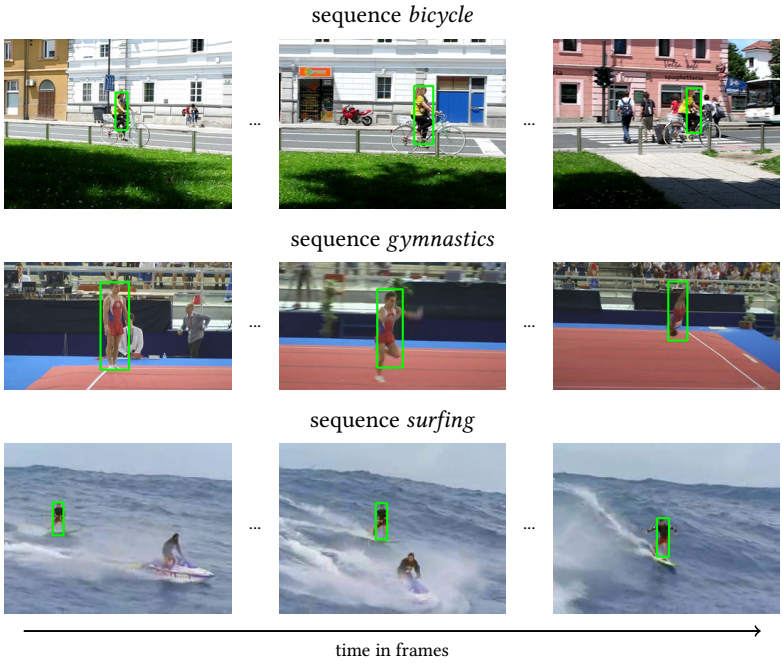

Figure 3.3: Example tracking sequences from the *VOT*2014 dataset [Ceh16, Kri14]. Unified bounding boxes of the objects are shown for the sequences *bicycle*, *gymnastics*, and *surfing*.

both noise processes were varied between 1, 2, 5, and 10.

Results: For every image sequence, the first 10 frames are excluded and used for initializing the filters. The update interval t_{update} for getting a new observation for the filter has been varied between every single, every third, and every fifth frame. Since the standard outputs of object detectors are a rectangular bounding box centered at the object location, we use the ground truth bounding boxes from the VOT2014 dataset for evaluating the prediction accuracy. Performance measures aim at summarizing the extent to which the tracker's prediction agrees with the ground truth annotation. In Cehovin et

Table 3.1: Results for the comparison between different IMM de-coupling configurations on the *VOT*2014 dataset. Settings: $\sigma_v^2 = 2, \sigma_w^2 = 5, t_{update} = 3$

	failure rate			average IoU		
sequence	IMM 1	IMM 2	IMM 3	IMM 1	IMM 2	IMM 3
ball	0.270	0.191	0.164	0.634	0.679	0.695
basketball	0.003	0.003	0.004	0.863	0.884	0.891
bicycle	0.304	0.233	0.173	0.602	0.641	0.692
bolt	0.080	0.044	0.027	0.774	0.810	0.842
car	0.340	0.261	0.108	0.610	0.642	0.710
david	0.167	0.152	0.141	0.697	0.715	0.720
driving	0.082	0.135	0.135	0.793	0.749	0.736
drunk	0.000	0.000	0.000	0.931	0.929	0.931
fernado	0.018	0.021	0.018	0.857	0.852	0.859
fish1	0.242	0.144	0.111	0.663	0.726	0.725
fish2	0.107	0.084	0.064	0.745	0.769	0.775
gymnastics	0.107	0.138	0.138	0.798	0.787	0.786
hand1	0.262	0.232	0.227	0.656	0.664	0.665
hand2	0.410	0.379	0.359	0.542	0.559	0.576
jogging	0.047	0.054	0.047	0.769	0.776	0.777
motocross	0.092	0.188	0.188	0.752	0.745	0.755
polarbear	0.011	0.008	0.011	0.848	0.849	0.852
skating	0.000	0.000	0.000	0.866	0.881	0.898
sphere	0.295	0.300	0.316	0.618	0.629	0.605
sunshade	0.559	0.571	0.484	0.426	0.442	0.488
surfing	0.111	0.081	0.048	0.699	0.746	0.773
torus	0.150	0.146	0.142	0.706	0.723	0.712
trellis	0.358	0.276	0.238	0.579	0.633	0.661
tunnel	0.129	0.078	0.101	0.695	0.734	0.729
woman	0.051	0.053	0.051	0.773	0.794	0.805

al. [Ceh14], a general definition of an object state description in a sequence with length N is established based on the center of the object and the region of the object at time k.

From the IMM filter, the predicted center location x, y, and scale s are used to calculate an unified bounding box A_0^k. The overlap between the predicted

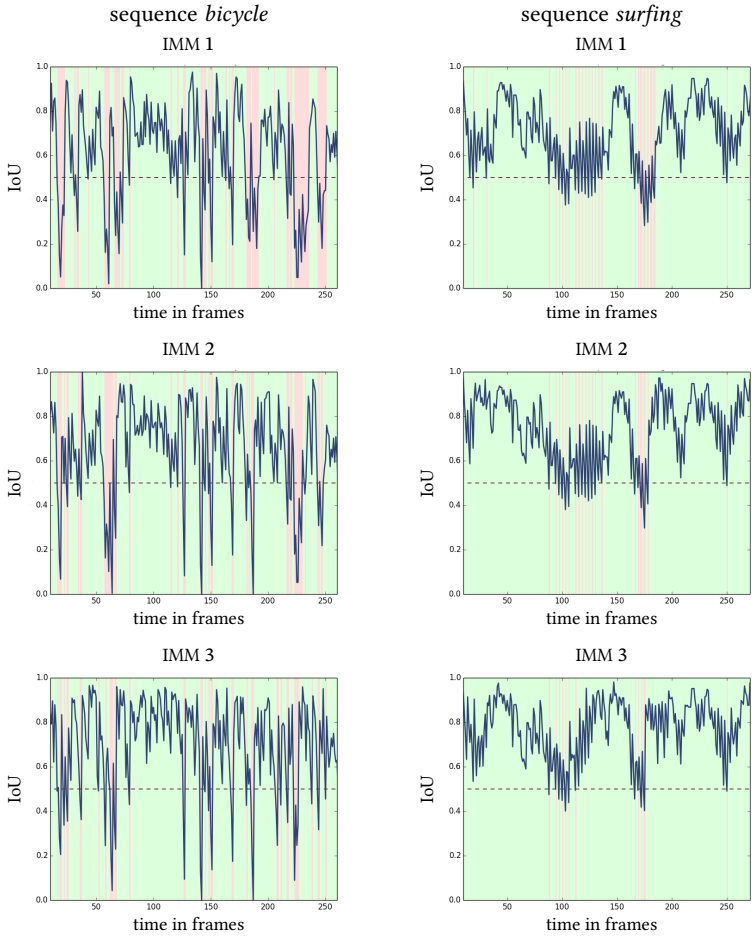

Figure 3.4: Visualization of the failure rate for the sequences *bicycle* and *surfing* with an overlap threshold of 0.5 for $\sigma_v^2 = 2$, $\sigma_w^2 = 5$, $t_{update} = 3$.

and the ground truth region can be calculated as

$$\text{IoU} = \frac{\left|\hat{A}_O^k \cap A_{GT}^k\right|}{\left|\hat{A}_O^k \cup A_{GT}^k\right|}. \tag{3.82}$$

The overlap ratio is often referred to as **intersection-over-union** (IoU) or Jaccard index [Jac08]. For the ground truth area A_{GT}^{k}, also an unified bounding box is considered. In general, the width of the enclosing bounding box is more strongly influenced by the body pose of the objects. Hence, a unified bounding box with a width of 1/3 of the bounding box height is used. Although the selected ratio is better suited for object categories such as pedestrians, this ratio also works for object categories with deviating width to height ratio to achieve a good assignment between observation and prediction. A property of the IoU is that they simultaneously account for position and size. Thus, there is no need for additional normalization considerations. The IoU is summarized over an entire sequence by an average IoU. In addition to the average IoU, the failure rate is used as a second comparative score. The failure is the number of frames in which the IoU is below a threshold of 0.5 is recorded and is computed as

$$\text{failure rate} = \frac{\text{number of frames IoU} < 0.5}{\text{number of frames}}. \tag{3.83}$$

The overall results for the three different IMM configurations are exemplary summarized for $\sigma_w^2 = 2$, $\sigma_r^2 = 5$, $t_{update} = 3$ in table 3.1. The corresponding visualization of the failure rate for the sequences *bicycle* and *surfing* (see figure 3.3) with an overlap threshold of 0.5 is shown in figure 3.4. Other parameter settings may differ slightly, but are equal at their core. This means that the achieved overlap varies and that for some specific sequences, the ranking of the IMM configuration changes. Overall it can be noticed that the IMM configuration that uses separated image space coordinates and scale, outperforms the other configurations. Due to the fact that the motion of objects in some particular sequence is highly non-linear, the chosen combination of motion models is not optimal. Moreover, this can also result in a changed ranking, but the trend towards the third configuration for achieving superior results is visible for all evaluated parameter settings. It should be noted that in particular two sequences (*gymnastics* and *diving*) do not comply with the results achieved on the other sequences. Firstly, both include objects which execute a rotation. Here, we used unified bounding boxes and associated the height with the object scale. When the object is rotating, this association is

wrong and leads to highly non-linear motion patterns due to the change in the bounding box orientation. Under such conditions, the current rotation should be considered in the state of the object or otherwise it is not possible to associate the object scale with the bounding box height. Besides, it is inevitably difficult to produce uncontroversial ground truth boxes for rotating objects. Hence, the annotations for these sequences include a stronger ambiguity for the enclosing bounding boxes, and are erroneous in some cases.
Due to the fact that the ground truth includes an uncertainty and the overlap values for ranking the different filter configurations are in some cases very close to each other, a hypotheses test is applied as follows. The VOT2014 dataset contains 25 videos, and the evaluation has been structured such that each sequence provides a single data point that can be used to conduct a pairwise test between the IMM filter configurations. Given two IMM configurations, a sequence is categorized in favor of one configuration based on the average overlap. Equal values are excluded. The counts for these cases will follow a binomial distribution. Furthermore, if the setups are equivalent, the probability of one versus the other should be 0.5. Binomial statistics can then be used to compute a p-value and determine whether or not to reject the null hypothesis that both configurations are equivalent. The setup for the applied exact binomial tests are listed in the following and the results are summarized in table 3.2.

- Hypothesis statement: Compared IMM configurations perform equally well.
- Null hypothesis : H_0: $P_0 = 1/2$
- Alternate hypothesis: H_1: $P_0 \neq 1/2$
- Test distribution: $X_{test} \sim \mathcal{B}in(N_{test}, P)$
- Test statistics: L = Number an IMM configuration performs better for a test sequence.
- p-value (two-sided): $p\text{-value} = 2 \cdot \sum_{l=0}^{L} \binom{N_{test}}{l} P_0^l (1 - P_0)^{N_{test} - l}$
- Significance level: $\alpha_{test} = 0.05$

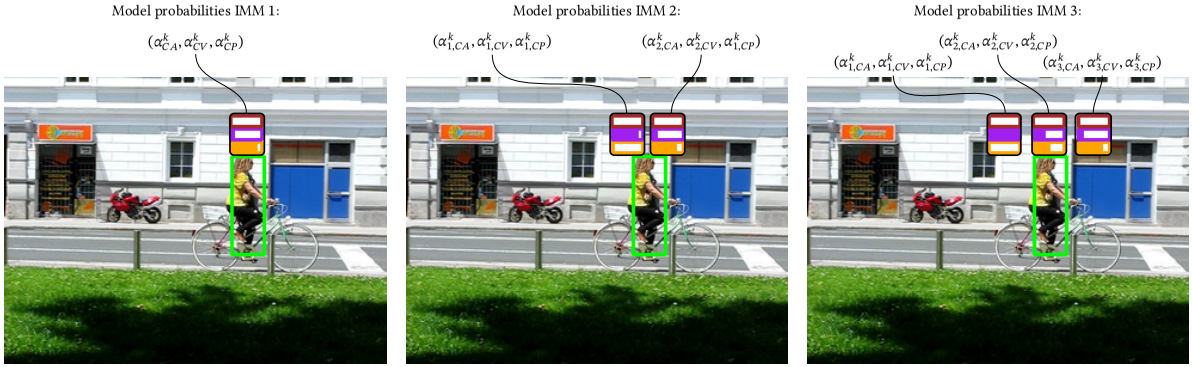

Figure 3.5: Visualization of the model probabilities of the different IMM filters using the described motion models for an example image from the sequence *bicycle* of the *VOT*2014 dataset [Ceh16, Kri14]. The length of the colored bars depict the value of the current model probability (CP = orange ▬; CV = purple ▬; CA = brown ▬). (Top) The current best fitting model is CP, even though the cyclist is in motion. (Middle) Here, the best fitting model for $\mathbf{x}_1^k$ and $\mathbf{x}_2^k$ is CV or rather CP. (Bottom) For $\mathbf{x}_1^k$ the currently best fitting model is CV. Whereas for $\mathbf{x}_2^k$ the model probabilities are relatively even split between CV and CP, and for $\mathbf{x}_3^k$ the best fitting model is CP.

Table 3.2: Statistical hypothesis tests for the different IMM configurations on the *VOT*2014 dataset based on the results from table 3.1.

Test: IMM 3 vs. IMM 2		Test: IMM 3 vs IMM 1		Test: IMM 2 vs IMM 1	
IMM 3 > IMM 2	19	IMM 3 > IMM 1	21	IMM 2 > IMM 1	20
IMM 3 < IMM 2	6	IMM 3 < IMM 1	3	IMM 2 < IMM 1	5
p-value (two-sided)	0.0146	p-value (two-sided)	0.0003	p-value (two-sided)	0.0041
Null hypothesis	rejected	Null hypothesis	rejected	Null hypothesis	rejected

In all cases, the null hypothesis is rejected due to p-values indicating that the differences in performance were significant. As mentioned, other parameter settings lead to only slight variations in performance and can therefore lead to a less distinctive result for a specific setup. Nevertheless, the overall trend towards a de-coupled IMM filter shows that this configuration is an improvement over the other IMM filter configurations. An improvement can also be perceived by de-coupling location and scale. Thus, the second configuration (IMM 2) outperforms the naive state extension (IMM 1). This state de-coupling is also recommended when the actual motion is described in 3D. These results follow the intuition that when tracking an object directly in image space, the motion in a particular direction can be independent from the other direction. Although this assumption is not always true, it is commonly used for many filter designs, including single model variants. Because the base filters are conditioned on the best fitting model, the final estimate is negatively influenced by a naive extension of the state vector. Figure 3.5 illustrates this effect by visualizing the model probabilities of the de-coupled IMM filter. There, the individual model probabilities differ for all de-coupled states. For combining the scale and its derived changes with the actual motion states this seems obvious. On the contrary, the presented results show how crucial this can also become for mixing image coordinates.

In summary, when relying on directly mapped observations, which is common for tracking in image space, a naive extension of the state vector should be avoided. However, the fact that independent states are affected by mixing the inputs from the base filters, which is a result of the required approximation for optimal filtering, can easily be overseen when applying IMM filters for tracking in image space. With this simple reminder, a better IMM filtering

can be achieved. While the overall performance can be further improved by selecting alternative motion models which better fit to the dynamics of the object in the scene, it is also crucial to not just naively extend the state. All states of an IMM state vector should depend on each other and thus, each additional independent state and its derivatives should be considered in an additional IMM filter. Thus, the conditioning on the current best fitting model can not negatively affect the overall performance. The motion of an object in image space is a good example of a case in which the dynamics along the image axes should be considered independently when applied to an IMM filter.

3.2.3 Re-coupled IMM filter

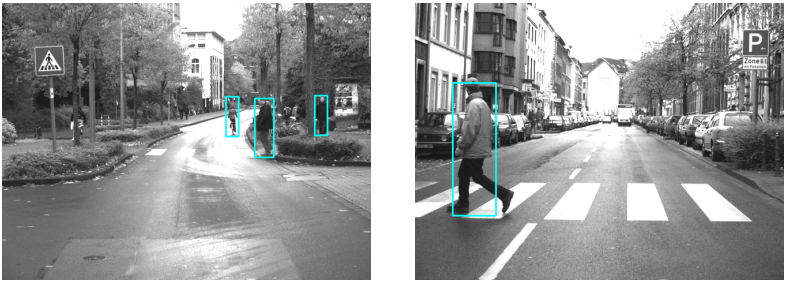

Figure 3.6: Example images of detected persons with the approach of Kieritz et al. [Kie16] on the *Daimler Mono Pedestrian* dataset [Enz09].

In this section, a state re-coupling scheme for the IMM filter configuration is introduced. The proposed re-coupled IMM filter provides an online adaptation scheme of the system noise parameters in order to better capture location uncertainties pertaining to image coordinates. The observation $\mathbf{z}^k$ for tracking in image space is often obtained from a object detector. As an example, figure 3.6 shows detected persons with the approach of Kieritz et al. [Kie16]. The applied person detector follows the widely used sliding window paradigm. Thereby, for every window location and scale a binary classification is done.

The classifier consists of weighted decisions trees using the integral over a rectangular region of a feature channel as nodes, and are generated by boosting [Fre97]. Thus, the output of such detectors, as well as that of most visual tracking approaches, is a rectangular bounding box. Similar to section 3.2.2, the performance of person detectors is also measured by the IoU between the detector output and the ground truth bounding box. A standard threshold for a detector output to be categorized as true positive is 0.5 (see for example Dollar et al. [Dol12]).

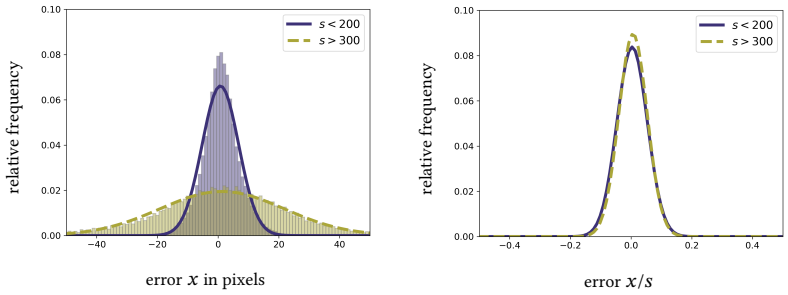

Figure 3.7: (Left) Distribution of the detection error in x for persons smaller than 200 pixels and person greater than 300 pixels. (Right) Distribution of the detection error in x relative to the ground truth person height.

The IoU criterion bears the risk that the observation noise scale dependency is easily overseen. For sequences where the range of person or other object scales is very limited, a fixed observation noise variance is adequate. Nevertheless, it is intuitively clear that the location accuracy is scale-dependent. For demonstrating the scale-dependency of the observation noise, we evaluated our person detector (Kieritz et al. [Kie16]) on the *MOT*16 dataset [Mil16] and the *Daimler Mono Pedestrian* dataset [Enz09] for covering a broad range of person scales. Only detections with an overlap greater than 0.5 are considered in the analysis. Without applying a non-maximum suppression on the detector output (multiple detections are thereby associated with a single annotation), a total number of 450826 detections were compared to the ground

truth data. By dividing the range of person scales into several intervals, the effect of scale dependency for the pixel distance in x and y direction can be seen. When the pixel distance is normalized by the object scale, a zero-mean error distribution with a relatively constant variance over the chosen scale intervals could be observed. This is shown for the error in x in figure 3.7. It can be clearly seen that the displacement or error is larger for higher person scales. Normalized with the true scale, the standard deviation of the error is nearly constant.
In order to take the shown dependency between the noise levels and the scale of the object into account, we propose a re-coupling of states of the IMM filter. The overall state-space model of the proposed IMM filter can be expressed as:

$$\begin{bmatrix} \mathbf{x}_1^{k+1} \\ \mathbf{x}_2^{k+1} \\ \mathbf{x}_3^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{1,i}^k & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{F}_{2,i}^k & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{F}_{3,i}^k \end{bmatrix} \begin{bmatrix} \mathbf{x}_1^k \\ \mathbf{x}_2^k \\ \mathbf{x}_3^k \end{bmatrix} + \begin{bmatrix} \mathbf{G}_1^k \mathbf{v}_{1,i}^k \\ \mathbf{G}_2^k \mathbf{v}_{2,i}^k \\ \mathbf{G}_1^k \mathbf{v}_{3,i}^k \end{bmatrix}, \tag{3.84}$$

$$\begin{bmatrix} \mathbf{z}_1^k \\ \mathbf{z}_2^k \\ \mathbf{z}_3^k \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{1,i}^k & \mathbf{0}_{3\times3} & \mathbf{D}_{1,i}^k \\ \mathbf{0}_{3\times3} & \mathbf{H}_{2,i}^k & \mathbf{D}_{2,i}^k \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{H}_{3,i}^k \end{bmatrix} \begin{bmatrix} \mathbf{x}_1^k \\ \mathbf{x}_2^k \\ \mathbf{x}_3^k \end{bmatrix} + \begin{bmatrix} \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times1} \\ \mathbf{w}_{3,i}^k \end{bmatrix}. \tag{3.85}$$

Here, $\mathbf{D}^k$ is a weighting matrix for tuning the scale-dependent noise. In the overall dynamical model, we realized a coupling of the scale dynamics to image coordinate dynamics which is very similar to augmenting a state for dealing with time-correlated noise (see for example Wendel et al. [Wen04]). It should be noted that the scale has to be strictly positive, but assuming a Gaussian distribution for the filter does allow negative values. Since the detection of persons requires a reasonable scale, the problem can be neglected in most cases. By using a relative scale, this problem can be avoided and even more important, the noise process can be modeled by a zero-mean white sequence. Without using the relative scale change, the noise of the scale is state-dependent. For considering this situation, we refer for example to Spinello et al. [Spi10]. In the case of visual tracking, it is common to describe

the scale change relative to the initial scale. Thus, the state vector $\mathbf{x}_3^k$ can be replaced with the relative scale change $s_r = {}^{s}/_{s_0}$ and the corresponding velocity and acceleration $[s_r, \dot{s_r}, \ddot{s_r}]^\mathsf{T}$. In sequences where the range of person scales is very limited, like the *VOT*2014 dataset, a fixed setting of the noise variances is sufficient. But in scenarios with a relatively fast change of the object scale, for example images collected from a driving vehicle, the proposed IMM filter helps to avoid an under- or over-estimation of the system uncertainties.

3.2.4 Evaluation: Re-coupled IMM Filter

The benefit of the complete scheme of re-coupling IMM filter states is evaluated on 10 selected sequences from the *Daimler Mono Pedestrian* dataset [Enz09], which was captured on-board a vehicle driving through an urban environment. Figure 3.8 depicts the trajectories of person ground truth boxes, where the increase of the scale is clearly visible. Although in such a scenario, the object motion is preferentially modeled in an ego-motion compensated vehicle centered coordination system on ground level, here no further contextual cues in order to enable an association between the observation and the 3*D* environment are considered. Nevertheless, the object type and sensor setup are known in this case, the constraint of tracking directly in image space is kept. The modeling of pedestrian motion relying on physical motion models is discussed in section 4.

Settings: For the following evaluation the model set combination from section 3.2.3 is used. We also kept the previously chosen transition matrix and the initial probability values. The fixed observation noise standard deviations for the x and y dynamics were set to $\left\lceil \sqrt{0.03 \cdot {}^{s_0}/_{\text{pixels}}} \right\rceil$ and the adaptive observation noise levels were set $\left\lceil \sqrt{0.03 \cdot {}^{\hat{s}^k}/_{\text{pixels}}} \right\rceil$. The noise levels for the scale dynamics were modeled as white Gaussian noise. The process noise standard deviation was set to $\left\lceil \sqrt{0.1 \cdot {}^{s_0}/_{\text{pixels}}} \right\rceil$ for all three filters and modeled as the acceleration increment during a sampling period. Since the error for the scale estimation is identical, it is excluded from the evaluation. Thus,

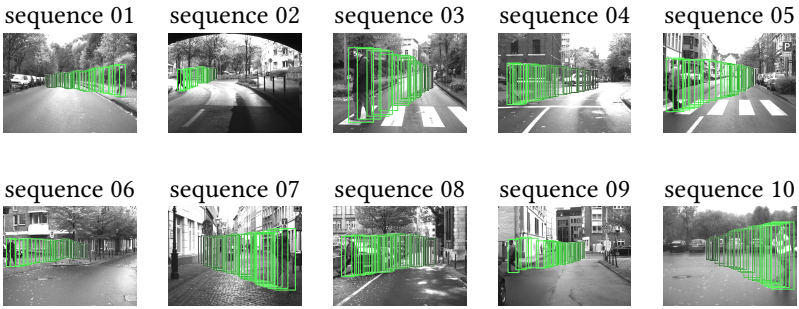

Figure 3.8: Visualization of trajectories of person bounding boxes from the Daimler Mono Pedestrian dataset [Enz09].

only the image location error $e = \sqrt{(x_{GT} - \hat{x})^2 + (y_{GT} - \hat{y})^2}$ in form of the **root-mean-squared error** (RMSE) is considered. For simulating different person detectors, the ground truth bounding boxes were used. Zero-mean white Gaussian noise was added to the ground truth center location and scale. For taking the scale-dependency into account, the additional noise term was set to $\sqrt{0.04 \cdot s^k/_{\text{pixels}}}$.

Results: The results for the described setup for $N_r = 1000$ runs is shown in table 3.3. Here, $\bar{e}$ is the average RMSE and σ_e the corresponding standard deviation. The last two columns include the average ratio of the sum of the RMSE ($r_{rec/dec} = \frac{1}{N_r}(\sum_0^N e_{k,rec})/(\sum_0^N e_{k,dec})$) of a sequence and its standard deviation. In these experiments, the adaptive noise tuning was done stepwise by mapping it to fixed levels in order to avoid an oversensitive tuning. Alternatively, the current uncertainty of the scale estimate can in addition be considered for preventing an oversensitive noise adjustment that might result in an erroneous assessment. As can be seen in table 3.3, the results achieved with the de-coupled filter is inferior to the re-coupled filter. The difference in terms of the average RMSE is small, but the variance for the re-coupled filter is lower. The results show the benefit of a re-coupled filter and prototypical situations for applying it. For sequences where the change in scale is

Table 3.3: RMSE analysis for the de-coupled and re-coupled IMM filter on selected sequences of the *Daimler Mono Pedestrian dataset* [Enz09]. Settings: $\sigma_v = \left\lceil \sqrt{0.1 \cdot s_{0/\text{pixels}}} \right\rceil$, $\sigma_w = \left\lceil \sqrt{0.03 \cdot s_{0/\text{pixels}}} \right\rceil$

sequence	frame numbers	$\bar{e}_{dec}$	$\sigma_{e_{dec}}$	$\bar{e}_{rec}$	$\sigma_{e_{rec}}$	$r_{rec/dec}$	$\sigma_{r_{rec/dec}}$
01	2678 - 2708	4.330	4.174	3.978	2.962	0.886	0.168
02	2875 - 2900	3.425	2.580	3.350	2.079	0.964	0.115
03	4686 - 4712	11.321	16.153	9.822	11.169	0.882	0.195
04	4892 - 4921	3.668	7.003	6.432	5.615	0.923	0.128
05	5974 - 6016	6.301	6.905	5.879	5.615	0.886	0.180
06	11047 - 11076	4.679	3.922	4.623	3.795	0.986	0.110
07	11248 - 11283	9.584	7.762	9.111	7.086	0.935	0.129
08	11485 - 11521	8.436	7.381	8.136	6.882	0.951	0.111
09	11796 - 11842	5.365	5.887	4.821	4.362	0.852	0.230
10	17342 - 17366	10.711	10.555	9.594	7.578	0.906	0.165

less pronounced, like sequence 06, both filters perform equally well, but the re-coupled filter bears the risk of a too strong noise adaptation because of the scale estimation uncertainty. This is also consistent with modeling the detection uncertainty with an additive fixed term, where the performance can accordingly shift towards the de-coupled IMM filter. But in sequences with rapid scale changes, the advantage of the re-coupled filter gets more significant. This can be seen from the lower $r_{rec/dec}$ values for the re-coupled filter for the chosen settings.

The results from table 3.3 mainly conduces to illustrate some effects for tracking solely in image space. But it also shows some limitations of the chosen linear filter setup for scenarios captured from a driving vehicle. However, the effect of fixed noise levels can directly be derived from the above error distribution analysis of the person detector. The filter gain multiplies the prior uncertainty $\mathbf{P}_{\mathbf{xx}}^{k,-}\mathbf{H}^{k^\mathsf{T}}$ with the inverse observation uncertainty (residual covariance: $\mathbf{S}^k = \mathbf{H}^k\mathbf{P}_{\mathbf{xx}}^{k,-}\mathbf{H}^{k^\mathsf{T}} + \mathbf{R}^k$; see equation 3.30). Although a division is not defined for matrices, we can think of the Kalman gain as a ratio that controls the influence of a new observation on the updated (posterior) state estimate. For example in sequence 03, the ground truth scale of the fully visible person changes from 103 pixels to 360 pixels. As mentioned, a true

positive is considered up to an IoU of 0.5. Hence, an admissible correct detection to ground truth association can result in relatively high localization errors. Hence, a suggested error confined to y can result in an error of half the person scale. For sequence 03, up to 180 pixels. Assuming that the prior uncertainty and the state estimate from the last frame are identical, the filter gain difference only depends on the noise variance. Thus, a small noise variance would strongly underestimate the observation uncertainty and lead to almost complete correction of the estimated position to the measured position and vice versa.
Besides the fact that the overall performance of person detectors increases for close ranges or rather large scales [Dol12], this is not true for the localization accuracy (see figure 3.7). In case the underlying detection and localization scheme improves for larger scales, a fixed observation noise level and de-coupled filter setup is sufficient. This also implies that the commonly used IoU value as a reference value for considering a true positive detection is only adequate for assessing the detection task, but should be more restrictive for larger scales, especially in combination with tracking. The combination of a filter and a detector with very large image location uncertainties is also impractical.
In summary, when tracking solely in image space the observation uncertainties provided by commonly used person detectors or visual trackers are scale-dependent. The proposed re-coupled IMM filter helps to improve dealing with these conditions. The advantage of the re-coupling scheme for an IMM filter is more significant in scenarios where tracked objects cover a broader range of scales, or their scales show a high dynamic (see figure 3.9).

Figure 3.9: Comparison between estimated trajectories with a re-coupled ( ) and a de-coupled) IMM filter, and the corresponding ground truth trajectory ( ). The noisy observation is highlighted in lime ( ). In the examples from sequence 03 and sequence 10, the positive effect of adjusting the observation noise level is visible.

3.3 Assets and Drawbacks of IMM Filters

A maneuver is any motion characteristic that an object is performing other than the dynamical model used by the filter. In case the object maneuvers are

in a set of finite number of models, *multiple-model* approaches are the preferred choice to deal with such a model mismatch. However, the tuning of a filter, the choice of its design parameters, requires a large amount of engineering. Since tuning of filters aims to systematically connect the filter parameters to physical system parameters, this is extremely hard for tracking directly in image space. Thus the filter setup consists of simplified models of the true motion mode with large noise levels to deal with the model uncertainty. The proposed modifications of a basic IMM filter design help to improve the filter performance, in particular for tracking in image space.

It is clear that for most application scenarios a mapping function to 3D is crucial to further improve the tracking performance. Nevertheless, filter tuning requires still a large amount of engineering and adequate physical models. To overcome the limitation of an IMM filter, we propose RNN-based alternatives that obtain the key abilities of an IMM. We shift from a *physics*-based modeling to a *pattern*-based modeling of the object dynamics. The goal is to describe complex object motion and capture the intention of switching in dynamics. Thus, the probabilities of the currently performed motion dynamics that is crucial for higher-level processing should also be provided by the system.

The assets and drawbacks of the IMM filter are summarized in table 3.4.

Table 3.4: Summary of the assets and drawbacks of IMM filters.

	Assets	**Drawbacks**
IMM filters	+ Most common architecture to capture switching dynamics. + Ability to describe complex object motion and capture latent intention. + Probability of motion dynamics that is currently performed. + Low computational complexity.	− Large amount of engineering required (model set structure, system/process and observation noise, jump structure and transition probabilities). − Limited expressive power. − Physical model required.

4 The Deep Learning Perspective

In this chapter, the RNN-based solutions for dealing with maneuvering objects are explored. For the exemplary tasks of *path prediction* and *intention prediction*, their behavior with regards to model mismatch is analyzed. Thereby, *path prediction* is mainly used for comparison to related approaches for motion prediction methods due to the fact that there exists a public standard benchmark (section 4.2.1). *Intention prediction*, on the other hand, is well-suited to give a detailed evaluation of the switching behavior (section 4.2.2). The first part of this chapter introduces some theoretical background required for the later proposed deep learning-based filter alternatives. This chapter is partly published in [Bec18c, Bec18b, Bec19a, Bec19b].

4.1 Background

Again, we start with the formalized prediction problem, (see equation 1.1)

$$\mathcal{Y} = f_\theta\left(\mathcal{Z}^{0:k}, \mathcal{C}^{0:k}\right) + \boldsymbol{\epsilon},$$

where $\mathcal{Y}$ describes the future states (or distribution over the states) of a trajectory, $\mathcal{Z}$ are the observations generated by the tracking system, $\mathcal{C}$ additional contextual cues extracted from the observed image sequences, and $\boldsymbol{\epsilon}$ describes an additional error term. As discussed in section 2.3, modeling motion and modeling contextual cues are two different aspects of the motion prediction problem. Without loss of generality, the contextual cues $\mathcal{C}^{0:k}$ are initially left out.

Our aim is to replace the f_θ of a dynamical model with a deep learning solution. In the case of a fixed observation window and realizing the function approximator as a regression problem, learning of the network parameters can be realized by an MLP, a feed-forward neural network, with an appropriate distance function for the predicted trajectory. MLPs are the quintessential deep learning models. The term feed-forward is used because information flows through the computational graph of the network from the input in general or here the fixed-length observed trajectory $\mathcal{Z}^{0:k}$, through the intermediate computations used to define $f(.)$, and finally to the target state $\mathbf{y}$ [Goo16]. Thus, an MLP defines a mapping $f_\theta\left(\mathcal{Z}^{0:k}\right)$ and learns the parameters θ that result in the best function approximation. MLPs have no feedback connections to feed model outputs back into itself. If feedback connections are included, such networks are refereed to as RNN. By drawing the connection between dynamical systems and RNNs in advance (see section 2.2), we motivated the transfer to a deep learning solution. However, MLPs are a conceptual stepping stone on the path to RNNs, which power our proposed IMM alternative and are thus explained in brief.

4.1.1 Multi-Layer Perceptron

MLPs combine several interconnected perceptrons together. Whereby a perceptron is a type of elemental neural unit (neuron) which has an input vector $\mathbf{z} \in \mathbb{R}^{n_z}$ and one scalar output o [Ros58]. The output is $\phi(\cdot)$ applied to the dot product of its inputs and a bias term b

$$o = \phi\left(\mathbf{w}^\mathsf{T}\mathbf{z} + b\right). \tag{4.1}$$

Here, $\mathbf{w} \in \mathbb{R}^{n_z}$ denotes weights for all the inputs, and ϕ a non-linear function referred to as activation function. Commonly used activation functions include the identity function, the sigmoid function, the hyperbolic tangent function, the *ReLU* (Rectified Linear Unit) function, and the *Leaky ReLU* function. By combining several neurons together, MLPs are able to approximate arbitrary non-linear mappings. Thus, an MLP is a feed-forward network of

neuron layers where the neurons of one layer are only connected to the neurons of the previous layer. The output of the ith layer is given by

$$\mathbf{l}_i = \phi\left(\mathbf{W}_i \mathbf{l}_{i-1} + \mathbf{b}_i\right) \tag{4.2}$$

where $\mathbf{l}_0 = \mathbf{z}$ is the input and $\mathbf{l}_N = \mathbf{o}$ being the output of an N-layer MLP. $\mathbf{W}_i \in \mathbb{R}^{n_m \times n_z}$ is the weight matrix of layer i with $\mathbf{W}_i = \left[\mathbf{w}_{i,1}, \cdots \mathbf{w}_{i,m}\right]^\mathsf{T}$ being the combined weight vectors of its m neurons. All weight matrices $\mathbf{W}$ and bias vectors $\mathbf{b}$ are the trainable parameters θ of the network controlling the behavior of f_θ. Figure 4.1 illustrates the structure of an MLP consisting of an input layer, an output layer, and one hidden layer. Due to the pairwise connections of neurons between layers, MLPs are also referred to as *fully connected* layers. The number of parameters quickly rises with an increasing number of neurons in the network. Although the term MLPs is sometimes strictly used for a class of feed-forward networks composed of multiple layers of perceptrons with threshold activation, here the term MLP refers loosely to any feed-forward network without being restricted to particular activation function including radial basis function networks [Bro88].

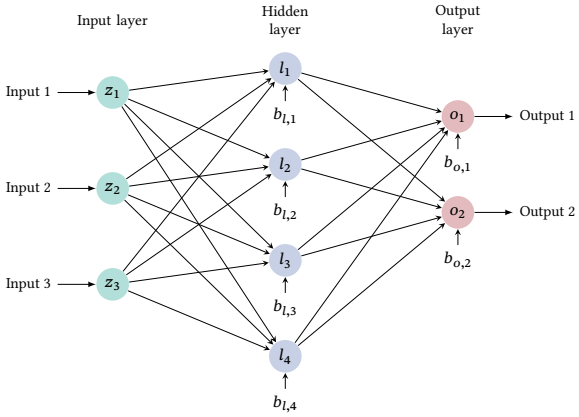

Figure 4.1: Visualization of an MLP. For better clarity, the bias terms are left out.

The parameters of feed-forward neural networks can be efficiently learned with **stochastic gradient descent** (SGD) together with the *backpropagation* [Rum88] algorithm. Backpropagation is an efficient technique of computing the gradients in directed graphs of computations, such as neural networks. The term *backpropagation* is the abbreviation for *backpropagation of errors*, where the errors are defined by the distance function, such as the distance to the path being predicted. Further details on *backpropagation* and optimization methods are given in 4.1.3. An MLP or *fully connected* feed-forward network is designed to have separate parameters for each input feature so that it learns all of the rules of the object motion separately at each position in the trajectory.

4.1.2 Recurrent Neural Networks

In order to extend MLPs for an improved processing of sequential data with variable input length, RNNs share parameters across different parts of a model. As explained in section 2.2, RNNs are extensions of MLPs, where hidden units $\mathcal{H} = \{\mathbf{h}^k : k \in \mathbb{N}\}$ are used to encode an internal latent state space. Unfolding a recurrent computation into a computational graph that has a repetitive structure results in parameter sharing across the network. The unfolded model structure corresponds, similarly to recursive Bayesian filtering, to a directed acyclic computational graph. Thus, the recurrent network processes information by incorporating it into the hidden state that is passed forward in time. As shown in equation 2.7, the hidden state for one time step can be given by

$$\mathbf{h}^{k+1} = f_\Theta\left(\mathbf{h}^k, \mathbf{z}^{k+1}\right). \tag{4.3}$$

A basic RNN [Elm90] can be defined as

$$\mathbf{h}^{k+1} = \phi\left(\mathbf{W}_{hh}\mathbf{h}^k + \mathbf{W}_{zh}\mathbf{z}^{k+1} + \mathbf{b}_h\right) \tag{4.4}$$

$$\mathbf{o}^k = \phi\left(\mathbf{W}_{ho}\mathbf{h}^k + \mathbf{b}_o\right) \tag{4.5}$$

$\mathbf{W}_{(.)}$ represents the weights, $\mathbf{b}_{(.)}$ biases of a recurrent layer, and $\phi(.)$ an activation function. The unfolding process introduces two advantages. Firstly, regardless of the trajectory length the input size is kept fixed. Secondly, it is possible to use the same transition function with the same parameters for every step. A single shared model allows to generalize to trajectory lengths not included in the training set.

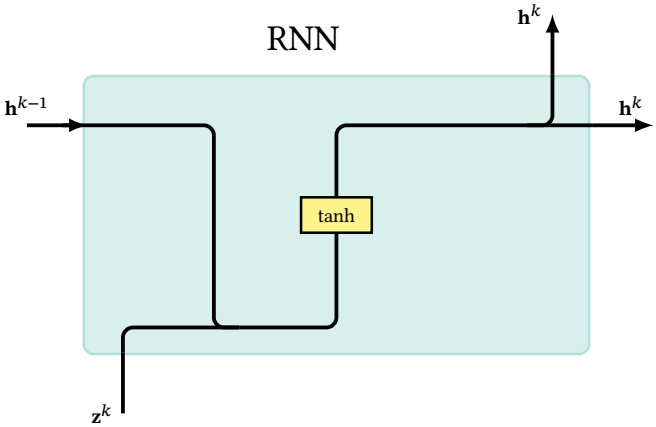

Figure 4.2: Visualization of a standard RNN unit.

The ideas of graph unrolling and parameter sharing enable the design of a wide variety of RNNs [Goo16]. The most effective sequence models used in practical applications are the so-called *gated* RNNs. These include LSTM [Hoc97] and networks based on the GRU [Cho14]. Together with the standard RNN, these variants are used in most of our experiments. The aim of *gated* RNNs is to reduce the effects of exploding and vanishing gradients during parameter learning [Ben93, Pas13]. They rely on the idea of creating a path through time and connecting weights that may change at every time step. In [Gre17], Greff et al. conducted a comparative study between different variants of *gated* RNN architectures for the task of speech recognition, handwriting recognition, and polyphonic music modeling. The results show that none of

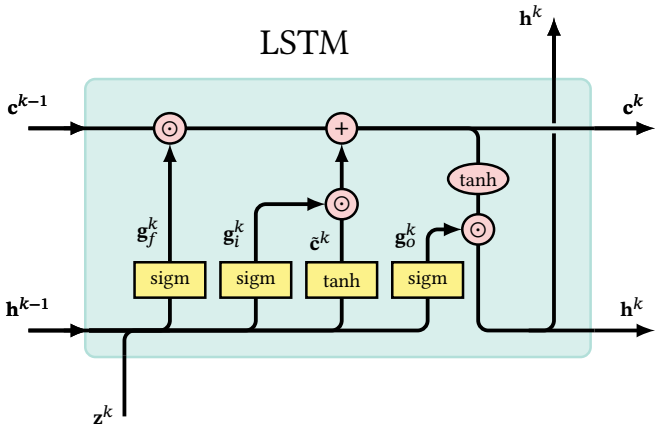

Figure 4.3: Visualization of an **long short-term memory** (LSTM) unit.

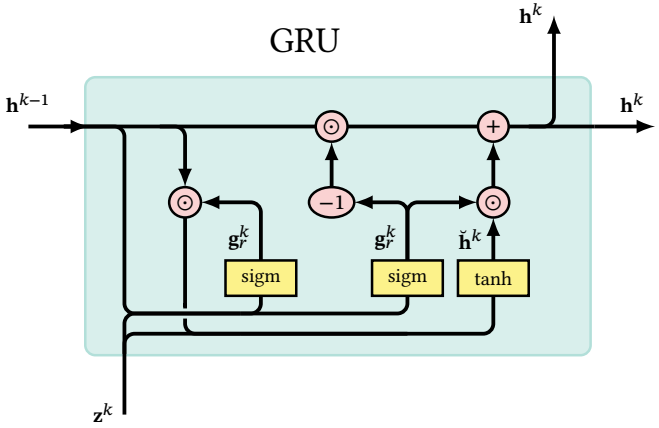

Figure 4.4: Visualization of a **gated recurrent unit** (GRU) unit.

the variants could significantly improve upon the LSTM. The computation of RNNs can be divided into three main blocks of parameters and corresponding transformations. Firstly, from input to hidden state. Secondly, from previous to next hidden state, and thirdly from hidden state to the output. All RNNs

have the form of a chain of repeated modules of neural networks called unit. Thus, an RNN unit includes the main blocks and additional interacting layers or gates depending on the solutions to the vanishing gradient problem. Therefore, gates are realized as linear layers with an activation function and an element-wise operation with the signal. LSTMs have an additional internal recurrence, called *cell* state or *memory cell* $\mathbf{c}$, to the outer recurrence state of the RNN. The introduced self-loop adds a path for the gradients. In addition, some trainable gates control the information added and removed from the *cell* state. The transition equations of an LSTM are given by

$$\begin{aligned}
\mathbf{g}_f^k &= \text{sigm}\left(\mathbf{W}_{hg_f}\mathbf{h}^{k-1} + \mathbf{W}_{zg_f}\mathbf{z}^k + \mathbf{b}_{g_f}\right) \\
\mathbf{g}_i^k &= \text{sigm}\left(\mathbf{W}_{hg_i}\mathbf{h}^{k-1} + \mathbf{W}_{zg_i}\mathbf{z}^k + \mathbf{b}_{g_i}\right) \\
\mathbf{g}_o^k &= \text{sigm}\left(\mathbf{W}_{hg_o}\mathbf{h}^{k-1} + \mathbf{W}_{zg_o}\mathbf{z}^k + \mathbf{b}_{g_o}\right) \\
\mathbf{c}^k &= \mathbf{g}_f^k \odot \mathbf{c}^{k-1} + \mathbf{g}_i^k \odot \tanh\left(\mathbf{W}_{hc}\mathbf{h}^{k-1} + \mathbf{W}_{zc}\mathbf{z}^k + \mathbf{b}_c\right) \qquad (4.6) \\
\mathbf{h}^k &= \mathbf{g}_o^k \odot \tanh(\mathbf{c}^k) \qquad (4.7)
\end{aligned}$$

with the input gate vector $\mathbf{g}_i$, forget gate vector $\mathbf{g}_f$ and output gate vector $\mathbf{g}_o$. The operator $\odot$ denotes the Hadamard product (element-wise product). The forget gate controls how much of the old cell values is used in the new cell value. The amount of input used in the new cell value is controlled by the input gate and the output gate controls how much of the new cell value is put out. In a GRU, the LSTM cell is simplified by combining forget and input gates into a single update gate and merging the cell and hidden states. This results in fewer trainable parameters with comparable performance in specific tasks. The GRU architecture is given by

$$\begin{aligned}
\mathbf{g}_u^k &= \text{sigm}\left(\mathbf{W}_{hg_f}\mathbf{h}^{k-1} + \mathbf{W}_{zg_f}\mathbf{z}^k + \mathbf{b}_{g_f}\right) \\
\mathbf{g}_r^k &= \text{sigm}\left(\mathbf{W}_{hg_i}\mathbf{h}^{k-1} + \mathbf{W}_{zg_i}\mathbf{z}^k + \mathbf{b}_{g_i}\right) \\
\check{\mathbf{h}}^k &= \tanh\left(\mathbf{W}_{z\check{h}}\mathbf{z}^k + \mathbf{W}_{\check{g}_r\check{\mathbf{h}}}\left(\mathbf{g}_r^k \odot \mathbf{h}^{k-1}\right) + \mathbf{b}_{\check{\mathbf{h}}}\right) \\
\mathbf{h}^k &= (1 - \mathbf{g}_u^k) \odot \mathbf{h}^{k-1} + \mathbf{g}_u^k \odot \check{\mathbf{h}}^k \qquad (4.8)
\end{aligned}$$

where $\mathbf{g}_u^k$ is the update gate vector and $\mathbf{g}_r^k$ is the reset gate vector. The GRU tackles the vanishing gradient problem without depending on an internal cell state. Thereby, the reset gate controls how much of the old output is kept as new input, and the update gate controls whether to use the new input or old output. For more information on different variants of RNNs, we refer to the following works [Gra13b, Gre17, Goo16]. The block diagrams of the repeating units of an RNN, an LSTM, and a GRU are illustrated in figures 4.2, 4.3, and 4.4. [1]

4.1.3 Training

As shown in section 2.2, the unfolded structure of an RNN corresponds to a directed acyclic computational graph. The graph maps the input trajectory $\mathcal{Z} = \{\mathbf{z}^k : k \in \mathbb{N}\}$ (or generally input sequence) to a corresponding sequence of outputs $\mathcal{O} = \{\mathbf{o}^k : k \in \mathbb{N}\}$. Thus, it provides an explicit description of which computation to perform and maps the input trajectory to outputs and losses. By information flow forward in time, the outputs and losses are computed. The parameter gradients are computed backward in time. Figure 2.4 depicts the unfolded computational graph of a vanilla RNN. Due to the fact that RNNs are composed of differentiable operators, training the parameters can be done by minimizing any differentiable loss function $\mathcal{L}(\boldsymbol{\Theta})$ using gradient descent. The basic update cycle of gradient descent is to find the derivative of the loss function with respect to network parameters $\boldsymbol{\Theta}$, then adjust the weights in the direction of the negative slope (parameter update). In section 2.2, we demonstrated that the loss function of RNNs corresponds to maximum likelihood estimation with deterministic dynamics. Further, the loss function corresponding to the probability given by the network to the observed sequence $\mathcal{Z} = \{\mathbf{z}^k : k \in \mathbb{N}\}$ is given by

$$\mathcal{L}(\boldsymbol{\Theta})_{RNN} = -\sum_k \log p(\mathbf{z}^k | \mathbf{o}^{k-1}). \tag{4.9}$$

[1] Visualization inspired by *Understanding LSTM Networks* (https://colah.github.io/posts/2015-08-Understanding-LSTMs/ last accessed 19.12.2019.).

The choice of a loss function is directly related to the design of the output layer. One can think of the design of the output layer about framing the prediction problem and the choice of the loss function corresponds to the way of calculating the error. Thus, the output $\mathbf{o}^k$ can be used to parameterize a predictive distribution $p(\mathbf{z}^{k+1}|\mathbf{o}^k)$ over the possible next observation $\mathbf{z}^{k+1}$. Due to the deterministic nature of RNNs, the computation of the predictive distributions is realized by the feed-forward operations in the unfolded network. Accordingly, the training of RNNs can be done similarly to feed-forward networks by applying gradient descent methods to minimize a differentiable loss function $\mathcal{L}(\boldsymbol{\Theta})$. The generalization of *backpropagation* for recurrent networks is called **backpropagation through time** (BPTT) [Wil95]. As explained briefly, *backpropagation* is a technique to efficiently calculate the gradients of scalar valued functions with respect to their inputs. It boils down to a recursive application of the chain rule from calculus for the partial derivatives. In order to perform a parameter update, the gradients of the loss function $\nabla_{\boldsymbol{\Theta}^k}\mathcal{L}(\boldsymbol{\Theta})$ with respect to the parameters are required. The same process enables a computation of the gradients for the inputs. By applying the chain rule, evaluating the gradients of the output with respect to the inputs reduces to a product of Jacobian matrices which is the final gradient. In the *forward pass* all intermediate values, corresponding to all intermediate transformations, and the loss for a given set of training samples and the current parameters is computed. Thus, inputs and loss function take on specific values using fixed functions. The backward pass, *backpropagation*, proceeds in the reverse order through all intermediate stages by applying the chain rule to estimate the influence of local gradients on the final output. Thus, the gradients are recursively chained through the functions that produced the values in the forward pass until the inputs are reached. For neural networks, the inputs of interest are the network parameters and their gradients provide information on how to change the current parameters for minimizing the expected loss [Kar16b].

The basic update rule of gradient descent as an iterative optimization algorithm can be written as

$$\boldsymbol{\Theta}^{k+1} \triangleq \boldsymbol{\Theta}^{k} - \lambda_{lr} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}^{k}} = \boldsymbol{\Theta}^{k} - \lambda_{lr} \nabla_{\boldsymbol{\Theta}^{k}} \mathcal{L}(\boldsymbol{\Theta}), \tag{4.10}$$

where λ_{lr} is the learning rate of the neural network that determines the magnitude of the parameter change $\Delta\boldsymbol{\Theta}^{k}$. The index k refers to the time before and $k+1$ to the time after parameter update. Designing and training a neural network is not much different from training any other machine learning model with gradient descent. In contrast to standard gradient descent, where the gradient is calculated from the entire training dataset, **stochastic gradient descent** (SGD) performs a parameter update for a randomly selected subset of training samples. Sometimes, there is a minor distinction for the terminology of SGD. SGD is used for a parameter update for every data sample, and *mini-batch gradient descent* for a parameter update for every mini-batch of training samples. However, most algorithms for deep learning use more than one but less than all training samples, here for the sake of simplicity, only the term SGD is used [Goo16].

Despite SGD as a stochastic approximation of standard gradient descent, there exist many further modifications to overcome challenges of gradient descent-based optimization strategies such as over-fitting, slow convergence, and local minima. Due to the fact that second-order optimization methods, such as Newton's method, require calculation of the Hessian matrix, in practice, almost exclusively first-order algorithms based on gradient descent are used. In the following, we briefly outline commonly used modifications, but leave out second-order alternatives.

The *momentum* method [Pol64] is designed to accelerate SGD, especially in the face of high curvature, small but consistent gradients, or noisy gradients. The *momentum* method accumulates an exponentially decaying moving average of past gradients and continues to move in their direction. Thereby, *momentum* damps oscillation and speeds up convergence in cases where the surface of the loss function is in one dimension much steeper than in the other.

Formally, the *momentum* method changes the parameter update by introducing a velocity term that captures the direction and speed at which the parameters move through parameter space. The adapted parameter update is given by

$$\begin{aligned} \boldsymbol{\Theta}^{k+1} &= \boldsymbol{\Theta}^{k} - \mathbf{v}_{\boldsymbol{\Theta}}^{k} \quad \text{, with} \\ \mathbf{v}_{\boldsymbol{\Theta}}^{k} &= \lambda_{mo}\mathbf{v}_{\boldsymbol{\Theta}}^{k-1} + \lambda_{lr}\nabla_{\boldsymbol{\Theta}^{k}}\mathcal{L}(\boldsymbol{\Theta}). \end{aligned} \tag{4.11}$$

The added hyper-parameter $\lambda_{mo} \in [0,1]$ is termed *momentum* and controls the influence of previous update values. Common values of λ_{mo} used in practice include 0.5, 0.9, and 0.99 or adapting λ_{mo} over time by starting from a small value and later increasing it. Rumelhart et al. [Rum86] showed that using a *momentum* term dramatically increases the convergence rate.

Nesterov's accelerated gradient [Nes83] adds a correction factor to the standard *momentum* method by evaluating the gradient after a one step prediction in the current direction. This update rule can be expressed as follows

$$\begin{aligned} \boldsymbol{\Theta}^{k+1} &= \boldsymbol{\Theta}^{k} - \mathbf{v}_{\boldsymbol{\Theta}}^{k} \quad \text{, with} \\ \mathbf{v}_{\boldsymbol{\Theta}}^{k} &= \lambda_{mo}\mathbf{v}_{\boldsymbol{\Theta}}^{k-1} + \lambda_{lr}\nabla_{\tilde{\boldsymbol{\Theta}}^{k}}\mathcal{L}(\boldsymbol{\Theta} - \mathbf{v}_{\boldsymbol{\Theta}}^{k-1}). \end{aligned} \tag{4.12}$$

This minor change intends to slow down earlier and reduce overshooting. In [Ben13], Bengio et al. showed that the resulting increased responsiveness helps to improve the performance of RNNs for a number of tasks. The *momentum* term helps to speed up SGD and adapts the parameter update with respect to the slope of the error function but to the expense of introducing another hyper-parameter. One alternative is to adapt the learning rate by applying a pre-defined schedule. A representative function is the exponential learning rate decay. Whereas the *momentum* term helps to speed up SGD and adapts the parameter update with respect to the slope of the error function, it is also possible to allow for individual parameter updates depending on their importance.

A heuristic approach to adapting individual learning rates for model parameters during training is the delta-bar-delta algorithm [Jac88], but it is not applicable to SGD optimization. Examples for this category of algorithms with a per-parameter learning rate method to perform more informative gradient-based learning are AdaGrad [Duc11], RMSprop [Tie12], Adadelta [Zei12], and Adam [Kin15].

The AdaGrad algorithm adjusts the learning rate individually by scaling them inversely proportional to the squared sum of all past gradients. AdaGrad performs well for some deep learning models, but the main weakness is that the accumulation of squared gradients from the beginning of training can result in an early and aggressive decrease in the effective learning rate [Goo16].

RMSprop and Adadelta are both extensions that help to reduce the effect of an aggressively decreasing learning rate. Contrary to AdaGrad, the changing gradient accumulation is replaced by an exponentially weighted moving average of the squared gradients. RMSprop is actually a special case of Adadelta with weight decay factor of $\lambda_{ad} = 0.9$ for the gradients. The parameter update can be described by

$$\boldsymbol{\Theta}^{k+1} = \boldsymbol{\Theta}^{k} - \lambda_{lr} \frac{\nabla_{\boldsymbol{\Theta}^k} \mathcal{L}(\boldsymbol{\Theta})}{\sqrt{\mathbb{E}[\nabla_{\boldsymbol{\Theta}^k} \mathcal{L}(\boldsymbol{\Theta}) \odot \nabla_{\boldsymbol{\Theta}^k} \mathcal{L}(\boldsymbol{\Theta}) + \epsilon_{Ada}]}},$$

with

$$\begin{aligned} \mathbb{E}[\nabla_{\boldsymbol{\Theta}^k} \mathcal{L}(\boldsymbol{\Theta}) \odot \nabla_{\boldsymbol{\Theta}^k} \mathcal{L}(\boldsymbol{\Theta})] = \lambda_{ad}\, \mathbb{E}[\nabla_{\boldsymbol{\Theta}^{k-1}} \mathcal{L}(\boldsymbol{\Theta}) \odot \nabla_{\boldsymbol{\Theta}^{k-1}} \mathcal{L}(\boldsymbol{\Theta})] \\ + (1 - \lambda_{ad}) \nabla_{\boldsymbol{\Theta}^k} \mathcal{L}(\boldsymbol{\Theta}) \odot \nabla_{\boldsymbol{\Theta}^k} \mathcal{L}(\boldsymbol{\Theta}). \end{aligned} \tag{4.13}$$

Here, ϵ_{Ada} is a smoothing term to avoid division by zero. The learning rate is divided by an exponentially decaying average of squared gradients. Tieleman and Hinton [Tie12] suggest to set the default value of the learning rate to 0.001.

Another optimization method that computes adaptive learning rates for each parameter is **adaptive moment estimation** (ADAM) [Kin15]. ADAM combines Adadelta and *momentum* by storing both the exponentially decaying

average of past squared gradients, like Adadelta, and exponentially decaying average of past gradients, like *momentum*. These are the first and second moment of the gradients respectively. Further, Adam counteracts biases during initialization in the first-order moments (*momentum*) and the (uncentered) second-order moments by corrections factors. With $\lambda_{bc,1}$, $\lambda_{bc,2}$ being the bias correction factors for the first and second moment estimates, the ADAM update rule is given by

$$\boldsymbol{\Theta}^{k+1} = \boldsymbol{\Theta}^{k} - \lambda_{lr} \frac{1}{\sqrt{\frac{\mathbb{E}[\nabla_{\boldsymbol{\Theta}^k}\mathcal{L}(\boldsymbol{\Theta}) \odot \nabla_{\boldsymbol{\Theta}^k}\mathcal{L}(\boldsymbol{\Theta})]}{(1-\lambda_{bc,2})}} + \epsilon_{ADAM}} \frac{\mathbb{E}[\nabla_{\boldsymbol{\Theta}^k}\mathcal{L}(\boldsymbol{\Theta})]}{(1-\lambda_{bc,1})}. \tag{4.14}$$

Kingma and Ba suggest to use $\lambda_{lr} = 0.002$, $\lambda_{bc,1} = 0.9$ and $\lambda_{bc,2} = 0.999$ as default parameters. Adadelta and RMSprop also incorporate an estimate of the second-order moment but lack the correction factor. Thus, the estimates may have high biases in early stages of the training. In general, ADAM is considered to be relatively robust to the choice of hyper-parameters. Nevertheless, no single best algorithm has emerged. In [Sch14], Schaul et al. conducted a comparative study on a large number of optimization algorithms across a wide range of learning tasks. The results show that the presented algorithms with adaptive learning rate perform fairly robustly, but without clear-cut best algorithm. Despite some further extension, such as Nadam [Doz16] and AMSGrad [Red18], the most actively used optimization algorithms are standard SGD, SGD with *momentum*, RMSProp, RMSProp with *momentum*, AdaDelta, and ADAM. Although ADAM and its extensions might be the best overall choices, SGD is much more reliant on a robust initialization and annealing schedule than other optimizers [Rud16].

4.1.4 Mixture Density Networks

So far, we introduced some theoretical background related to the recurrent units and methods for training the networks. One remaining central question is how to parametrize the predictive distribution $p(\mathbf{z}^k|\mathbf{o}^{k-1})$. Similar to

Bayesian filtering, the conditional probability density should allow to predict observations and has to deal with real-valued inputs from the observed trajectory. The idea of **mixture density networks** (MDNs) [Bis94, Bis06] is to use the outputs of a neural network to parameterize a Gaussian mixture distribution. MDNs provide a complete framework for modeling conditional density functions. They overcome the limitations of conventional least-square approaches for dealing with multi-modal target data. Since the selected tasks of *path* and *intention prediction* are multi-modal problems by nature and MDNs can be used with RNNs [Sch00], our basic neural network to infer object states consists of an RNN with an MDN on top. This combination was originally introduced by Graves for the generation and prediction of handwriting [Gra13a]. It is subsequently referred to as an RNN-MDN model or respectively as an LSTM-MDN model.

Consider a *path prediction* problem where $\mathbf{y}^{k+1} = p(\mathbf{z}^{k+1}|\mathbf{z}^{0:k})$ describes the next location and $\mathbf{z}^{0:k}$ are the corresponding observations leading to output $\mathbf{o}^k$. The conditional probability for a bi-variate Gaussian mixture is defined as follows:

$$p(\mathbf{z}^{k+1}|\mathbf{o}^k) = \sum_{l=1}^{L} w_l^k \mathcal{N}(\mathbf{z}^{k+1}|\boldsymbol{\mu}_l^k(\mathbf{o}^k), \boldsymbol{\sigma}_l^k(\mathbf{o}^k), \rho_l^k(\mathbf{o}^k)). \tag{4.15}$$

Here, $\mathbf{o}^k$ is used to parametrize the L component Gaussian mixture model $\mathbf{o}^k = (\boldsymbol{\mu}_l^k, \boldsymbol{\sigma}_l^k, \rho_l^k, w_l^k)_{l=1}^L$. In order to reduce notation clutter, the dependency on the network output will only be denoted in case it is not clear. In an MDN, the same function is used to predict the parameters of all of the densities components as well as the mixing coefficients. So, the non-linear hidden units are shared amongst the input-dependent functions. For a mixture of bi-variate Gaussians with L components, the network output generates $6 \cdot L$ parameters where mean and standard deviation are two-dimensional vectors, whereas the mixture weights and correlations are scalars. In order to ensure that the mixture density forms a valid categorical distribution, the weights are normalized

with a softmax activation function:

$$w_j^k = \frac{\exp \hat{w}_j^k}{\sum_{l=1}^{L} \exp \hat{w}_l^k}. \tag{4.16}$$

The softmax function ensures that w_j^k lie in the required range $w_j^k \in (0,1)$ and $\sum_{l=1}^{L} w_l^k = 1$. It realizes a generalization of the Bernoulli distribution corresponding to the usual logistic sigmoid. The variances can be represented in terms of the exponential of the corresponding network [Bis94, Gra13a]. The originally proposed exponential can lead to numerical instability, thus we employ a variant of the **exponential linear unit** (ELU) activation function [Cle16]. The transformation function is given by

$$\sigma_j^k = \begin{cases} \hat{\sigma}_j^k + 1 & \text{for} \quad \hat{\sigma}_j^k > 0 \\ \lambda_{elu}(\exp \hat{\sigma}_j^k - 1) + 1 & \text{for} \quad \hat{\sigma}_j^k \leq 0 \end{cases}. \tag{4.17}$$

Both, the originally proposed exponential and the ELU variant avoid configurations with variances which go to zero. As an alternative, also a softplus activation function can be used [Glo11]. The means represent location parameters and can be represented directly with the network output as

$$\mu_j^k = \hat{\mu}_j^k. \tag{4.18}$$

Ensuring that the correlation coefficients lies in range $\rho_l^k \in (-1,1)$ is realized with a hyperbolic tangent activation function. The parameters of the MDN can be determined by maximum likelihood estimation, or equivalently by minimizing an error function defined to be the negative logarithm of the likelihood.

Substitute equation 4.15 into 4.9 yields the sequence loss

$$\begin{aligned}\mathcal{L}(\mathcal{Z},\Theta)_{RNN} &= \sum_{k=1}^{K} -\log\left(\sum_{l=1}^{L} w_l^k \mathcal{N}(\mathbf{z}^{k+1}|\boldsymbol{\mu}_l^k, \boldsymbol{\sigma}_l^k, \rho_l^k)\right) \\ &= \sum_{k=1}^{K} -\log\left(\sum_{l=1}^{L} w_l^k \frac{1}{2\pi\sigma_{l,1}^k\sigma_{l,2}^k\sqrt{1-{\rho_l^k}^2}} \exp\left(\frac{-Z_l^k}{2(1-{\rho_l^k}^2)}\right)\right),\end{aligned}$$

with

$$Z_l^k = \frac{(x_{1,l}^k - \mu_{1,l}^k)^2}{{\sigma_{1,l}^k}^2} + \frac{(x_{2,l}^k - \mu_{2,l}^k)^2}{{\sigma_{2,l}^k}^2} - \frac{2\rho_l^k(x_{1,l}^k - \mu_{1,l}^k)(x_{2,l}^k - \mu_{2,l}^k)}{\sigma_{1,l}^k\sigma_{2,l}^k}. \tag{4.19}$$

As a technical detail, the loss function is arranged, such that the *log-sum-exp*-trick [Pre07] can be applied. Thus, improving numerical stability:

$$\begin{aligned}\mathcal{L}(\mathcal{Z},\Theta)_{RNN} &= \sum_{k=1}^{K} -\log\left(\sum_{l=1}^{L} w_l^k \mathcal{N}(\mathbf{z}^{k+1}|\boldsymbol{\mu}_l^k, \boldsymbol{\sigma}_l^k, \rho_l^k)\right) \\ &= \sum_{k=1}^{K} -\log\left(\sum_{l=1}^{L} \exp\left(\log\left(w_l^k \mathcal{N}(\mathbf{z}^{k+1}|\boldsymbol{\mu}_l^k, \boldsymbol{\sigma}_l^k, \rho_l^k)\right)\right)\right) \\ &= \sum_{k=1}^{K} -\log\left(\sum_{l=1}^{L} \exp\left(\log\left(w_l^k\right)\right.\right. \\ &\quad \left.\left. - \log\left(2\pi\sigma_{1,l}^k\sigma_{2,l}^k\sqrt{1-{\rho_l^k}^2}\right) - \frac{Z_l^k}{2(1-{\rho_l^k}^2)}\right)\right).\end{aligned} \tag{4.20}$$

As explained, RNN-MDN variants are commonly an adaptation of the model introduced by Graves [Gra13a]. Especially in the context of sequence prediction, including *path prediction*, these models are preferred to other neural networks which can also generate a probabilistic distribution over the outputs. Popular alternatives mostly rely on Bayesian neural networks [Mac92, Bis95] or their recurrent extension [For17]. Instead of placing a distribution

over the output of the model, Bayesian neural networks use probabilistic neurons. Similar to models discussed in section 3.1, Bayesian inference has to be applied during training in order to determine the posterior distribution of the network parameters. Due to the intractable probability distributions, either Monte Carlo methods or approximate inference methods, such as variational inference [Blu15], inference based on expectation propagation [Her15], and Monte Carlo dropout [Gal16], has to be applied. The requirement of approximate inference make training computationally more intensive and potentially less stable. Hence, despite their ability to allow probabilistic predictions and provide information about model uncertainty, these models are less widely used in practice [Hug19]. Compared to Bayesian neural networks or respectively Bayesian RNNs, RNN-MDNs have a simple structure and are thus easier to train and control. Hence, RNN-MDNs have been not only successfully applied to model handwriting data [Gra13a], but also to model sketch drawings [Ha18], speech synthesis [Wan17], and, more importantly, in the context of this thesis to generate trajectory predictions [Vem18, Ala16, Zha19, Xue19].

Due to the capabilities of RNNs to model arbitrary functions and the success of the RNN-MDN in a variety of sequence processing tasks, we use RNN-MDN as basic architecture for our *pattern*-based solutions to model object trajectories and additionally capture the predictive distribution.

4.2 RNN-based Solutions

In this chapter, the ability of RNN-based solutions to deal with maneuvering objects is analyzed. Thereby, we distinguish between the two types of maneuvers which are normally tackled with *multiple-model* approaches. These maneuver types are the switch in noise levels and the switch in motion behavior. They are considered separately. The analysis is done for the two selected tasks of *path* and *intention prediction*. For both tasks, higher-level processing strongly relies on the state estimation performance. The approaches are realized as a top-down component as part of a vision-based tracking system.

Due to cross-disciplinary interest, as shown in section 2.3, there exists a fast-growing amount of different approaches. The requirements on the performance quality depend on the application domain and particular use cases within. Due to the wide variety of applications, different levels of *contextual* cues, and the amount of existing diverse methods, a standardized benchmarking is difficult to achieve. Here, *path prediction* is used for comparison to related approaches for motion prediction due to the fact that there exists a public standard benchmark, and the corresponding data includes variation in the noise levels. *Intention prediction* is selected for capturing scenarios within the application domain of intelligent vehicles to evaluate the ability of the proposed solutions with respect to the switching dynamics of objects. For specific use-cases, such as pedestrian crossing, there are not only strict requirements on the prediction horizons, but in addition, there are suited solutions for physically modeling the pedestrian motion. As discussed in section 2.3, the essential difference is the short time window for prediction, and thus the more dominant role of *physics*-based *multiple-model* approaches.

4.2.1 Path Prediction

Although the integration of more *contextual* cues can be crucial to improve motion prediction and *pattern*-based methods can theoretically capture all contextual cues present in the training data, we only rely on the information provided by an underlying object tracker. For *path prediction*, this is a sequence of past positions in order to infer future positions. Firstly, the *physics*-based traditional alternatives, which we aim to replace, also solely rely on the observed object states. Secondly, not only the incorporation of contextual cues lead to very complex algorithms for *physics*-based methods, but also the design and training of the network get more complicated, e.g., due to the increased dimension of input data.

However, even without additional cues, there are many pitfalls when using neural network-based alternatives for *path prediction*. In the following our analysis reveals failure cases and gives explanations for observed phenomena. Further, we provide recommendations for overcoming shortcomings which

enabled our proposed solution to achieve top-rank at the public *TrajNet* 2018 challenge. Our aim is to achieve a more reliable prediction network. Despite the simple core architecture, the network can achieve a performance comparable to more elaborated models with regards to considering more cues than solely position information.

Achieving the objective of finding an effective prediction network involves, on the one hand, evaluation of different deep neural networks and, on the other hand, an analysis of the properties of the dataset. The dataset analysis directly motivates the required modification to enable a robust prediction.

4.2.1.1 Dataset Analysis

Although many datasets for *path prediction* are publicly available, the *TrajNet* [Sad18] benchmark and the corresponding challenge is the first attempt to build a standard benchmark for *path prediction* and provides a platform for comparison. The challenge is in particular called the *world plane human-human TrajNet* challenge (*World H-H TrajNet*). The *TrajNet* dataset is a superset of the commonly used surveillance datasets which cover real-world scenarios with varying crowd densities and varying complexity of trajectory patterns. In most datasets, the scene is observed from a bird's eye view, but there are also scenarios where the scene is observed under a higher depression angle. Details of the datasets are summarized in table 4.1 (adapted from *TrajNet* website). The selection includes the following datasets. The *BIWI* dataset [Pel09] also referenced to as *ETH Walking Pedestrians*, which is split into two sets (*ETH* and *Hotel*). The *UCY* dataset also referred to as *Crowds-by-Example* dataset [Ler07] contains three scenes from an oblique view, where the first (*Zara*) shows a part of a shopping street, the second (*Students/Uni Examples*) captures a part of the university campus, and the third scene (*Arxiepiskopi*) captures a different part of the campus. Then, the *Stanford Drone Dataset (SDD)* [Rob16] consists of multiple aerial images capturing different locations around the Stanford campus. Furthermore, the *PETS 2009* dataset [Fer09], where different outdoor activities of crowds are observed by multiple

static cameras. Sample images with full trajectories and tracklets are shown in figure 4.5. The term tracklet refers to a part of a longer trajectory.

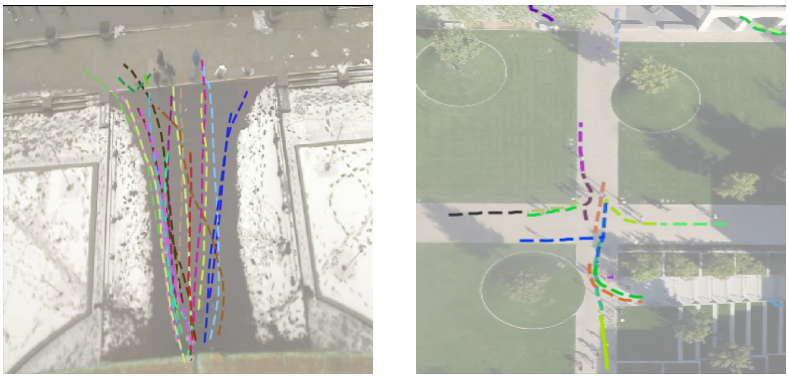

Figure 4.5: Example trajectories from the *BIWI ETH* dataset and example tracklets from the sequence *Hyang_07* from the *Stanford Drone Dataset (SDD)*.

It should be noted that for the *World H-H TrajNet* challenge, *path prediction* is performed on ground level in a world coordinate system with available cues from the *dynamic environment* in form of observed trajectories from other dynamic objects in the scene (human-human). Hence, contextual cues are implicitly used to realize a mapping from the image space to a 3D reference system. Since the scenarios capture static surveillance scenes, this is realized under a flat world assumption and by estimation of individual homographies for every scene. In accordance with our requirements, *path prediction* relies for the challenge on position data provided by an underlying system. In particular, the ground truth trajectories are generated by a visual tracker or are manually annotated. It is common and good practice to apply cross-validation. For the *TrajNet* challenge, this is done by omitting complete datasets for testing. This is reasonable, given the fact that the interaction behavior of humans in open spaces is scene-independent and in order to measure the generalization capabilities of various approaches across datasets.

Table 4.1: Training (gray ▭) and test (blue ▭) dataset of the world plane human-human dataset challenge (adapted from the *TrajNet* website[a] [Sad18]).

Name	Resolution	#Pedestrians	Frame rate / fps	Reference
BIWI Hotel	720 × 576	389	2.5	[Pel09]
UCY Zara	720 × 576	204	2.5	[Ler07]
UCY Students	720 × 576	415	2.5	[Ler07]
UCY Arxiepiskopi	720 × 576	24	2.5	[Ler07]
PETS 2009	768 × 576	19	2.5	[Fer09]
Stanford Drone Dataset (SDD)	595 × 326	3295	2.5	[Rob16]
BIWI ETH	640 × 480	360	2.5	[Pel09]
UCY Zara	720 × 576	148	2.5	[Ler07]
UCY Uni Examples	720 × 576	118	2.5	[Ler07]
Stanford Drone Dataset (SDD)	595 × 326	3297	2.5	[Rob16]

[a] *TrajNet* website: (http://trajnet.stanford.edu/, last accessed 19.12.2019)

Nevertheless, by combining all training sets the spatial context of scene-specific motion and the reference systems are lost. When only relying on observed motion trajectories, positional information is crucial in order to learn spatio-temporal variation. For example, the sidewalks in the *Hyang* sequences (see figure 4.5) lead to a spatial-dependent change in the curvature of a trajectory. Since our focus is on deep neural networks including RNNs, shifting from position information to offsets helps to overcome some drawbacks. Before RNNs were successfully applied for tracking pedestrians in a surveillance scenario, they gained attention due to their success in tasks like speech recognition [Gra13c, Chu15] and caption generation [Don15, Xu15]. Since these domains are particularly different from trajectory prediction in certain aspects, their position-dependent movement is not important. Accordingly, RNNs can benefit from conditioning on offsets, instead of absolute positions, for scene-independent motion prediction. This insight is not new, yet utilizing offsets helps not only to stabilize the learning process but also to improves the prediction performance for the evaluated networks. This shift to offsets or rather velocities has also been successfully applied for example for the prediction of human poses based on RNNs [Mar17]. In the context of deep networks, the same effect can also be achieved by adding residual connections, which have been shown to improve performance on

deep convolutional networks [He16]. Presumably due to the limitation of the input and output spaces, for applying on the *TrajNet* challenge instead of predicting the next position (where will the person be next) predicting the following offsets (where will the person go next) [Hug17, Hug18] also contributed to increased prediction accuracy. This becomes immediately apparent by looking at the complete tracklets of the training and test set (see figure 4.6). Firstly, it takes a considerably higher modeling effort to represent all possible positions instead of modeling particular velocities. Further, input data outside the training range can lead to undefined states in the deep network, which results in an unreasonable output. Some of the initialization tracklets clearly lie outside the training input space. Also, approaches which benefit from human-human interaction such as [Gup18, Has18, Ale17, Ala16] in combination with deep networks lack at this point information about surrounding persons to interact with, so that the decoding of relative distances is not possible because of a reduced person density. Note that the ability of RNN-based solutions to capture *environment* cues from position data only is per se a positive ability, but they require a sufficient amount of training data without too large gaps in the input data. This is further discussed in the subsequent sections.

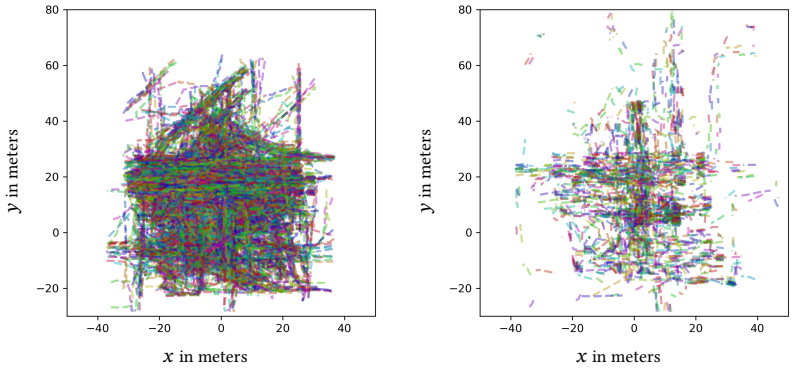

Figure 4.6: (Left) Visualization of all tracklets of the training set from the *TrajNet* dataset collection. (Right) Visualization of all initialization tracklets of the test set.

Another factor for improving prediction performance is becoming apparent when contemplating the offset distribution of the data. Figure 4.7 shows the offsets histograms for x and y separately. Due to the loss of the reference system, it is impossible to assume a reasonable location distribution prior. In contrast, the offset and magnitude distribution clearly reflects the preferred walking speeds in the data. The histograms also show that a large number of persons are standing. In the recent work of Hasan et al. [Has18], it was emphasized that forecasting errors are in general higher when the speed of persons is lower and argued that when persons are walking slowly, their behavior becomes less predictable, due to physical reasons (less inertia). During our testing, we discovered the same phenomenon. In particular, RNN-based networks tend to overestimate slow velocities and do sometimes not accurately identify the standing behavior. Despite this problem, the range of offsets is very limited compared to the location distribution and shows a clear tendency towards expected prior values. Common techniques for sequence prediction problems are normalization and standardization of the input data. Whereby normalization has a similar role on the position data, applying standardization on position input data shows no benefit. In our experiments, standardization worked slightly better than normalization or an embedding layer for input encoding. Although the effect on the performance is quite low for the *TrajNet* challenge, our best result is achieved using standardized offsets as input. It is rarely necessary to standardize the inputs, but there are practical reasons like accelerating the training or reducing the chances of getting stuck in local optima [Bro17]. Predicting offsets also guarantees that the output directly conforms better with the range of common activation functions. Through standardization of the offsets, the network uses the deviations from the preferred pedestrian walking behavior to predict changes in their behavior.

Without discretization artifacts, the dynamics of humans are smooth and persistent. The trajectory data from the *TrajNet* dataset includes varying discretization artifacts or noise levels resulting from different methods with which ground truth data was generated. As explained, part of the ground truth trajectories are generated by a visual tracker. For approximating the amount of noise in the datasets, the distance between a smoothed spline fit

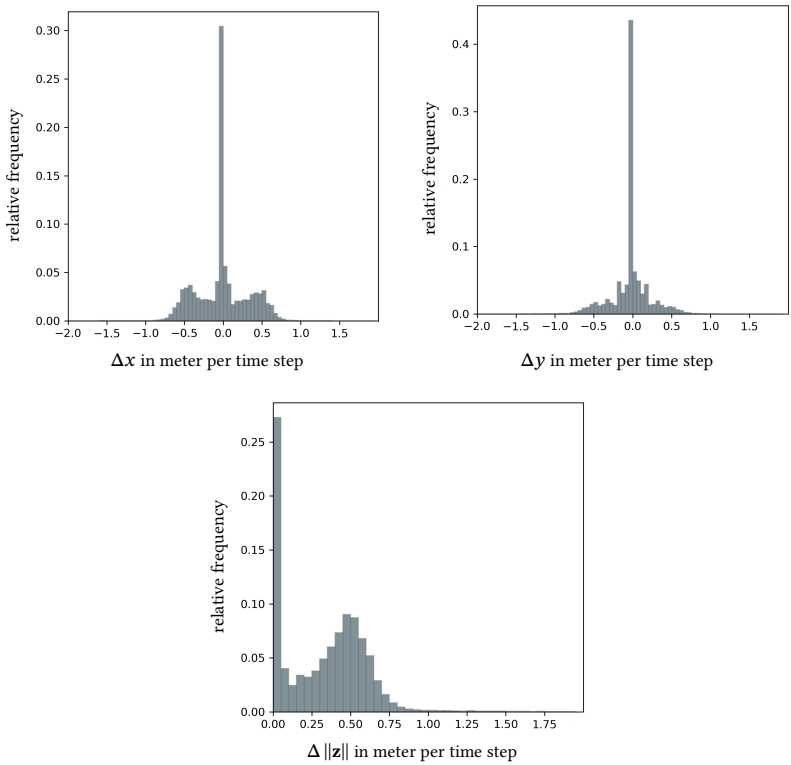

Figure 4.7: (Top left, Top Right) Offset histograms of the training set. (Bottom) Magnitude histogram of the offsets.

through the complete tracklets is compared to the provided ground truth tracklet points. The spline fitting is done with a polynomial function with a varying degree (1,2,3,4) independent for the x and y values. By selecting the individual best fit in regards to the mean squared error for a single trajectory fit, over- and under-fitting is prevented. The remaining error is used to approximate the noise for single trajectories. Despite the fact that using a fitted trajectory as pre-processed ground truth can also induce some errors, the fitted trajectories capture the continuous, persistent motion better.

Further, the complete history or rather full trajectories are considered instead of only short time windows which makes the fit more robust. Nevertheless, the achieved fitted trajectories form a smooth and natural path and are used as rough assessment for the noise levels in the ground truth trajectory data. The results for the training set are summarized in table 4.2.

The approximated noise levels show the variation in the ground truth data. In order to outperform a linear baseline predictor, the learned model must be able to successfully model different velocity profiles and capture curved paths out of input data with different noise levels. Initial experiments to solely train on smoothed fitted trajectories with synthetic noise performed worse. Nevertheless, for the prediction of future steps, the best performing predictor is trained to forecast smoothed paths. Before the different evaluated models are introduced, the last data analysis of the training set is intended to assess the complexity in terms of the non-linearity of the trajectories. Therefore, the coefficient of determination R^2 for linear interpolation is calculated separately for the x and y values. This linear interpolation serves as a baseline predictor for the *TrajNet* challenge. The histograms of R^2 for the training set are shown in figure 4.8. R^2 is the percentage of the variation that is explained by the model and is used to determine the suitability of the regression fit as a linearity measure [Dra66]. The average $\bar{R}^2$ values are summarized in table 4.2. It can be seen that for most tracklets, a linear interpolation works well. In order to outperform the linear interpolation baseline, it is crucial to not only cover a variety of complex observed motions but to also produce robust results in simpler situations. As shown, the person's velocity has to be effectively captured by the model.

The analysis of the *TrajNet* dataset shows that the main challenges to achieve a good, robust prediction performance for the *World H-H TrajNet* challenge are the following. Firstly, generalization ability across datasets. Secondly, the ability to deal with varying noise levels due to mainly including straight walking persons with different noise levels. Nevertheless, the scenarios also include behavior changes resulting, inter alia, from human-human interaction. But for *unaware* motion modeling, a change in behavior has to be estimated from the corresponding individual tracklet alone.

Table 4.2: Standard deviation of the distance between a smoothed spline fit and the ground truth trajectory data. The average $\bar{R}^2$ score for all tracklets in the subsets.

Name	$\sigma_{x,\text{spline}}$ / m	$\sigma_{y,\text{spline}}$ / m	$\bar{R}_x^2$	$\bar{R}_y^2$
Overall	0.067	0.069	0.889	0.811
BIWI Hotel	0.042	0.031	0.637	0.876
UCY Zara_02	0.029	0.035	0.952	0.758
UCY Zara_03	0.026	0.031	0.935	0.716
UCY Students_01	0.033	0.029	0.868	0.852
UCY Students_03	0.039	0.040	0.915	0.760
UCY Arxiepiskopi_01	0.050	0.027	0.959	0.677
PETS 2009 S2L1	0.037	0.026	0.781	0.877
SDD Bookstore_00	0.060	0.063	0.889	0.844
SDD Bookstore_01	0.054	0.053	0.879	0.878
SDD Bookstore_02	0.068	0.073	0.861	0.921
SDD Bookstore_03	0.069	0.061	0.951	0.830
SDD Coupa_03	0.057	0.043	0.954	0.937
SDD Deathcircle_00	0.072	0.079	0.893	0.808
SDD Deathcircle_01	0.086	0.103	0.850	0.818
SDD Deathcircle_02	0.151	0.158	0.772	0.591
SDD Deathcircle_03	0.116	0.134	0.816	0.770
SDD Gates_00	0.054	0.073	0.980	0.735
SDD Gates_01	0.064	0.084	0.859	0.890
SDD Gates_03	0.086	0.106	0.847	0.860
SDD Gates_04	0.071	0.155	0.820	0.906
SDD Hyang_04	0.048	0.050	0.829	0.842
SDD Hyang_05	0.059	0.081	0.872	0.740
SDD Hyang_06	0.070	0.066	0.875	0.811
SDD Nexus_00	0.076	0.082	0.886	0.742
SSD Nexus_02	0.069	0.074	0.934	0.726
SDD Nexus_07	0.053	0.069	0.935	0.764

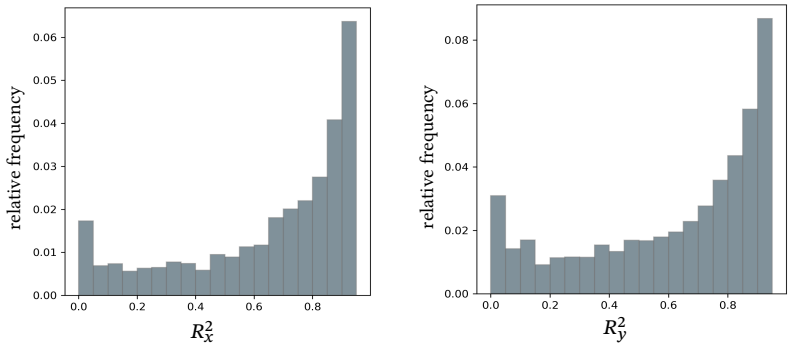

Figure 4.8: Coefficient of determination R^2 for x and y for all training tracklets of the *World H-H TrajNet* challenge.

4.2.1.2 Models and Evaluation

Finding an effective prediction network is done by using a coarse-to-fine searching strategy to reach the maximum achievable prediction accuracy without further cues like human-human interaction or human-space interaction based on basic networks. Towards this end, we started with a set of networks with a limited set of hyper-parameters to narrow it down to one network, in order to then extend the hyper-parameter set for a more exhaustive tuning.

For the *World H-H TrajNet* challenge, the performance is compared using the two error metrics of **average displacement error** (ADE) and **final displacement error** (FDE). These metrics are commonly used to assess path prediction performance (see for example [Ala16, Vem18, Pel09, Gup18, Xue18, Has18]). The average of both combined values are then used as an overall average to rank the approaches. The ADE is defined as the average L2 distance between ground truth and the prediction over all predicted time steps, and the FDE is defined as the L2 distance between the predicted final position and the true final position. For the *World H-H TrajNet* challenge, the unit of the error metrics is meter. For all experiments, 8 (3.2 seconds) consecutive positions are observed, before predicting the next 12 (4.8 seconds) positions. Since the

maximum-likelihood path is used for evaluation, the networks are initially realized as regressors by using the squared loss as a distance function for the path being predicted.

For a fixed observation window, as used in the *World H-H TrajNet* challenge, MLP-based networks can also be used to realize a prediction network, they are also considered in our coarse evaluation. Moreover, by adapting the architecture or by applying a fixed time window solution in a sliding-window fashion, the network can be used for longer input length. In contrast to other sequence prediction tasks such as natural language processing, the relevant time horizon for *path prediction* is shorter. Besides the described architecture from section 4.1, **temporal convolutional networks** (TCNs) are more often used to encode the observations for fixed time horizons. Due to their less complex structure, they are easier to train and to control [Bai18, Mil19]. Recent results indicate that TCNs can compete with RNNs in terms of sequence prediction tasks such as audio synthesis and language processing.

The following basic neural networks and corresponding variants are selected for a coarse evaluation in addition to approaches from the community and approaches provided by the organizers of the *TrajNet* challenge.

MLP: The MLP is tested with different linear and non-linear activation functions. One variation concatenates all inputs and predicts 24 outputs directly. Further, cascaded architectures with a step-wise prediction are examined. We vary between different coordinate systems of Euclidean and polar coordinates. As discussed in section 4.2.1.1, positions and offsets (also orientation normalized) are considered as inputs and outputs.

RNN-MLP: Vanilla RNNs produce an output at each time step. For the evaluation of the RNN-MLP, we vary only the MLP which is used for the decoding of the positions and offsets.

RNN-encoder-MLP: In contrast to the RNN-MLP network, the complete initialization tracklet is used to generate the internal representation before a prediction is done. The RNN-encoder-MLP is varied by alternating activation functions for the MLP and by alternatively predicting the complete future path/offsets instead of only next steps. As a further alternative, the full path

is predicted as offsets to one reference point instead of applying path integration in order to predict the final position.

RNN-encoder-decoder-model (Seq2Seq): In addition to RNN-encoder-MLPs, Seq2Seqs include a second network. This second decoder network takes the internal representation of the encoder and then starts predicting the next steps. The different settings for the evaluation of this model were due to alternating activation functions for the MLP on top of the decoder RNN.

Temporal convolutional networks (TCN): As an alternative to RNNs and based on *WaveNets* [van16], Bai et al. [Bai18] introduced a general convolution architecture for sequence prediction. We tested their standard and extended architecture with a gating mechanism (GTCN). For a more detailed description, we refer to the original papers.

All networks were trained with varying numbers of layers (1 to 5) and hidden units (4 to 64) using stochastic gradient descent with a fixed learning rate of 0.005. The models are trained for 100 epochs using ADAM optimizer [Kin15] and have been implemented in *Tensorflow* [Aba15]. Firstly, only standard RNN cells are used for the experiments. Later, we also tested with RNNs variants LSTM [Hoc97], and GRU [Cho14] (see section 4.1). As loss the mean squared error between the predicted and the ground truth position or offsets over all time steps is used.

In order to emphasize trends, an excerpt of the experiments' results is summarized in table 4.3. The best results were achieved with the RNN-encoder-MLP. However, in most cases the different architectures perform very similarly. These initial results also show that the best performing networks lie close to the result achieved with linear interpolation. Since the previous dataset analysis revealed that for a large amount of tracklets linear interpolation works quite well, it is a crucial requirement to produce stable results also in simple situations. Thus, factors such as strong overestimation of slow person velocities and some undefined random predictions when using positions can lead to weak performances compared to simple baselines. For reducing the effect of overestimation of slowly walking persons, Hasan et al. [Has18] integrated head pose information. However, this information requires a suitable head

pose detector and this additional cue is not available for the *TrajNet* challenge. We can only remark for the tested networks that this effect can also differ for different runs. Naturally, it is important that during training, the networks see enough samples of standing and slow-moving situations. Instead of excluding such samples through heuristics or probabilistic filtering, which can help during application, we counteract this by data augmentation (see next section).

Table 4.3: Results from our coarse evaluation on the data corresponding to the world plane human-human dataset (*World H-H TrajNet* challenge). In contrast to the results in table 4.4, the shown results are not generated with the official benchmark toolkit. Although the same datasets are used for training and testing, the exact test set selection of ground truth trajectories or tracklets for the challenge are not publicly available, and thus the results may vary.

Approach	Overall Average	FDE / m	ADE / m
Linear interpolation	0.894	1.359	0.429
Linear MLP (Pos)	1.041	1.592	0.491
Linear MLP (Off)	0.896	1.384	0.407
Non-linear MLP (Off)	2.103	3.181	1.024
Linear RNN	0.951	1.482	0.420
Non-linear RNN	0.841	1.300	0.381
Linear RNN-encoder-MLP	0.892	1.381	0.404
Non-linear RNN-encoder-MLP	0.827	1.276	0.377
Linear Seq2Seq	0.923	1.429	0.418
Non-linear Seq2Seq	0.860	1.331	0.390
TCN	0.841	1.301	0.381
Gated TCN	0.947	1.468	0.426

These results show that using deep neural networks for *path prediction* can produce undesired unstable results. Further, there is no clear-cut best performing model and clear guidance towards a class of models. Thus the gap between an MLP predictor and a Seq2Seq model is very narrow in the test

scenarios. However, besides the factors derived from the data analysis, a prediction of the full path instead of step-wise prediction helps to overcome an accumulation of errors that are fed back into the networks. For the *TrajNet* challenge with a fixed prediction horizon, we thus prefer the RNN-encoder-MLP over a Seq2Seq model. In the domain of human pose prediction based on RNNs, Zhou et al [Zho17] reduced this problem with an auto-conditioned RNN, and Martinez et al. [Mar17] propose using a Seq2Seq model along with a sampling-based loss. In [Zim12], Zimmermann et al. reported that extending prediction into the future helps to balance the information flow and to achieve a better input-output relationship. They called the prediction of more future steps from the encoded representation *overshooting*. In accordance to the results presented by [Mil19] for other sequence prediction tasks, TCNs perform also very similar to RNNs for *path prediction*. About the same time as our results on *path prediction* were published [Bec18c, Bec18b], [Nik18] Nikhil and Morris concordantly reported that TCNs yield competitive results for pedestrian *path prediction*.

Despite several benefits of TCNs over RNNs and variants, we stick with an RNN-based solution due to its connection to Bayesian filtering. Furthermore, RNNs are more common as part of architectures which model interactions [Ala16, Ale17, Has18, Xue18] to represent single motion (see section 2.3). Due to results from the initial evaluation and the discussed reasons, we chose an RNN-encoder-MLP as our favored model to further apply the ablation study to achieve more robust performance results.

4.2.1.3 RNN-based prediction network: RED Predictor

For the comparison to existing approaches, the selected model is an RNN-encoder-MLP. In this section, the final design choices which lead to the submitted predictor and achieved top-rank at the *World H-H TrajNet* challenge, are summarized.

The RNN-encoder can generalize to deal with varying noisy inputs and is thus able to better capture the person's motion compared to the linear interpolation

baseline. The main insight is that motion continuity is easier to express in offsets or velocities because it takes considerably more modeling effort to represent all possible conditioning positions. Especially for the *World H-H TrajNet* challenge, with different ranges for positions in the training and test set, this has a significant influence on whether a good performance can be obtained. Instead of using the given input sequence $\mathcal{Z} = \{(x^k, y^k) \in \mathbb{R}^2 | k = 1, \dots, k_{obs}\}$ of k_{obs} consecutive pedestrian positions along a trajectory, here the offsets are used for conditioning the network $\mathcal{Z} = \{(\Delta_x^k, \Delta_y^k) \in \mathbb{R}^2 | k = 2, \dots, k_{obs}\}$. Apart from the smaller modeling effort to represent conditioned offsets and the prevention of undefined states due to a suitable data range, this domain shift makes data pre-processing like the used standardization more reasonable. Since the offset or rather velocity distribution follows approximately a normal distribution around the expected walking speeds of pedestrians in contrast to the position distribution, through standardization of the offsets, the expected behavior is straight walking, and thus the network uses the deviations from the dominant walking pattern as inputs.

In our work [Hug17], we demonstrated that RNN-based solutions can capture environment cues from position data only. Therefore, they incorporate scene-specific knowledge which in turn can be a hindrance for generalizing across scenes. This actually positive ability relies on additional *contextual* cues to enable a better transfer to other scenarios. Thus, by using trajectory samples from other datasets, an undesired scene-prior is included. Naturally, this effect gets stronger in case the scene includes a clear spatially-dependent behavior, such as roundabouts and crossings which are present in the *Stanford Drone Dataset (SDD)*. For short time horizons and to better generalize across different and unseen environments, the switch to relative positions (all training tracklets start at the origin), and to use offsets should be preferred. Thus, the spatial information only persists in an implicit fashion by performing path integration. As discussed in section 2.3, for longer time-horizons the intention of the object motion, here pedestrians, is more strongly motivated by its goals. Classifying the goals of pedestrians in a scene requires further scene knowledge. The above input modifications help in training neural networks by scaling the inputs to a reasonable range, although in theory the desired scaling can be achieved only with appropriate weights and biases. By using

non-linear activation functions such as sigmoid type functions, it is impossible to achieve an ever increasing trend. Nevertheless, a network can achieve output values greater than the bound of a single neuron, but the network can saturate at minimum or maximum values, in particular for trending input data. Straight walking can be interpreted as increasing trended data of position along a trajectory. The last observation can be used to provide a direct connection to the output layers, which is realized as linear MLP. By this non-squashing connection, the saturation problem can be countered. Due to careful scaling around the known preferred walking behavior inside reasonable bounds, the network can better handle the trend in trajectory data dominated by straight walking. Since the pedestrian motions are not raw trended sequential data and due to careful pre-processing, a direct connection for reducing the saturation effect is not mandatory but should be kept in mind for using position data only.

In order to deal with discretization artifacts in the ground truth trajectories and to make further training easier, smoothed trajectories are used as the desired output. As described, the minimal spline fit with a polynomial function of varying degree for a complete tracklet is used to achieve smoother and more persistent dynamics. Nonetheless, RNNs can generalize over the outputs and produce smooth predictions, but the intention is not to synthesize the noise, but make training easier in terms of reducing the artifacts in the ground truth data. As a drawback, the fit can produce incorrect results in some cases, but overall the trajectories look more natural and smooth. Especially if longer tracklets or rather complete trajectories can be considered, the fitting results improve due to the incorporation of more data points. In case of poor spline fit corresponding to a large fitting error for all degrees of the polynomial function, the original trajectory is kept. Examples of fitted ground truth tracklets are depicted in figure 4.9.

In order to reduce the effect of error accumulation during a step-wise prediction, *overshooting* is applied [Zim12]. Instead of feeding back RNN outputs step-wise, the encoded representation is used to apply a multi-step prediction.

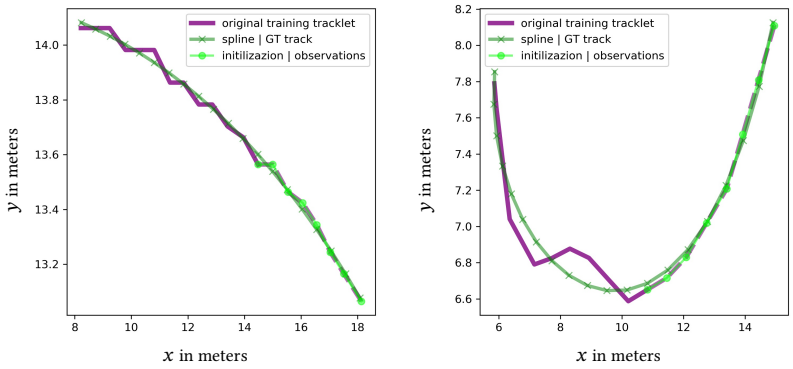

Figure 4.9: Example visualization of pre-processed ground truth trajectories to produce a more persistent motion behavior and reduce ground truth discretization artifacts.

For the *TrajNet* challenge, the whole future path is predicted. Full path integration works similarly well, but here offsets to the reference positions (last observed position) are predicted.

Physics-based models, such as a CV model, are independent of the absolute values of states despite not being physically plausible. In spite of the loss of physical intractability and related problems, such models can be applied as general translational models without modification. For example, this applies to tracking an object in image space as discussed in section 3.2.

In contrast, deep learning-based models can over-fit to limited training samples and are not able to generalize to input data which is too far outside the seen input range. For example, in the *BIWI ETH* dataset or *BIWI Hotel* dataset, the trajectories include a strong bias along a specific direction. Thus, training on *BIWI ETH* trajectories and inferring on rotated trajectories lead to underestimation of the true velocity along the dominating direction in the training samples. This effect can result from over-fitting to data bias or scene-prior, deviation from input range or it can be seen as a lack of generalization ability. In addition to the explained factor, this potential shortcoming can be counteracted by data augmentation. Due to the fact that there exist several reasonable *physics*-based models to describe pedestrian behavior, it is possible to

augment the training data by simulating realistic motion profiles. Although the scenarios of the *TrajNet* challenge are relatively well suited to improve training by simulation due to the fact that pedestrian behavior is evaluated on ground level in a world coordinate system, data augmentation is only done by reverting all training tracklets of the provided challenge data. Thereby, the amount of training samples reflecting single object behavior is directly doubled. For the submitted results, no further data augmentation techniques are applied.

The discussed effect of over-fitting to a scene is depicted in figure 4.10. The images show the prediction of an RNN-encoder with an MDN, which parametrizes a bi-variate Gaussian, as last layer for an original tracklet of the *ETH* dataset and a tracklet rotated by 90°. As a loss, a linear combination of the negative log-likelihood for the ground truth future positions under the predicted positions (see section 4.19) and the mean squared error to the ground truth trajectories is used. The models are both trained on a subset of the *ETH* dataset. On the left, an original test tracklet is used to infer future positions and covariances. On the right, a rotated test tracklet. It is clearly visible that the model on the right produces a bad prediction due to the effects described above.

The proposed simple but effective predictor for the *TrajNet* challenge combines all the listed factors. At its core, the architecture is an **RNN-encoder with a dense MLP stacked on top** (RED). Hence, the predictor is referred to as RED predictor when it is realized as pure regression model. In the demonstration example from figure 4.10 where the MLP is replaced with an MDN to capture also the predictive distribution, the term RED-MDN is used. Realized as a regression model and without direct connection to the last observation, the RED predictor can be defined by:

$$\begin{aligned} \mathbf{h}_{enc}^{k} &= \mathrm{RNN}(\mathbf{h}_{enc}^{k-1}, \Delta_{(x,y)}^{k}; \boldsymbol{\Theta}_{enc}) \\ y^{K_{pre}} &= \{(\Delta_x^{k+k_{pred}}, \Delta_y^{k+k_{pred}}) + (x^k, y^k)\}_{k_{pre}=1}^{K_{pre}} = \mathrm{MLP}(\mathbf{h}_{enc}^{k}; \boldsymbol{\Theta}_{MLP}) \end{aligned} \tag{4.21}$$

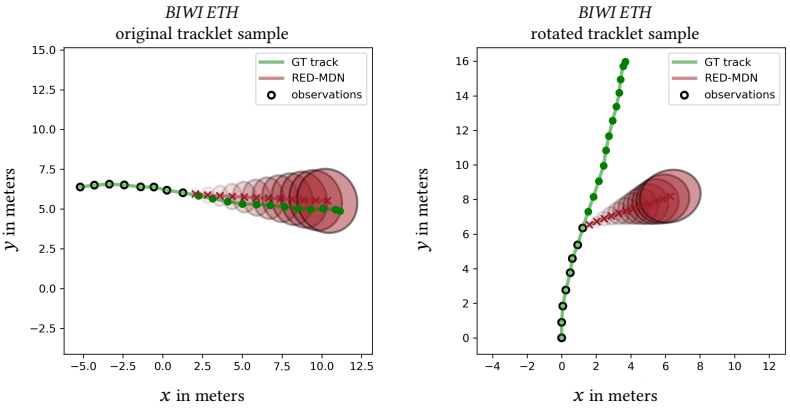

Figure 4.10: Qualitative results with an RNN-encoder model with well adapted input range and without. On the left and on the right the same observations are used, but on the right they are rotated. (Left) The model is able to represent the motion and make a reasonable prediction. (Right) Due to unbalanced training samples with trajectories mainly orientated along one direction, the network produces a poor estimate for a 90° rotated trajectory.

Here, RNN(·) is the recurrent network, $\mathbf{h}_{enc}$ the hidden state of the RNN-encoder with corresponding weights $\mathbf{W}_{enc}$ and biases $\mathbf{b}_{enc}$ (parameters $\mathbf{\Theta}_{enc}$), which is used to generate the full, smoothed path. The term MLP(·) reflects an MLP including the conforming weights $\mathbf{W}_{MLP}$ and biases $\mathbf{b}_{MLP}$ to map the vector $\mathbf{h}_{enc}$ to the observation space ($\mathbf{\Theta}_{MLP} = \{\mathbf{W}_{MLP}, \mathbf{b}_{MLP}\}$). The overall architecture is visualized in figure 4.11.

The submitted results of our final RED predictor is highlighted in red in table 4.4, using the official benchmark toolkit. Some qualitative predictions examples from the RED-MDN on the *BIWI ETH* and *BIWI Hotel* dataset are visualized in figures 4.12, 4.13, and respectively in figures 4.14, 4.15. Although the dominating behavior of the pedestrians is straight walking, the scenarios also include diverse and more complex behaviors. The images depict that the network is able to capture the different motion types and to adapt by incorporating new observations. Prediction is done for individual pedestrians solely based on the observed trajectory. The examples show that, despite using any

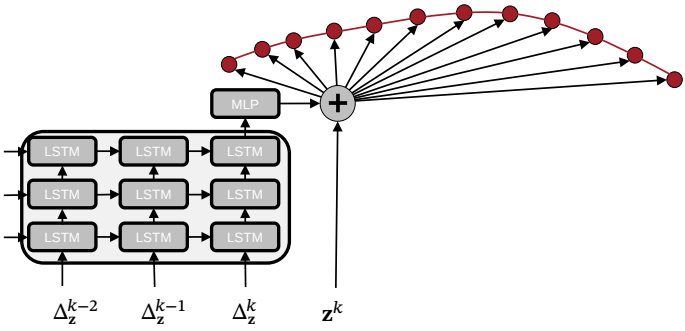

Figure 4.11: Visualization of the RED architecture. The conditioning is done for the full initialization sequence $\mathcal{Z} = \{(\Delta_x^k, \Delta_y^k) \in \mathbb{R}^2 | k = 2, \dots, 8\}$. The internal representation is then used to predict the desired path at once (all 12 positions) using the last observed position (x^8, y^8) as reference for localization.

cues from other persons, the RED-MDN predicts similar dynamical behavior for persons walking closely together in a group. By comparing different sample situations, it can be seen that the network is able to model different walking speeds. It can also be seen, how the prediction is adapted when the dynamics are changing. In figure 4.14 and 4.14, a deceleration and a stopping behavior is correctly captured. The ability of RNN-based network to deal with the maneuver type of changing dynamics is discussed in detail in section 4.2.2.

After a performed fine search for the selected network, the shown result of table 4.4 is produced with an LSTM unit (state size of 32) and one recurrent layer. The proposed predictor is able to produce sophisticated results compared to elaborate models which additionally rely on interaction information such as the model from Yamaguchi et al. [Yam11] (extended social-force-field model based on the approach of Helbing and Molnár [Hel95]) and the Social-LSTM [Ala16]. Compared to all submitted approaches of the *World H-H TrajNet* 2018 challenge, the RED predictor achieved the best result. All other results were either officially submitted or provided by the organizers.

Table 4.4: Results for the world plane human-human dataset (*World H-H TrajNet*) challenge including our submitted RNN-based approach (RED predictor). The results of approaches marked with (*) are directly obtained from corresponding papers. Other results are taken from the *TrajNet* website[a] [Sad18].

Approach	Overall Average	FDE / m	ADE / m	Reference
RED predictor	**0.783**	**1.207**	**0.359**	**Ours**
SR-LSTM	0.815	1.229	0.370	[Zha19]
Social Forces (EWAP)	0.819	1.266	0.371	[Yam11]
FISHY	0.820	1.256	0.375	
JHU	0.844	1.304	0.384	
Predictor SUL	0.887	1.374	0.399	
Linear interpolation	0.894	1.359	0.429	
Social Forces (ATTR)	0.904	1.395	0.412	[Yam11]
LVA*	0.945	1.449	0.438	[Xue19]
SGAN	1.334	2.107	0.506	[Gup18]
OSG	1.385	2.106	0.664	
Social LSTM	1.387	2.098	0.675	[Ala16]
LV*	1.398	2.072	0.723	[Xue19]
Interactive Gaussian Processes	1.642	1.038	2.245	[Ell09]
Vanilla LSTM	2.107	3.114	1.100	
Occupancy LSTM	2.111	3.12	1.101	[Ala16]

[a] *TrajNet* website: (http://trajnet.stanford.edu/, last accessed 19.12.2019)

Nevertheless, the Social-LSTM is one of the first proposed RNN-based architectures which includes human-human interaction and laid the basis for architectures like presented in the work of Hasan et al. [Has18], Xue et al. [Xue18], and Zhang et al. [Zha19]. Single motion is modeled with an RNN or rather an LSTM network. By applying some of the proposed factors to the models, it is expected that the models and corresponding extensions are able to outperform the proposed single motion predictor in datasets with high pedestrian density.

4.2.1.4 Summary

Although a part of the presented modifications is tailored to the *TrajNet* challenge, the results show how to enhance prediction performance for RNN-based networks. The provided analysis of the datasets reveals some weaknesses of the challenge, such as no defined categorization into linear and non-linear or with and without human interaction. However, it is difficult to provide standardized benchmarking in particular due to the fast-growing body of different approaches for capturing object dynamics. In terms of our goals, the trajectory data reflects the desired properties of observations provided by an underlying visual tracking component with varying noise levels. Hence, the results show that RNN-based models are able to deal with this maneuver type. Due to the emphasized modifications, the proposed network achieves state-of-the-art performance compared to existing alternative approaches on a public benchmark dedicated to *path prediction*. It is clear that independently from the model complexity, approaches restricted to observing only information from a single trajectory are in range to a reachable performance limit on the current dataset repository. However, the *TrajNet* benchmark also provides human-human and human-space information. Recent work such as the approaches of Gupta et al. [Gup18] or Xua et al. [Xue18] (human-human) and Sadeghian et al. [Sad19] (human-human, human-space) show possibilities of how to better anticipate the pedestrian behavior based by integrating these *contextual* cues.

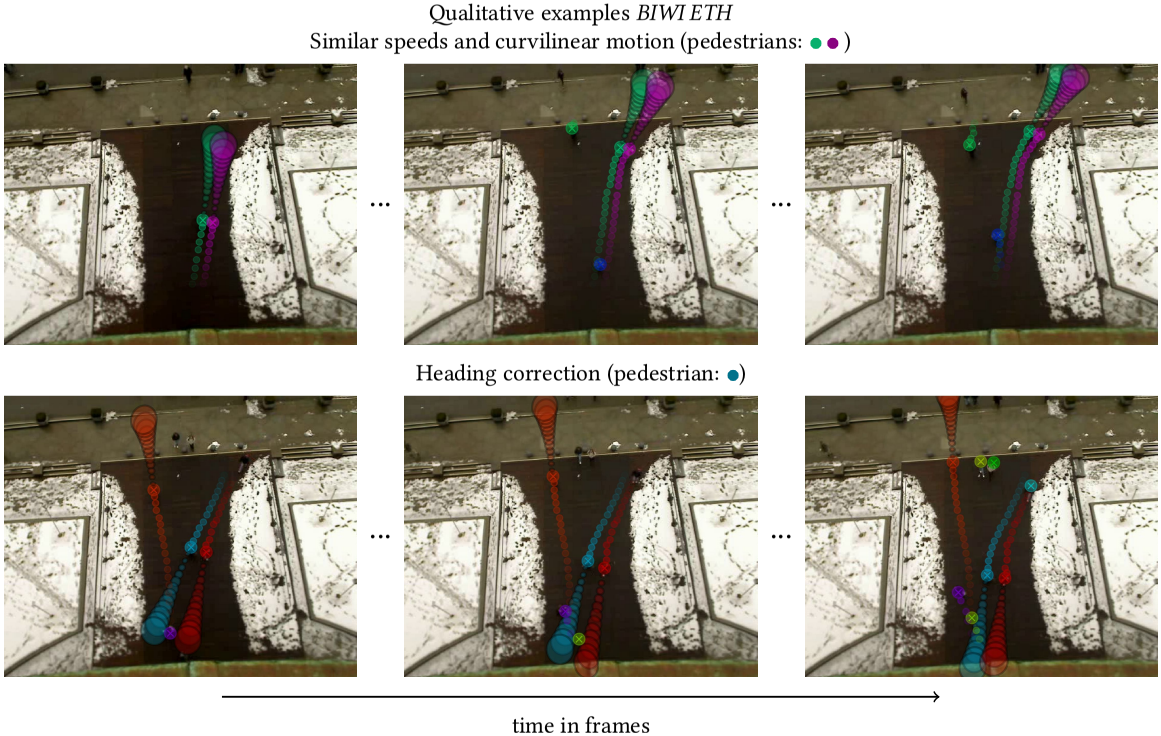

Figure 4.12: Diverse predictions generated by the RED-MDN on the *BIWI ETH* dataset. Each row shows three time steps of different sample situations. Prediction is done for individual pedestrians (color-coded) solely based on the observed trajectory.

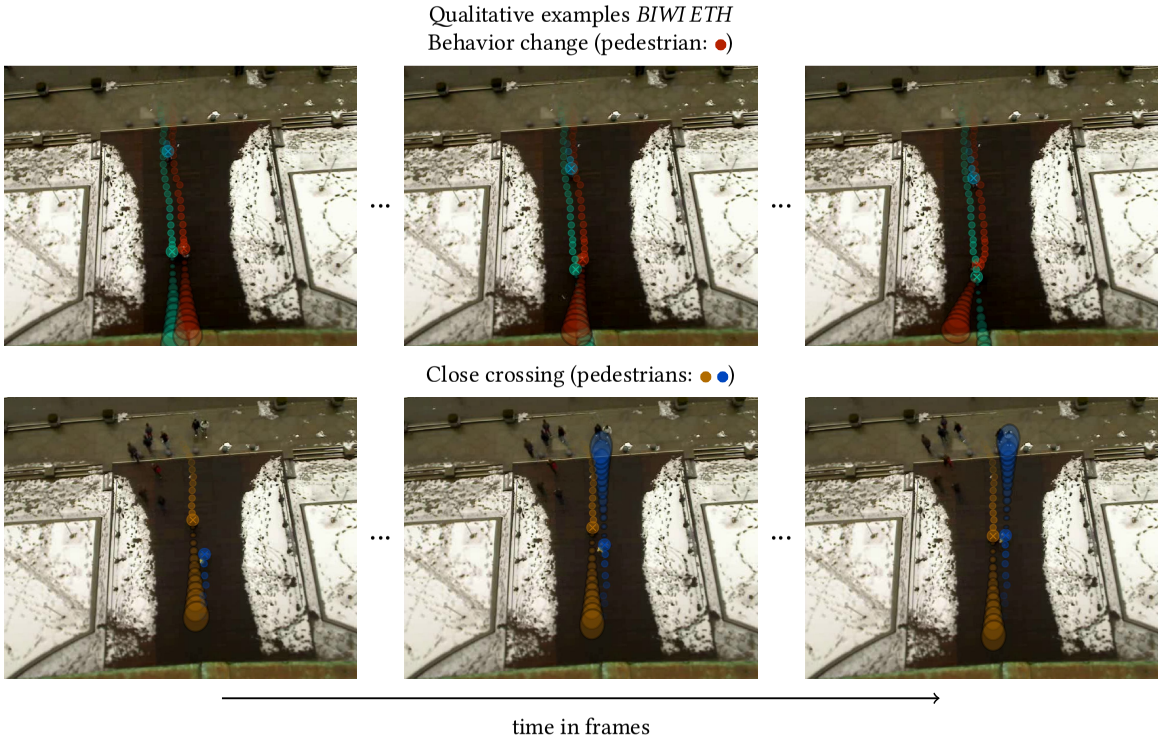

Figure 4.13: Additional diverse predictions generated by the RED-MDN on the *BIWI ETH* dataset. Each row shows three time steps of different sample situations. Prediction is done for individual pedestrians (color-coded) solely based on the observed trajectory.

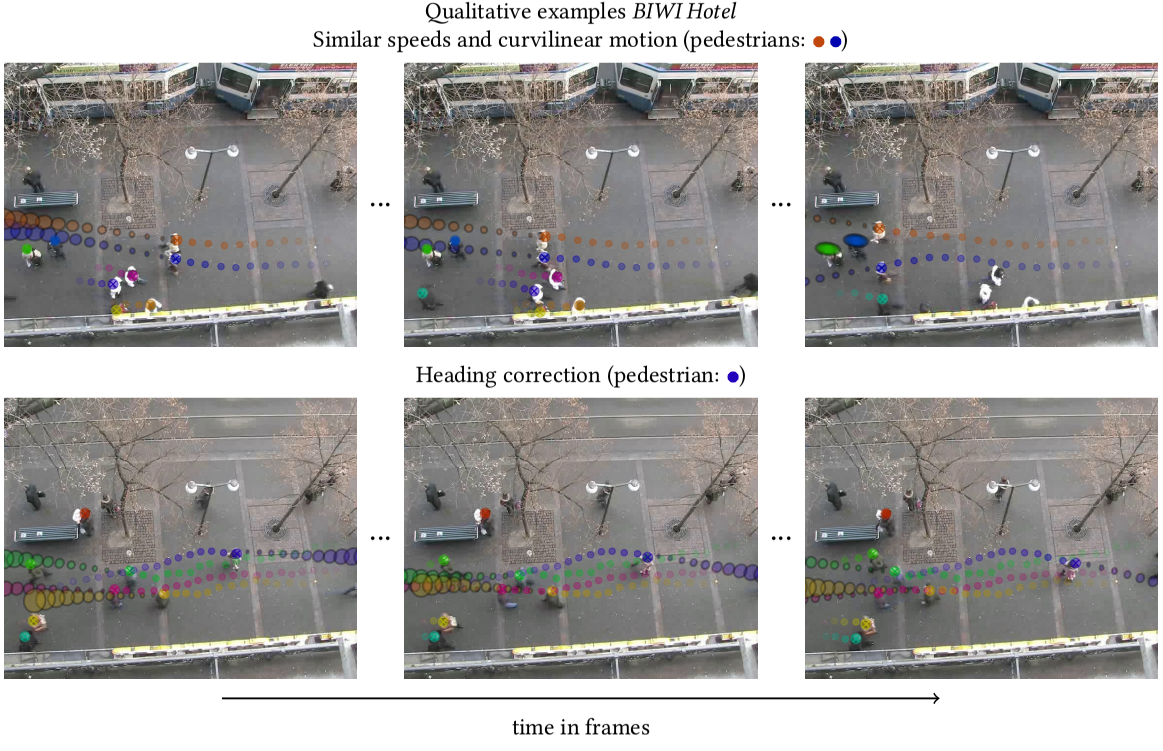

Figure 4.14: Diverse predictions generated by the RED-MDN on the *BIWI Hotel* dataset. Each row shows three time steps of different sample situations. Prediction is done for individual pedestrians (color-coded) solely based on the observed trajectory.

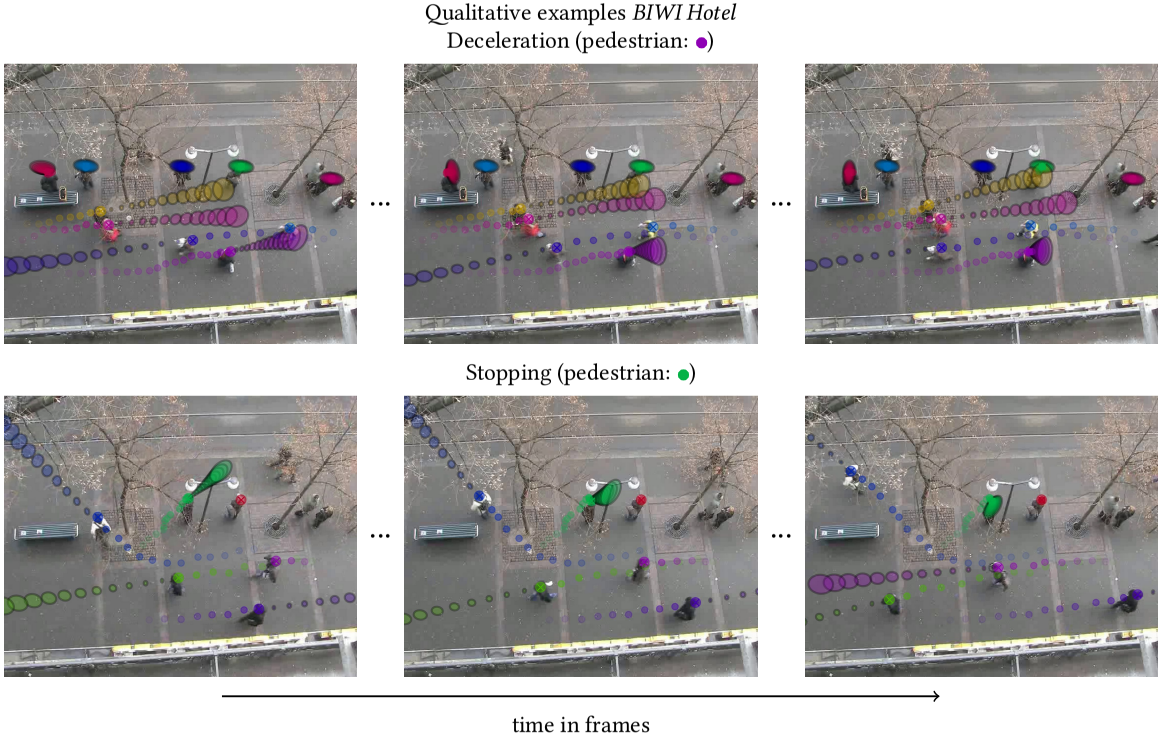

Figure 4.15: Additional Diverse predictions generated by the RED-MDN on the *BIWI Hotel* dataset. Each row shows three time steps of different sample situations. Prediction is done for individual pedestrians (color-coded) solely based on the observed trajectory.

4.2.2 Intention Prediction

In this section, the ability of a proposed RNN-based solution with respect to switching dynamics of pedestrians is evaluated for scenarios within the application domain of intelligent vehicles. In the domain of intelligent vehicles, pedestrian intention is mainly used as part of the overall behavior analysis of a vision-based active safety system and applied jointly with *path prediction*. Due to the short time window for emergency breaking, *physics*-based approaches are the preferred solution. We consider concrete scenarios that are tackled with *multiple-model* approaches such as the IMM filter. Although many approaches can relatively reliably predict the location of objects a few seconds ahead, they still struggle to predict when the object will stop [Rid18, Has15b]. Hence, the scenario of stopping pedestrians with the corresponding *physics*-based *multiple-model* solutions is analyzed in particular.

Inspired by an IMM filter solution, we propose an RNN-based IMM filter surrogate for improved handling of varying dynamics over time. On the one hand, the presented RNN-based model is able to provide a confidence value for the performed dynamics. On the other hand, it can overcome some limitations of the classic IMM filter. The proposed RNN-IMM incorporates the insight from section 4.2.1 and is based on an RNN-encoder-decoder network introduced by Deo and Trivedi [Deo18] for the case of freeway traffic prediction.

In the following, a description of the RNN-IMM is provided before different maneuver scenarios including corresponding common *physics*-based dynamical models for Kalman and IMM filters are analyzed.

4.2.2.1 RNN-based IMM Filter Surrogate

The goal is to devise a model that can successfully predict future paths of pedestrians and represent alternating pedestrian dynamics, e.g., dynamics that can transition from straight walking to a turning maneuver or to stopping. Both the IMM filter and the RNN-IMM predict a parametric distribution

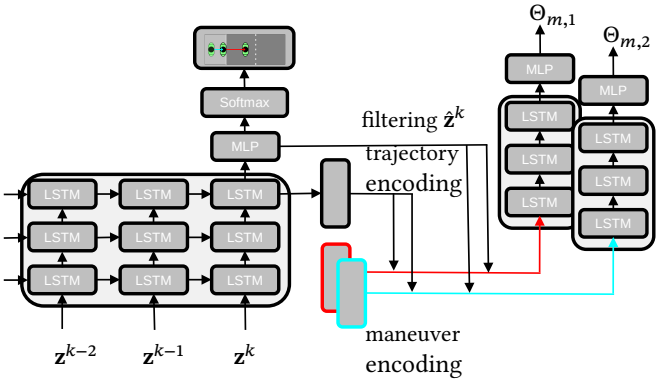

Figure 4.16: Visualization of the RNN-based IMM filter surrogate (RNN-encoder-decoder network) for jointly predicting specific dynamical probabilities and corresponding distributions of future trajectory positions. The encoder predicts the dynamical probabilities and the filtered position for the current time step. The decoder uses the context vector and the position estimate to predict future pedestrian locations.

over object states and jointly capture maneuver probabilities for subsequent processing.

As in the previous section, trajectory prediction is formally stated as the problem of predicting the future trajectories of a pedestrian, conditioned on its track history. Given an input sequence $\mathcal{Z} = \{(x^k, y^k) \in \mathbb{R}^2 | k = 1, \dots, k_{obs}\}$ of k_{obs} consecutive observed pedestrian positions $\mathbf{z}^k = (x^k, y^k)$ at time step k along a trajectory, the task is to filter the current position $\hat{\mathbf{z}}^k = (x^k, y^k)$ and to generate a multi-modal prediction for the next k_{pred} positions $\{\mathbf{z}^{k+1}, \mathbf{z}^{k+2}, \dots, \mathbf{z}^{k+k_{pred}}\}$.

As discussed in section 4.2.1, it takes considerably more modeling effort to represent all possible conditioning positions. Thus motion continuity is easier to express in offsets or velocities. In order to exploit scene-specific knowledge for trajectory prediction, additional use of the position information is required. In our work [Hug17], we showed that RNN-based trajectory prediction models are able to capture spatially dependent behavior changes only from motion data for sufficient training samples. In order to analyze the

RNN-based model capabilities in prototypical maneuver scenarios, mostly synthetic data is used as an ideal sanity check performance evaluation. For a fixed reference system, position information is used to estimate the true position. In addition, the offsets are used for conditioning the network $\mathcal{Z} = \{(x^k, y^k, \Delta_x^k, \Delta_y^k) \in \mathbb{R}^4 | k = 2, \dots, k_{obs}\}$. The future trajectory is denoted with $\mathcal{Y} = \{(x^k, y^k) \in \mathbb{R}^2 | k = k_{obs} + 1, \dots, k_{pred}\}$ and the filtered position with $\mathbf{x}'^k = \mathbf{z}^k$. Compared to Bayesian filtering, $\mathbf{x}'^k$ is not the full dynamical state, but the observable state $\mathbf{z}^k$after applying the RNN equivalent of the observation model (see equation 2.9). This expression is used to highlight the analogy to the IMM filter, but the notion of deterministic inputs for RNNs is kept. The model estimates the conditional distribution $p(\mathcal{Y}, \mathbf{x}'^k | \mathcal{Z})$. In order to identify specific dynamics under M desired maneuver classes (e.g., turning maneuvers, stopping, and straight walking), this term can be given by:

$$p(\mathcal{Y}, \mathbf{x}'^k | \mathcal{Z}) = \sum_i^M p_{\Theta_{RNN-IMM}}(\mathcal{Y}, \mathbf{x}'^k | m_i, \mathcal{Z}) P(m_i | \mathcal{Z}). \tag{4.22}$$

Here, $\mathbf{\Theta}_{RNN-IMM} = \{\mathbf{\Theta}_{RNN-IMM}^{k_{obs}+1}, \dots, \mathbf{\Theta}_{RNN-IMM}^{k_{pred}}\}$ are the parameters of an L component Gaussian mixture model $\mathbf{\Theta}_{RNN-IMM}^k = (\boldsymbol{\mu}_l^k, \mathbf{\Sigma}^k, w_l^k)_{l=1,\dots,L}$. By adding the maneuver context in form of the posterior mode probability, $P(m_i | \mathcal{Z}) \stackrel{\wedge}{=} \alpha_i$ the analogy to the classic IMM filter becomes apparent.

For an IMM filter, the mode probability is used to calculate the mixing probabilities to combine the set of chosen candidate models into a merged estimate (see equation 3.50). In case of using an IMM filter, the time behavior of the basic filter set is modeled as a homogeneous (time-invariant) Markov chain with a fixed transition probability matrix (TPM) $p_{ij} \triangleq P(m_i^k | m_j^{k-1})$. As shown in section 3.1.2, the posterior density of the IMM filter can be written under the assumption that M models describe the variation of the dynamics as follows:

$$p(\mathbf{x}^k | \mathcal{Z}) = \sum_i^M p_{\Theta_{IMM}}(\mathbf{x}^k | m_i, \mathcal{Z}) P(m_i | \mathcal{Z}). \tag{4.23}$$

Here, $p_{\Theta_{IMM}}(\mathbf{x}^k|m_i,\mathcal{Z})$ is in the context of an IMM filter a Gaussian distribution and $P(m_i|\mathcal{Z}) \stackrel{\wedge}{=} \alpha_i$ is the posterior mode probability for the IMM filter. As explained, the transition between different dynamics is modeled as a first-order Markov chain for an IMM filter. The law of total probability allows computing new mode probabilities based on the transition probabilities. Given the current mode probabilities and transition probabilities, the weighting probabilities $\alpha_{i|j}$ for the mixing step of the IMM filter can be calculated. For each model m_i and m_j, they are calculated as $\alpha_{i|j}^{k-1} = {}^1/_{\bar{c}_j} p_{ij}\alpha_i^{k-1}$ with a normalization factor $\bar{c}_j = \sum_{i=1}^{M} p_{ij}\alpha_i^{k-1}$ (see section 3.1.2). Then, in the prediction stage, each filter is applied independently using the calculated mixed initial condition. Subsequently, the model probabilities are adapted according to the likelihood of each filter.
Whereas explicit modeling of the switching behavior and the object dynamics of the IMM filter stands in contrast to an implicit dynamic encoding of an RNN-based approach. In order to provide an IMM filter surrogate, the proposed model also estimates mode probabilities and filters or rather de-noises the current position based on noisy observations $\mathcal{Z}$. By writing the conditional distribution $p(\mathcal{Y},\mathbf{x}'|\mathcal{Z})$ of the RNN-based approach in form of equation 4.22, the desired estimates can be inferred from the hidden states of the RNN $\mathbf{h}$. This formulation does not require to set the parameter of the TPM matrix manually, which is commonly done based on the mean sojourn time (the mean time an object stays in a motion type [Sch13, Bar02]) or as stated in the work of Bar-Shalom [Bar02], an ad-hoc approach is filling the diagonals with values close to one. For the proposed RNN-based IMM filter surrogate, the basic architecture is a recurrent encoder-decoder model. The encoder takes the frame by frame input sequence $\mathcal{Z}$. The hidden state vector of the encoder is updated at each time step based on the previous hidden state and the current observation. The generated internal representation is used to predict mode probabilities $\boldsymbol{\alpha}^k$ at the current time step and the filtered position $\mathbf{x}'^k$. With an

embedding of the current observations, the encoder can be defined as follows:

$$\begin{aligned}
\mathbf{e}_{enc}^{k} &= \text{EMB}(\mathbf{z}^{k}; \boldsymbol{\Theta}_{ee}) \\
\mathbf{h}_{enc}^{k} &= \text{RNN}(\mathbf{h}_{enc}^{k-1}, \mathbf{e}_{enc}^{k}; \boldsymbol{\Theta}_{enc}) \\
\boldsymbol{\alpha}_{logits}^{k}, \hat{\mathbf{z}}^{k}, \hat{\boldsymbol{\Sigma}}^{k} &= \text{MLP}(\mathbf{h}_{enc}^{k}; \boldsymbol{\Theta}_{MLP}) \\
\hat{\boldsymbol{\alpha}}^{k} &= \frac{\exp\left(\boldsymbol{\alpha}_{logits}^{k}\right)}{\sum_{j=1}^{M} \exp\left(\alpha_{logits,j}^{k}\right)}.
\end{aligned} \tag{4.24}$$

Here, RNN$(\cdot)$ is the recurrent network, $\mathbf{h}$ the hidden state of the RNN, MLP$(\cdot)$ the multi-layer perceptron, and EMB$(\cdot)$ an embedding layer. $\boldsymbol{\Theta}_{(\cdot)}$ represents the weights $\mathbf{W}_{(\cdot)}$ and biases $\mathbf{b}_{(\cdot)}$ of the MLP, EMB, and respectively RNN. The final state of the encoder can be expected to encode information about the track history. For generating a distribution over trajectories conditioned on dynamical modes, the encoder hidden state is appended with an one-hot encoded vector corresponding to specific dynamics and the filtered current position. The decoder of the model can be defined as follows:

$$\begin{aligned}
\mathbf{h}_{dec}^{k-1} &= \mathbf{h}_{enc}^{k} \\
\mathbf{h}_{dec}^{k} &= \text{RNN}(\mathbf{h}_{dec}^{k-1}, \hat{\mathbf{z}}^{k}, \boldsymbol{\alpha}^{k}; \boldsymbol{\Theta}_{dec}) \\
\hat{\mathcal{Y}} &= \{(\hat{\boldsymbol{\mu}}_{l}^{k} + \hat{\mathbf{x}}^{k_{obs}}, \hat{\boldsymbol{\Sigma}}_{l}^{k}, \hat{w}_{l}^{k})\}_{k=k_{obs}+1}^{K} = \text{MLP}(\mathbf{h}_{dec}^{k}; \boldsymbol{\Theta}_{de})
\end{aligned} \tag{4.25}$$

The decoder is used to parametrize an MDN or rather $\boldsymbol{\Theta}_{RNN-IMM}$ directly for several positions in the future. Although the overall RNN-IMM uses the trajectory prediction and dynamical classification jointly, the loss function for training is split into three parts. Dynamical classification is trained to minimize the cross-entropy of the different M dynamical modes:

$$\mathcal{L}(\mathcal{Z})_{maneuver} = -\sum_{j=1}^{M} \alpha_{j,GT}^{k} \log(\hat{\alpha}_{j}^{k}). \tag{4.26}$$

Additionally, the encoder is trained by minimizing the filtering loss $\mathcal{L}(\mathcal{Z})_{filter}$ in form of the negative log-likelihood of the ground truth current pedestrian

position under the predicted position. Finally, the complete encoder-decoder is trained by minimizing the negative log-likelihood for the ground truth future pedestrian locations conditioned on the performed maneuver class. The context vector is appended with the ground truth values of the dynamical classes for each training trajectory. This results in the following loss function:

$$\begin{aligned}\mathcal{L}(\mathcal{Z})_{pred} &= -\log\left(p_{\Theta_{RNN-IMM}}(y_{GT}|m_{GT},\mathcal{Z})P(m_{GT}|\mathcal{Z})\right) \\ &= \sum_{k=k_{obs}+1}^{K} -\log\left(\sum_{l=1}^{L} \hat{w}_l^k \mathcal{N}(\mathbf{z}^k|\hat{\boldsymbol{\mu}}_l^k + \hat{\mathbf{z}}^{k_{obs}}, \hat{\boldsymbol{\Sigma}}_l^k; m_{GT})\right).\end{aligned} \tag{4.27}$$

The overall architecture is visualized in figure 4.16. The context vector combines the encoding of the track history with the encoding of the alternating dynamical classes. Together with the filtered position, it is used as input for the decoder.

4.2.2.2 Evaluation

This section consists of an evaluation of the proposed RNN-IMM. The evaluation is concerned with verifying the overall viability of the approach in maneuver situations in terms of switching motion behavior. Firstly, synthetic test conditions are used in order to gain insight into the model behavior in different typical pedestrian maneuvers. By doing that, factors such as a restricted amount of training samples are avoided. Later, the evaluation is also done on the *Daimler context path prediction* dataset [Koo14], which is a real-world dataset designed to capture pedestrian maneuvers from a driving vehicle. The synthetic data reflects the *Daimler context path prediction* dataset and *Daimler path prediction* dataset [Sch13] by capturing similar condition and using the statistics in the data for generating samples. Both datasets capture sequences recorded with the same sensor setup or rather the same vehicle. The 2014 dataset version is focused on *crossing* and *stopping* maneuvers.

In the domain of intelligent vehicles, *intention prediction* of pedestrians is commonly done in an ego-motion compensated vehicle centered coordinate system. The detections provided by an object detector are mapped onto a ground plane in world coordinates. For the *Daimler* datasets, a stereo camera-system (baseline 22 cm, 16 **frames per second** (fps), 1176×640 pixels) is used, inter alia, for mapping the observation to the physical world. Thereby, the incorporation of prior knowledge about the dynamics of pedestrian motion is enabled. As explained in section 2.3.2, there exist several *physics*-based dynamical models that are applied in combination with Bayesian filters under these conditions. The choice of selected *physics*-based reference approaches is orientated on a comparative study from Schneider et al. [Sch13] on recursive Bayesian filters for pedestrian *path prediction* at short time horizons (below 2 seconds) on the *Daimler path prediction* dataset. For *single-models* in combination with a Kalman filter, a CV and a CA model are considered for *crossing* scenarios. These dynamical models are also used for the prediction of pedestrian positions by Bertozzi et al. [Ber04], Meuter et al. [Meu08], Møgelmose et al [Møg15], Binelli et al. [Bin05], and Elnagar et al. [Eln01] to name a few. Further, the proposed IMM filters of Schneider et al. are the core of the introduced extensions for including several *contextual* cues to control the transition probability between single dynamical models (see section 2.3.2 [Koo19, Koo14, Sch15]).

Since the *Daimler path prediction* dataset provides only a maximum number of 23 sequences for single motion types, in order to avoid problems such as a limited number of training samples and to gain some insights into a controlled setup, synthetic data is used to perform a sanity check analysis. Furthermore, there is a location bias in the *Daimler* datasets, which is more present for the 2013 version and the *bending in* maneuvers. Since recursive Bayesian filters make no use of the spatial context of a scene in their standard formulation, this does not harm their mutual comparison. In section 2.3.1, we discussed the ability of RNN-based prediction networks to capture spatially dependent behavior changes. In order to make a fairer comparison for the real-world data evaluation, all tracklets are normalized to start at the origin.

The evaluation on the *Daimler* 2014 dataset is done in an ego-motion compensated reference system. The frame rate of the camera system inside the recording vehicle is 16 fps and it is adopted accordingly for our experiments. Pedestrians change their behavior abruptly. Therefore, sensible time horizons are short. Here, 8 (0.5 seconds) consecutive positions are observed, before predicting the next 8 (0.5 seconds), 12 (0.75 seconds), and 16 (1 second).

For generating synthetic trajectories of a basic maneuvering pedestrian, random agents are sampled from a Gaussian distribution according to a preferred pedestrian walking speed [Tek02] ($\mathcal{N}(1.38\ m/s, 0.37\ (m/s)^2)$) from the distribution of starting positions of the corresponding *Daimler* dataset sequences. The distribution of starting position is approximated with a mixture-of-Gaussians by using the expectation-maximization (EM) algorithm [Bis06] with five components. The chosen preferred walking speed corresponds approximately to *TrajNet* dataset analysis in section 4.2.1.

The RNN-IMM models have been implemented using *Tensorflow* [Aba15] and are trained for 2000 epochs using ADAM optimizer [Kin15] with a decreasing learning rate, starting from 0.01 with a learning rate decay of 0.95 and a decay factor of 1/10. For the experiments, the RNN variant LSTM [Hoc97] is used.

4.2.2.3 Scenario: *Crossing*

In the first scenario, the pedestrian intending either to cross or not to cross the street laterally is considered. During a single trajectory simulation, the agents head laterally and can perform a stopping maneuver or cross the street. Figure 4.17 illustrates such maneuvers with example images from the Daimler dataset [Sch13]. For mapping the pedestrian detections to a vehicle-motion compensated ground plane, Schneider et al. used onboard sensors for velocity and yaw rate, and a stereo camera system to compute the median disparity based on a dense stereo approach (semi-global matching) [Hir08].

Due to the non-linear observation model based on a perspective camera model, an inevitable linearized extension for the Kalman and IMM filter observation models is required. Here, the observation uncertainty is assumed to

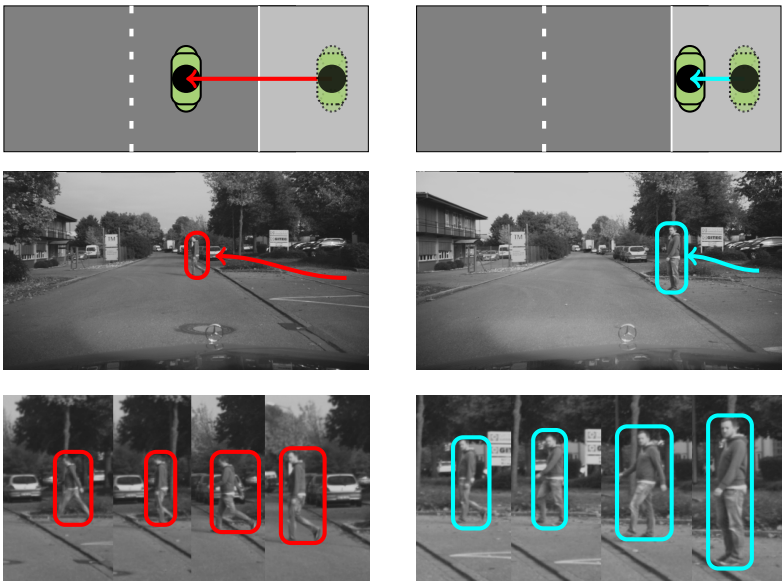

Figure 4.17: Illustration of typical pedestrian motions. The above images depict the two chosen maneuver classes of straight walking or crossing and stopping. The images on the left show a person crossing the street. The images on the right show a person transitioning from walking to standing at the curbside of the street. In particular changing from straight walking to stopping [Sch13].

be Gaussian distributed $\mathbf{w}^k \sim \mathcal{N}(0, (0.01\ m)^2)$ in the compensated reference system. Thus, the standard formulation of the Bayesian filters is well suited for this task. For the stopping maneuver or rather the event of deceleration until standing, a mean sojourn time of 1 second with a standard variation of 0.1 seconds is used. As a reminder, the mean sojourn time τ_{so} is the mean time an object stays in a motion type or dynamical mode [Sch13, Bar02]). Blackman [Bla99] suggests using this time to specify the TPM instead of

using the ad-hoc approach to fill the diagonals with values close to one. Thus, the model-to-model transition is set to $m_{ii} = 1 - {}^{\Delta T}/_{\tau_{so}}$ (see equation 3.37). The sojourn times are derived from the *Daimler* datasets.

As long as a person moves in a straight line at a nearly constant speed, their dynamics can be captured with a Kalman filter using a CV model. During the maneuver, the relation to one fixed motion model describing the dynamics fails due to an additional deceleration. Similar to Schneider et al. [Sch13] or Kooij et al. [Koo14], one reference IMM filter is set up by combining two basic models, in particular the CV and the CA model. As discussed in section 3.2.1, side effects due to independent motions in different directions are avoided by only considering the crossing direction or - from the vehicle perspective - the lateral motion. In other existing work, a combination of a CP model and a CV model is suggested because of the relatively short decelerating phase (see for example [Kel11]). Furthermore, an IMM filter with the three dynamic models of a CP, a CV, and a CA model is for example used in the work Goldhammer et al. [Gol14]. Thereby, a transition from straight walking to stopping is modeled in a physically plausible manner with a deceleration phase. Following the aforementioned explanations, the IMM-RNN is compared to the described IMM filters with two dynamical models ((CV, CA); (CP, CV)), an IMM filter with three dynamical models (CP, CV, CA), a Kalman filter with a single CV model, a Kalman filter with a single CA model, and as baseline to a linear interpolation.

Also correspondingly to Schneider et al., the process noise $\mathbf{v}^k$ is determined by $Q^k = Q_0^k q$, where $q \in \{\sigma_{CP}^2, \sigma_{CV}^2, \sigma_{CA}^2\}$ describes the changes in position, in velocity or respectively in acceleration over a sampling period ΔT. The covariance matrix Q_0 of a CV (white noise acceleration) [Bar02] model is given by

$$\mathbf{Q}_{CV}^k = \begin{bmatrix} 0.25\Delta T^4 & 0.5\Delta T^2 \\ 0.5\Delta T^2 & \Delta T^2 \end{bmatrix} \sigma_{CV}^2. \tag{4.28}$$

The physical dimension of σ_{CV} is that of an acceleration. For the CA (Wiener process acceleration) [Bar02] model

$$\mathbf{Q}_{CA}^{k} = \begin{bmatrix} 0.25\Delta T^4 & 0.5\Delta T^3 & 0.5\Delta T^2 \\ 0.5\Delta T^3 & \Delta T^2 & \Delta T \\ 0.5\Delta T^2 & \Delta T & 1 \end{bmatrix} \sigma_{CA}^2. \tag{4.29}$$

In this model, the process noise $\mathbf{v}^k$ is the acceleration increment during the kth sampling period. Based on the above process noise model, Schneider et al. [Sch13] estimated the process noise parameters for the different chosen filters (IMM filter (CV, CA), Kalman filter CV, CA) on the Daimler dataset. These noise parameters are for the IMM filter (CV, CA) $\sigma_{IMM,CV} = 0.7\ m/s^2$, $\sigma_{IMM,CA} = 0.8\ m/s^3$ and for the single Kalman filters $\sigma_{CV} = 0.77\ m/s^2$ and $\sigma_{CA} = 0.44\ m/s^3$. For the CP model, the noise parameter can be set based on the steady walking pace. For the IMM filter (CP, CV), we used the settings similar to Keller et al. [Kel11] ($\sigma_{IMM,CP} = 0.1\ m/s, \sigma_{IMM,CV} = 0.09\ m/s^2$). The noise parameters of the three model IMM filters are $\sigma_{IMM,CP} = 0.1\ m/s, \sigma_{IMM,CV} = 0.7\ m/s^2$, and $\sigma_{IMM,CA} = 0.8\ m/s^3$. These parameters are consistent with the suggested practical setting in Bar-Shalom [Bar02] and the chosen sojourn time for the simulation.

Compared to vehicle maneuvers, such as lane changing, a definition of maneuver classes for pedestrians is harder to establish. Since in most cases, the standard behavior of pedestrians is straight walking, the deviation from a standard behavior, and whether the pedestrian is in a *normal* mode is here detected. A fixed deviation in velocity, deceleration, along with the tangential ground truth trajectory is used to assign a maneuver label to a time step of a single trajectory. Thus, the RNN-IMM and IMM filters have a similar dynamical model set description. As the distribution over the trajectories for the RNN-IMM is captured with a Gaussian mixture model, the maneuver description for a single model can still be multi-modal. Since the IMM filter predicts a multi-modal distribution in the form of a combination of the unimodal model-specific predictions, the RNN-IMM is set to equally predict a uni-modal Gaussian distribution conditioned on a single maneuver class in the presented results.

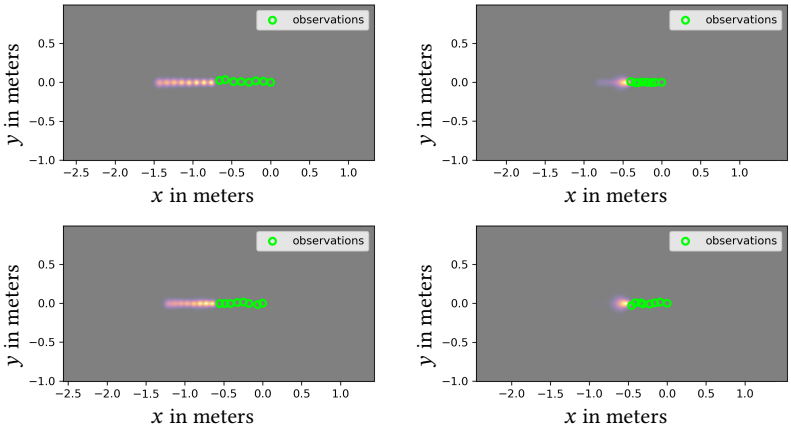

Figure 4.18: Visualization of the predicted multi-modal distributions of future position as heatmap. (Left) Density plots for crossing or rather straight walking examples. (Right) Density plots for stopping examples in which the maximum of the predicted distribution is visible close to the last observation.

In figure 4.18, predictions for two differently performed motion types are depicted for 8 future positions weighted by the predicted maneuver probability. In the shown images, the positions are normalized to start at the origin. The resulting multi-modal prediction is visualized as a heatmap. On the right, it can be seen that for a crossing sequence with straight walking, the RNN-IMM mainly uses the corresponding straight walking model. On the left, where the deceleration started, the straight walking probability is visibly lower, and the predicted distribution maximum is very close to the last observation. For the quantitative evaluation, 1000 noisy trajectories have been synthetically generated, where 80 % are used for training and 20 % for the comparison to the recursive Bayesian filters. The results are summarized in table 4.5.

The performance is compared using the final displacement error (FDE) (see for example [Pel09]) of the lateral motion (from the vehicle perspective) for three different time horizons, in particular 8 steps (0.5 seconds), 12 steps (0.75 seconds) and 16 steps (1 second). These results show that the presented

Table 4.5: Results for the comparison between an RNN-IMM and IMM filters with several dynamical model setups, Kalman filters with single models, and using linear interpolation on the simulated maneuver situations of *crossing* and *stopping*. The prediction is done for 8, 12, and 16 time steps conditioned on 8 observations for a frame rate of 16fps.

	8/8		8/12		8/16	
Approach	FDE / m	σ_{FDE} / m	FDE / m	σ_{FDE} / m	FDE / m	σ_{FDE} / m
RNN-IMM	0.0309	0.0404	0.0427	0.0817	0.0517	0.0941
IMM filter (CP, CV, CA)	0.0612	0.0606	0.113	0.130	0.1736	0.1901
IMM filter (CV, CA)	0.0674	0.0602	0.1188	0.1255	0.1862	0.1915
IMM filter (CP, CV)	0.1073	0.0916	0.2031	0.1623	0.3101	0.214
Kalman filter (CA)	0.0796	0.0638	0.1575	0.1137	0.2386	0.1696
Kalman filter (CV)	0.1578	0.1601	0.2890	0.2965	0.4701	0.4700
Linear interpolation	0.1587	0.1610	0.2903	0.2978	0.4724	0.4718

RNN-IMM is able to capture the changing varying dynamics for the synthetically generated data faster. In terms of the single motion models (CV versus CA), one can observe the benefits for the CA in capturing the deceleration. Since pedestrian acceleration or declaration phases are relatively short due to pedestrians quickly reaching their preferred walking speed or a stopping state [Gol17], the CA and higher-order dynamical models can lead to high prediction errors for longer time horizons. In order to capture the maneuver, the switch to other models is beneficial. For the crossing and stopping simulation, the IMM filters show an overall improvement over the single models. The best result is achieved with the three model IMM filter. The RNN-based IMM filter surrogate is able to capture the switch to a stopping mode. The engineering task of finding the best model set for IMM filters and their extensions can lead to improved behavior (see for example Keller et al. [Kel14]) in specific maneuver situations, but it is also a very tedious process to find a good setting. As discussed, recent work like the approaches of Kooij et al. [Koo19] show options how to further improve the prediction performance of IMM filters by using a DBN on top and thereby including scene context and more cues than pedestrian point kinematics (e.g. head orientation, gaze, body tilt, articulated body information). The integrated cues and accordingly additional latent states modify only the transition probability between single dynamical models, which is not required for the RNN-IMM.

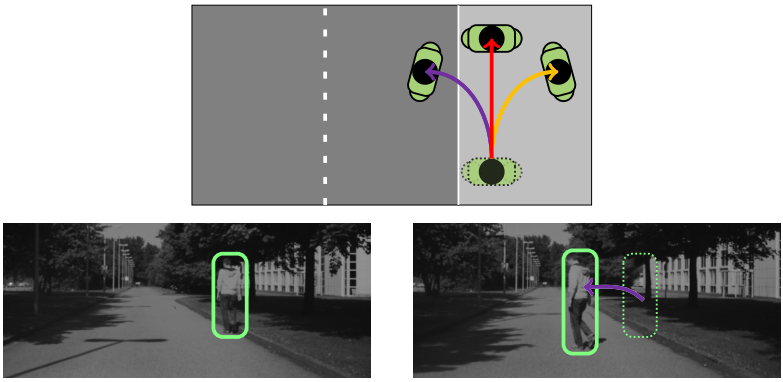

Figure 4.19: Illustration of a typical pedestrian maneuver. The above images depict a change from straight walking to turning and thus a sudden crossing of the street (*bending in*).

However, the presented RNN-IMM is able to also provide a confidence value $P(m_i|\mathcal{Z}) \stackrel{\wedge}{=} \alpha_i$ for the performed dynamics, but without an explicit modeling of the dynamics transitions in the form of a fixed TPM. Similar to the provided mode probabilities of IMM filters, this can be used for subsequent processing stages (see for example our works [Bec15, Mün16a, Mün16b]). Further, instead of choosing the basic filter set, the prediction model is learned. In case there exists some well-known model for describing the standard dynamics of the desired object, only deviations from the known dynamics can be used to define additional maneuver classes.

4.2.2.4 Scenario: *Turning*

Another prototypical maneuver performed by a pedestrian, which is of keen interest in the context of intelligent vehicles, is a turning maneuver. For

such a maneuver, the pedestrian's dynamics changes from a straight walking to a *bending in* behavior. Similar to the simulation of the crossing/stopping scenario, for simulating a basic maneuvering pedestrian, random agents are sampled from a Gaussian distribution in accordance with common pedestrian walking speeds [Tek02] ($\mathcal{N}(1{,}38\ ^m/_s, (0.37\ ^m/_s)^2)$) from the distribution of starting positions of annotated *bending in* sequences from the *Daimler path prediction* dataset. Hence, the same fixed frame rate of 16 fps is used. During a single trajectory simulation, the agents can perform a turning maneuver. The change in heading is sampled from an uniform distribution between 45° and 100°. The duration of the turning event is sampled from a Gaussian distribution based on the mean sojourn time estimated from the ground truth sequences ($\mathcal{N}(1.83s, (0.29s)^2)$). For comparing to common *physics*-based dynamical models, the simulation is done on ground level. In this scenario, the longitudinal motion is also crucial to capture such a maneuver. In addition to the above chosen filters, Keller et al. suggest a combination of a *CV* model and a *CT* model as elemental filters to model the switching behavior. The corresponding joint state vector of the CT model with the turning rate ω can be expressed as

$$\mathbf{x}_{CT}^k = [x, y, \dot{x}, \dot{y}, \omega]^\mathsf{T}. \tag{4.30}$$

Since the model is non-linear, the estimation of the state is done via an EKF [Bar02]. The corresponding dynamical model for state prediction is given by

$$\mathbf{x}_{CT}^{k+1} = \underbrace{\begin{bmatrix} 1 & \frac{\sin(\omega^k \Delta T)}{\omega^k} & 0 & \frac{\cos(\omega^k \Delta T - 1)}{\omega^k} & 0 \\ 0 & \frac{1-\cos(\omega^k \Delta T)}{\omega^k} & 1 & \frac{\sin(\omega^k \Delta T)}{\omega^k} & 0 \\ 0 & \cos(\omega^k \Delta T) & 0 & -\sin(\omega^k \Delta T) & 0 \\ 0 & \sin(\omega^k \Delta T) & 0 & \cos(\omega^k \Delta T) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{F_{CT}^k} \mathbf{x}_{CT}^k + \mathbf{G}_{CT}^k \mathbf{v}^k. \tag{4.31}$$

Despite the matrix form, the model can be equivalently expressed as a set of equations f_{CT}. To use an EKF for estimating, the Jacobian $\mathbf{J}_{CT}^k = \frac{\delta f_{CT}}{\delta \mathbf{x}_{CT}}|_{\hat{\mathbf{x}}_{CT}^{k,+}}$ of the dynamical equations must be computed for calculating the transition for the state covariance.

$$\mathbf{J}_{CT}^k = \begin{bmatrix} 1 & \frac{\sin(\omega^k \Delta T)}{\omega^k} & 0 & \frac{\cos(\omega^k \Delta T - 1)}{\omega^k} & \frac{\delta x}{\delta \omega}|_{\hat{\mathbf{x}}_{CT}^{k,+}} \\ 0 & \frac{1-\cos(\omega^k \Delta T)}{\omega^k} & 1 & \frac{\sin(\omega^k \Delta T)}{\omega^k} & \frac{\delta \dot{x}}{\delta \omega}|_{\hat{\mathbf{x}}_{CT}^{k,+}} \\ 0 & \cos(\omega^k \Delta T) & 0 & -\sin(\omega^k \Delta T) & \frac{\delta y}{\delta \omega}|_{\hat{\mathbf{x}}_{CT}^{k,+}} \\ 0 & \sin(\omega^k \Delta T) & 0 & \cos(\omega^k \Delta T) & \frac{\delta \dot{y}}{\delta \omega}|_{\hat{\mathbf{x}}_{CT}^{k,+}} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{\hat{\mathbf{x}}_{CT}^{k,+}} \tag{4.32}$$

with the partial derivatives with respect to the turning rate

$$\begin{aligned} \frac{\delta x}{\delta \omega}|_{\hat{\mathbf{x}}_{CT}^{k,+}} &= \frac{\omega^k \Delta T \cos(\omega^k \Delta T) - \sin(\omega^k \Delta T)}{{\omega^k}^2} \dot{x}^k \\ &\quad - \frac{\omega^k \Delta T \sin(\omega^k \Delta T) + \cos(\omega^k \Delta T - 1)}{{\omega^k}^2} \dot{y}^k \\ \frac{\delta \dot{x}}{\delta \omega}|_{\hat{\mathbf{x}}_{CT}^{k,+}} &= -\Delta T \sin(\omega^k \Delta T) \dot{x}^k - \Delta T \cos(\omega^k \Delta T) \dot{y}^k \\ \frac{\delta y}{\delta \omega}|_{\hat{\mathbf{y}}_{CT}^{k,+}} &= \frac{\omega^k \Delta T \sin(\omega^k \Delta T) + \cos(\omega^k \Delta T)}{{\omega^k}^2} \dot{x}^k \\ &\quad - \frac{\omega^k \Delta T \cos(\omega^k \Delta T) - \sin(\omega^k \Delta T - 1)}{{\omega^k}^2} \dot{y}^k \\ \frac{\delta \dot{y}}{\delta \omega}|_{\hat{\mathbf{x}}_{CT}^{k,+}} &= -\Delta T \cos(\omega^k \Delta T) \dot{x}^k - \Delta T \sin(\omega^k \Delta T) \dot{y}^k. \end{aligned} \tag{4.33}$$

For the noise process model used by Schneider et al., the process noise covariance matrix has the form

$$\mathbf{Q}_{CT}^{k} = \begin{bmatrix} 0.25\Delta T^4 & 0.5\Delta T^2 & 0 & 0 & 0 \\ 0.5\Delta T^2 & \Delta T^2 & 0 & 0 & 0 \\ 0 & 0 & 0.25\Delta T^4 & 0.5\Delta T^2 & 0 \\ 0 & 0 & 0.5\Delta T^2 & \Delta T & 0 \\ 0 & 0 & 0 & 0 & \frac{\sigma_{CT}^2 \Delta T^2}{\sigma_{CV}^2} \end{bmatrix} \sigma_{CV}^2. \tag{4.34}$$

Here, σ_{CT}^2 is the turning rate variance and σ_{CV}^2 corresponds to a random acceleration (see equation 4.28). For the CT model as part of the IMM (CV, CT) filter, the noise parameters are chosen according to Schneider et al. [Sch13] ($\sigma_{IMM,CV} = 0.4\ ^{m}/_{s^2}$, $\sigma_{IMM,CT} = 0.9\ ^{rad}/_{s^2}$). The noise parameters for the other dynamical models are chosen as in section 4.2.2.3. As discussed in section 3.2.1, using the proposed de-coupled IMM filter can be beneficial due to independent motion along a particular direction. Hence, a de-coupled IMM filter with similar parameter setup is additionally used for comparison.

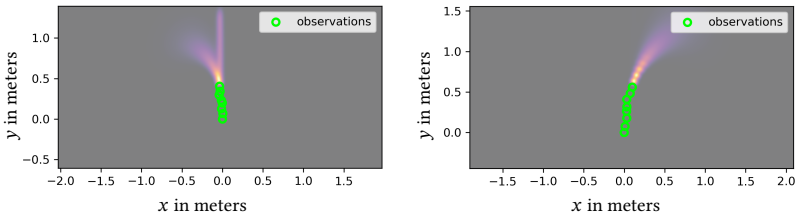

Figure 4.20: Visualization of the predicted multi-modal distributions of future position as heatmap for the *bending in* scenario. Two density plots showing different points in time during the turning maneuver.

In case of turning, the motion changes from a rectilinear dynamics to a curvilinear motion, in relation to the dynamics this results in an additional acceleration or rather change in heading. Therefore, a change from a constant

velocity model to a turning model or acceleration model indicates a critical situation from the vehicle perspective. Figure 4.19 illustrates such a turning or *bending in* maneuver.

Since the standard behavior of pedestrians is straight walking, a fixed deviation in heading for a required time-horizon is used to assign maneuver labels to single trajectories. The RNN-IMM outputs a multi-modal path distribution based on the one-hot encoded maneuver classes. Here, the conditioning is done for the three maneuver classes of turning left, turning right and straight walking. For the synthetic *bending in* scenario, the maneuver distributions could be captured with one Gaussian component even though the trajectory distribution can still be multi-modal by using an MDN with more components. In figure 4.20, example multi-modal predictions for the *bending in* maneuver scenario are visualized. The predicted density is visualized as a heatmap with normalized positions starting at the origin. The two example images show two turning situations at different points in time. On the left, the straight walking probability is still dominant. On the right, the RNN-IMM captures the maneuver and predicts a clockwise change in heading.

Table 4.6: Results for the comparison between an an RNN-IMM and IMM filters with different dynamical model structures, a Kalman filter with a single CV model, a Kalman filter with a single CA model, and using linear interpolation on the simulated *bending in* maneuver situations. The prediction is done for 8, 12, and 16 time steps conditioned on 8 observations for a frame rate of 16 fps.

	8/8		8/12		8/16	
Approach	FDE / m	σ_{FDE} / m	FDE / m	σ_{FDE} / m	FDE / m	σ_{FDE} / m
RNN-IMM	0.1009	0.1066	0.1833	0.2073	0.2479	0.3387
IMM filter (CV, CA)	0.1641	0.1068	0.3274	0.2131	0.5109	0.3732
IMM filter (CV, CA; de-coupled)	0.1626	0.1248	0.3119	0.2519	0.5055	0.4138
IMM EKF (CT, CV)	0.1988	0.1575	0.3482	0.2745	0.5102	0.4227
Kalman filter (CA)	0.1809	0.1145	0.3664	0.2497	0.5344	0.3257
Kalman filter (CV)	0.2098	0.1707	0.3729	0.3007	0.5301	0.4523
Linear interpolation	0.2530	0.2084	0.4326	0.3543	0.6122	0.5100

For training and evaluation, 1000 noisy trajectories have been synthetically generated with a split of using 80% for training and 20% for evaluation. The

results are summarized in table 4.6. For comparison, the final displacement error (FDE) is calculated as average L2 distance between the predicted final positions and the ground truth positions for three different time-horizons.

As before, the proposed RNN-IMM achieves the best result for the simulated scenario and is able to capture the switch to another motion type. The IMM filter solutions perform better than single model Kalman filters. Comparing the different model set structures of the IMM filters, our de-coupled IMM filter yields slightly better results, but with no significant difference. For the longer time-horizon, the effect of curvilinear motion is more pronounced, thus the benefit of the RNN-IMM is more visible, and the inclusion of the CT model in the filter setup has a more positive effect. The CT model and its variants are more common for being utilized for tracking other road users, such as bikes and vehicles (see for example [Koo19, Bat09]) or capturing the turning maneuver of tracked air-crafts [Li03]. Solely the amount of existing *physics*-based models and model set combinations clearly show why the trend is to shift to a *pattern*-based alternative. Also, in the *bending in* scenario, the RNN-based IMM filter surrogate is able to capture the switch in modes without the engineering task of finding the best model set for the IMM filter.

4.2.2.5 Scenario: Real-World Data

For the real-world data scenario, the evaluation is done on the *Daimler context path prediction* dataset [Koo14] consisting of 58 sequences recorded with the described sensor setup in an ego-motion compensated reference system. All sequences involve individual pedestrians intending to cross the street or to stop at the curbside (*crossing* and *stopping* maneuvers). The sequences are further labeled with **time-to-event** (TTE) (in frames) information to focus on the critical situations. For stopping pedestrians, the frame when the last foot of the pedestrian is placed on the ground of the curbside is labeled with TTE$=0$ and for crossing pedestrians, the closest point before stepping on the road. Only the lateral motion and frames between TTE< -15 and TTE> 15 are considered. 5-fold cross-validation is done for tracklets capturing the time windows of 16, 20, and 24 consecutive position, where 8 positions are

used for initialization. This matches with the setting of the simulated *crossing* sequences. In order to reduce side effects due to limited training samples, additional sequences are augmented to extend the total amount of training sequences to 200. According to the estimated lateral observation error, the observation noise is set to $\mathbf{w}^k \sim \mathcal{N}(0, (0.06\ m)^2)$ in the vehicle-motion compensated reference system. The evaluation of the real data is done on 20 % of the real-world tracklets. Table 4.7 shows the final displacement errors for the lateral positions for three time-horizons. In table 4.8 and 4.9 the results are split into the *crossing* and *stopping* sequences.

Table 4.7: Results for the comparison between an RNN-IMM and several filters, including different IMM filters and single Kalman filter on the *Daimler context path prediction* dataset [Koo14]. The prediction is done for 8, 12, and 16 time steps conditioned on 8 observations.

	8/8		8/12		8/16	
Approach	FDE / m	σ_{FDE} / m	FDE / m	σ_{FDE} / m	FDE / m	σ_{FDE} / m
RNN-IMM	0.0811	0.1165	0.1244	0.2076	0.2137	0.3313
IMM filter (CP, CV)	0.1609	0.1495	0.2746	0.2518	0.3881	0.3711
IMM filter (CP, CV, CA)	0.1721	0.1688	0.3060	0.2753	0.4202	0.3925
IMM filter (CV, CA)	0.1792	0.1641	0.3242	0.2790	0.4602	0.4178
Kalman filter (CA)	0.2061	0.2240	0.5260	0.4055	0.8112	0.6300
Kalman filter (CV)	0.1618	0.1507	0.2749	0.2519	0.3885	0.3711
Linear interpolation	0.1628	0.1511	0.2773	0.2541	0.3918	0.3745

As reference models, the IMM and Kalman filters with the corresponding dynamical models as described in section 4.2.2.3 are used. Also on the real-world sequences, the RNN-IMM achieves the best performance. For the comparison of the different model set configuration, the difficulty of choosing the best configuration becomes clearly visible. Despite the fact that the performance difference is relative small, including a CP or CA helps to better capture the stopping behavior. As expected, in pure *crossing* sequences a Kalman filter with single CV model performs well. Due to fact that the pedestrians walk with a relatively constant speed, the second-order CA model interprets noisy observations as additional acceleration and thus performs worse than the first-order CV model. The IMM filters are able to better handle this situation. Due

to larger observation noise in the real-world sequences, the distinction between the maneuver situation and finding the best dynamical model combination is more difficult. As shown, the amount of different proposed dynamical model structures in combination with the general problem of finding suited *physics*-based model, show the benefit of capturing switching dynamics with RNN-based solutions. In figure 4.21 and figure 4.22, the predicted density distributions with an RNN-IMM are visualized for a *crossing* and respectively a *stopping* maneuver for 8 future steps. The lengths of the colored bars above the pedestrians depict the model probabilities. For the visualization, the predicted distribution is mapped back to image space using the calibration information. The future distributions are obtained by sampling from the predicted distribution. In the crossing situation, the model classifies the straight walking behavior correctly and predicts *crossing* with a high probability. In figure 4.22, the switching behavior is highlighted. In the images on the left, the declaration begins but straight walking is still dominant. In the images in the middle, the selected model has changed and *stopping* behavior is classified. The predicted positions are closer to the current positions. On the right, the pedestrian stands at the curbside and the RNN-IMM recognizes the situation correctly. Thus, the predicted density is very close to the current position. Although the observation noise level is larger than for the synthetic scenarios, the RNN-IMM tends to be overconfident towards one mode. This effect is further analyzed in the next section for simulated trajectories under better-controlled conditions.

Table 4.8: Evaluation results for the *crossing* sequences of the *Daimler context path prediction* dataset [Koo14]. The final displacement error is shown for predicting 8, 12, and 16 steps into the future.

Crossing sequences						
	8/8		8/12		8/16	
Approach	FDE / m	σ_{FDE} / m	FDE / m	σ_{FDE} / m	FDE / m	σ_{FDE} / m
RNN-IMM	0.0841	0.1219	0.1224	0.2076	0.1978	0.3174
IMM filter (CP, CV)	0.1490	0.1387	0.2522	0.2405	0.3456	0.35197
IMM filter (CP, CV, CA)	0.1601	0.1640	0.3160	0.2798	0.4202	0.3946
IMM filter (CV, CA)	0.1492	0.1435	0.3217	0.2809	0.4450	0.4201
Kalman filter (CA)	0.1906	0.16623	0.5764	0.4090	0.8770	0.6400
Kalman filter (CV)	0.1491	0.1387	0.2523	0.2406	0.3460	0.3520
Linear interpolation	0.1526	0.1442	0.2545	0.2427	0.3490	0.3553

Table 4.9: Evaluation results for the *stopping* sequences of the *Daimler context path prediction* dataset [Koo14]. The final displacement error is shown for predicting 8, 12, and 16 steps into the future.

Stopping sequences						
	8/8		8/12		8/16	
Approach	FDE / m	σ_{FDE} /m	FDE / m	σ_{FDE} / m	FDE /m	σ_{FDE} / m
RNN-IMM	0.0693	0.0934	0.1316	0.2075	0.2766	0.3755
IMM filter (CP, CV)	0.1890	0.1627	0.3588	0.2742	0.5207	0.4027
IMM filter (CP, CV, CA)	0.1581	0.1450	0.2886	0.2542	0.4404	0.384
IMM filter (CV, CA)	0.1797	0.1602	0.3336	0.2715	0.5567	0.3959
Kalman filter (CA)	0.1799	0.1784	0.3376	0.3254	0.5507	0.5114
Kalman filter (CV)	0.1897	0.1628	0.3592	0.2745	0.5572	0.3963
Linear interpolation	0.1990	0.1685	0.3627	0.2768	0.5619	0.3993

Figure 4.21: Visualization of RNN-IMM predictions for *crossing* sequences from the *Daimler* dataset. The model probability is depicted with the length of the colored bars above the pedestrian (*crossing* ■ and *stopping* ■). By sampling from the predicted distribution and mapping back to the vehicle view, the predictive density is visualized. In the above sequences, the RNN-IMM correctly identifies the *crossing* mode with a high probability.

Figure 4.22: Visualization of RNN-IMM predictions for *stopping* sequences from the *Daimler* dataset. The model probability is depicted with the length of the colored bars above the pedestrian (*crossing* ■ and *stopping* ■). By sampling from the predicted distribution and mapping back to the vehicle view, the predictive density is visualized. The transition from straight walking to stopping and the corresponding change of the mode probabilities can be seen in the above sequences.

4.2.2.6 Scenario: *Intention* Classification

Table 4.10: *Intention* classification results for a comparison between an IMM filter and the proposed RNN-IMM for generated pedestrian trajectories with a Markovian switch between a CV and CA model for increasing noise levels. The firs column shows the sensitivity and the second column the average predicted mode probability of correct classified dynamical modes.

	sensitivity / average true mode probability									
Approach	$\sigma_w = 0.001$		$\sigma_w = 0.005$		$\sigma_w = 0.01$		$\sigma_w = 0.05$		$\sigma_w = 0.1$	
RNN-IMM-Mode	0.946	0.963	0.818	0.839	0.7606	0.787	0.629	0.669	0.587	0.633
IMM filter (CV, CA)	0.893	0.835	0.767	0.745	0.703	0.613	0.619	0.589	0.583	0.554

For the experiments so far, pedestrian *intention prediction* is considered jointly with *path prediction*. In this section, it is considered as a pure classification task. In order to better control the conditions for training and the test scenario, again synthetic data is used. For the real-world sequences as well as for the simulated scenarios in this section, the pedestrian trajectories are considered purely deterministic trajectories which consist of fixed-length non-maneuvering and maneuvering motion. Although this is common and reasonable, the underlying assumption of *hybrid* state systems such as the IMM filter is that the true dynamical mode is modeled as a Markov-chain with discrete dynamical modes. In this section, a pedestrian tracking scenario with switching dynamics is simulated as a probabilistic trajectory with random Markovian transitions. The scenario of pedestrian motion orientated on the *Daimler* dataset statistic is kept. Both for reference IMM and as well for generating sample trajectories, the combination of a CV model and a CA restricted to lateral motion is used. In the experiments, the observation noise is increased step-wise, and the classification accuracy of the RNN-IMM is compared to its IMM filter counterpart. For the simulation, the dynamics of a pedestrian agent can switch between the two models with a switching probability of $P(m_i^k|m_j^{k-1}) = 0.1 \quad (j \neq i)$ or stay in its dynamical mode with $P(m_i^k|m_i^{k-1}) = 0.9$. The acceleration during a CA phase is sampled from a Gaussian distribution ($\mathcal{N}(1\ ^m/_{s^2}, (0.1\ ^m/_{s^2})^2)$). Drawn decelerations are rejected. The starting velocity is again sampled from a distribution of common

walking speeds in accordance with Teknomo [Tek02]. The velocity of a simulated agent can exceed the maximum of the physically possible velocity of pedestrians to prevent an undesired switch to CV model due to a constant maximum value. Thereby, the conditions for the IMM with the two corresponding dynamic models are idealized. The process noise is modeled with the two noise models explained in section 4.2.2.3. For comparison, the true positive dynamics classification rate, also referred to as sensitivity, is used. The predicted model probability for the current time step is compared with the known ground truth mode. The first 8 steps are excluded for filter initialization. In table 4.10, the results for an increasing observation noise are summarized. The frame rate is set to 16 fps and the process noise parameters to $\sigma_{CV} = 0.01\ ^{m}/_{s^2}$ and $\sigma_{CA} = 0.01\ ^{m}/_{s^3}$. For training and evaluation, 1000 trajectories of 4 seconds with random Markovian transitions between a CV model and CA model are generated. The evaluation is done on 20 % of the generated samples. Alongside the sensitivity, the average mode probability for a correctly classified dynamics is shown. Although the conditions are suited for an IMM filter, the proposed RNN-IMM can better classify the true dynamical mode from the noisy observations. For increasing observation noise levels, the sensitivity decreases to less than 60 % correctly classified dynamical modes for both models. The results show that the RNN-IMM can faster switch to other dynamics. However, a distinction between the selected dynamical models gets more difficult for larger noise levels. The IMM filter assigns each model a similar probability, and thus both models are equally used to estimate the dynamical state. The predicted mode probability of the IMM filter fits better to a sensitivity close to 0.5. Hence, the RNN-IMM tends to be overconfident with the assigned dynamical model probabilities. The model probabilities over time for both models are visualized in figure 4.23.

The true dynamics for a particular time step is indicated with the background color. The CA model is visualized with dark blue (▬) and the CV model with dark yellow (▬). The IMM filter probabilities $P(\cdot)_{IMM}$ are shown as a dotted line and the RNN-IMM probabilities $P(\cdot)_{RNN}$ as a solid line. The example results demonstrate the effect of stronger combining the selected dynamical models to describe the dynamical behavior of the object with an IMM filter.

The RNN-based solution, on the other hand, makes a harder decision towards one model.

4.2.2.7 Summary

The presented results demonstrate the ability of the proposed RNN-based IMM surrogate to deal with switching motion behavior of a tracked object. For the exemplary task of *intention prediction* in the context of intelligent vehicles, the RNN-IMM and the IMM filter counterpart are realized as a top-down component as part of a visual tracking system. By using a stereo camera system, applying semi-global matching [Hir08] and correcting the ego-motion, the detections are mapped to a physical reference system. Thus, the IMM filter can rely on prior knowledge of the dynamics of pedestrian motion provided by several proposed *physics*-based dynamical models or rather model sets. Under these conditions, the RNN-IMM is able to recognize the change in motion type faster and achieves a better performance for jointly estimating future path or for a pure classification task. The model capabilities were shown on synthetic data and real-world data, both reflecting typical pedestrian maneuvers. For comparison, both approaches describe the motion of pedestrian point kinematics. It is clear as a basic principle, *pattern*-based methods are better suited to integrate more *contextual* cues. Hence, without the restriction of sufficient training data and the fast-growing body on datasets for intelligent vehicles, the RNN-IMM offers much more potential for extension (see section 2.3). Compared to the traditional IMM filter, the amount of engineering is reduced since the transition probabilities are not explicitly modeled. Another reason for the reduced engineering effort is that for the RNN-IMM, a maneuver is simply modeled as a deviation from standard straight walking behavior without modeling dynamical model combinations.

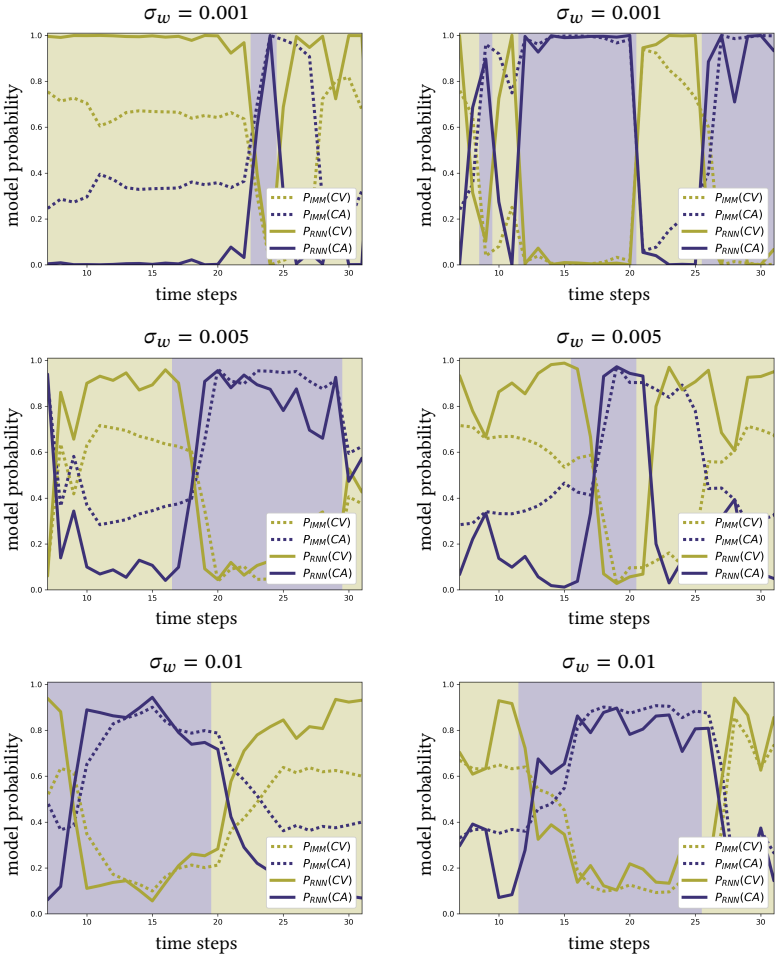

Figure 4.23: Visualization of the mode probabilities of an IMM filter and an RNN-IMM model for simulated trajectories with Markovian transition behavior.

4.2.3 Tracklet Alignment with a Minimum Variance Prototype

So far, we discussed the effect of hard-coded normalization to reduce the variation in translation and rotation of pooled tracklet data in order to enable a better generalization across datasets. Since the *object* motion cues are independent of these transformations, the amount of required modeling of object motion states can be reduced. Nevertheless, by applying these normalizations on tracklet data, the reference point and the reference rotation are arbitrarily chosen. Hence, the variation is eliminated from the references and just shifted along the tracklet. This makes clustering of such tracklets very challenging. In addition, rotation normalization is sensitive to out-of-distribution input tracklets. When using just two observations for estimating the rotation angle, the error in rotation depends on the distance between the two observations. Thus, using the first two observations can lead to high rotation error. Using observations which lie further apart relies on the assumption that the dynamics do not change between both observations.

In this section, a neural network solution that learns to align the input tracklets is proposed. The alignment network learns the required transformation of the input tracklets to achieve an optimal matching with an adjustable prototype. Instead of an arbitrarily chosen reference point and a reference rotation, a reference tracklet - the prototype - that reflects the minimum variance in the input data is learned. The alignment network enables to assess the training data by analyzing the prototype. Furthermore, the distance to the learned prototype can be used for clustering or identifying out-of-distribution tracklets.

The analysis of the alignment network is done on synthetically generated trajectories reflecting different dynamical behaviors. In addition, the path prediction data from the *BIWI* sequences [Pel09] and the *UCY* sequences [Ler07] is used.

4.2.3.1 Alignment Network

The alignment network can be combined with a prediction network by framing the prediction network with two transformation networks (forward and backward network). Besides the transformation networks, the central block to align the tracklets is the prototype network. In the following, it is referred to as bottleneck network. The bottleneck network consists only of a freely adjustable prototype tracklet $\mathcal{Z}^{\star}_{proto}$ without additional learnable parameters. This prototype tracklet is a sequence of randomly chosen points $\{(x^k, y^k) \in \mathbb{R}^2 | k = 1, \dots, k_{proto}\}$ of k_{proto} time steps. The bottleneck network can be defined as follows:

$$\hat{\mathcal{Z}}^{\star}_{proto} = \text{Bott}(\Theta_{Bott}). \tag{4.35}$$

Here, Θ_{Bott} are the adjustable prototype points. The idea is to find the best possible alignment with the prototype tracklet by simultaneously adjusting the prototype and learning the transformation parameters for the input tracklets. The resulting prototype represents a tracklet with minimum variance. In other words, it reflects the dominant input tracklet structure. In order to align the input tracklets with the prototype, the tracklets are transformed by transformation networks. The forward network ($\text{FW} : \mathbf{z} \rightarrow \hat{\mathbf{z}}^{\star}$) transforms the input trajectory to another trajectory space conditioned on the input tracklet. Thereby, the affine transformations of translation, rotation, and scaling are successively applied. Since the *object* motion cues are independent of these transformations, the estimate of the *object* motion state can be applied in transformed trajectory space. The forward network can be defined as follows:

$$\begin{aligned} \hat{\mathbf{t}}_{\mathcal{Z},FW}, \hat{r}_{\mathcal{Z},FW}, \hat{s}_{\mathcal{Z},FW} &= \text{MLP}(\mathbf{z}^{1:k_{obs}}; \Theta_{FW}), \\ \hat{\mathbf{z}}^{\star,1:k_{obs}} &= \text{FW}(\mathbf{z}^{1:k_{obs}}, \hat{\mathbf{t}}_{\mathcal{Z}}, \hat{r}_{\mathcal{Z}}, \hat{s}_{\mathcal{Z}}). \end{aligned} \tag{4.36}$$

The terms $\mathbf{t}_{\mathcal{Z}}, r_{\mathcal{Z}}, s_{\mathcal{Z}}$ are the learned translation, rotation, and scale for a given input tracklet $\mathbf{z}^{1:k_{obs}}$ of a fixed observation window of k_{obs} time steps. The parameters of the forward network and the prototype tracklet are learned by

minimizing the mean squared error between the transformed input tracklets and the prototype. The loss for a set of N input tracklets can be defined as

$$\mathcal{L}_{Bott}(\Theta_{Bott},\Theta_{FW}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{\mathbf{z}}_i^{\star,1:k_{obs}} - \hat{\mathcal{Z}}_{proto}^{\star})^2. \tag{4.37}$$

Alternatively, the Huber loss function [Hub64] or M-estimators [Ham11] can be used. The aligned tracklets can then serve as input for the filtering or prediction model. An RNN-based prediction model for the next k_{pred} steps can be defined as

$$\begin{aligned} \mathbf{h}_{enc}^{k_{obs}} &= \text{RNN}(\mathbf{h}_{enc}^{k_{obs}-1},\hat{\mathbf{z}}^{\star,k_{obs}};\Theta_{enc}),\\ \hat{\mathbf{z}}^{\star,k_{obs}+1:k_{obs}+k_{pred}} &= \text{MLP}(\mathbf{h}_{enc}^{k_{obs}};\Theta_{pred}). \end{aligned} \tag{4.38}$$

The estimated next steps in the transformed tracklet space are mapped back to the original input space by the backward network (BW : $\hat{\mathbf{z}}^{\star} \rightarrow \hat{\mathbf{z}}$). The backward network performs the affine transformations in reverse order.

$$\begin{aligned} \hat{\mathbf{t}}_{\mathcal{Z},BW}, \hat{r}_{\mathcal{Z},BW}, \hat{s}_{\mathcal{Z},BW} &= \text{MLP}(\mathbf{z}^{1:k_{obs}};\Theta_{BW}),\\ \hat{\mathbf{z}}^{k_{obs}+1:k_{obs}+k_{pred}} &= \text{BW}(\hat{\mathbf{z}}^{\star,k_{obs}+1:k_{obs}+k_{pred}},\hat{\mathbf{t}}_{\mathcal{Z},BW}, \hat{r}_{\mathcal{Z},BW}, \hat{s}_{\mathcal{Z},BW}) \end{aligned} \tag{4.39}$$

Accordingly, the actual prediction is done in the transformed space, where all input tracklets are ideally in alignment with the prototype in a scene-independent reference system. Thus, extensive deviations from the prototype tracklets can be used to identify out-of-distribution input tracklets, which can lead to poor prediction results. Further, the prototype reflects the minimum variation of the dynamical behavior and enables to draw conclusions to the dominating dynamics in the training samples. The observed sequence can also be used as input for the backward network to incorporate spatial-dependent *contextual* cues into the overall network. Instead of only applying the inverse transformations from the forward network, the prediction can by further adapted. With this proposed cascade of successive transformation

and estimation steps, a distinction between the spatial-context and temporal-context is realized. Similarly to sections discussed above, the prediction or filtering network can be trained by minimizing the loss in form of the negative log-likelihood of the ground truth positions under the predicted positions or using the mean squared error between ground truth positions and the predicted positions. A combined network of an alignment network together with a prediction network is visualized in figure 4.24.

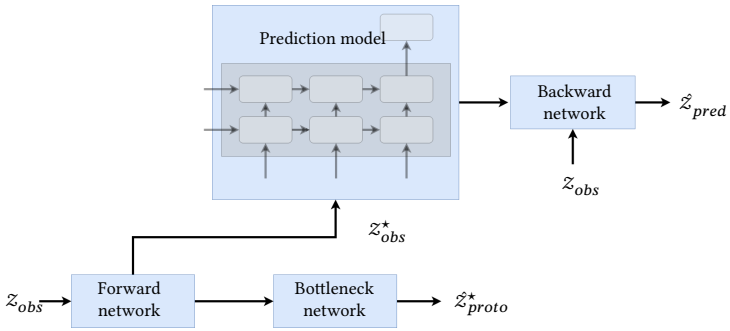

Figure 4.24: Visualization of the alignment network with an integrated RNN-based prediction network. The forward-network transforms the input tracklet $\mathcal{Z}_{obs} = \mathbf{z}^{1:k_{obs}}$ to align with the prototype tracklet. Prediction is performed in the transformed tracklet space. Then the tracklets are transformed in reverse order to the original tracklet space $\mathcal{Z}_{pred} = \mathbf{z}^{k_{obs}+1:k_{obs}+k_{pred}}$ by the backward-network.

4.2.3.2 Evaluation

The abilities of the alignment network are analyzed on synthetic data with one specific dynamical behavior. For real-world data, the *path prediction* sequences from the *BIWI* sequences [Pel09] and the *UCY* sequences [Ler07] are chosen.

The alignment models have been implemented using *Tensorflow* [Aba15] and are trained for 20000 epochs using ADAM optimizer [Kin15] with an initial

leaning rate of 0.0003. For the experiments, the forward network is realized with two hidden layers of size 50.

4.2.3.3 Scenario: Synthetic Data

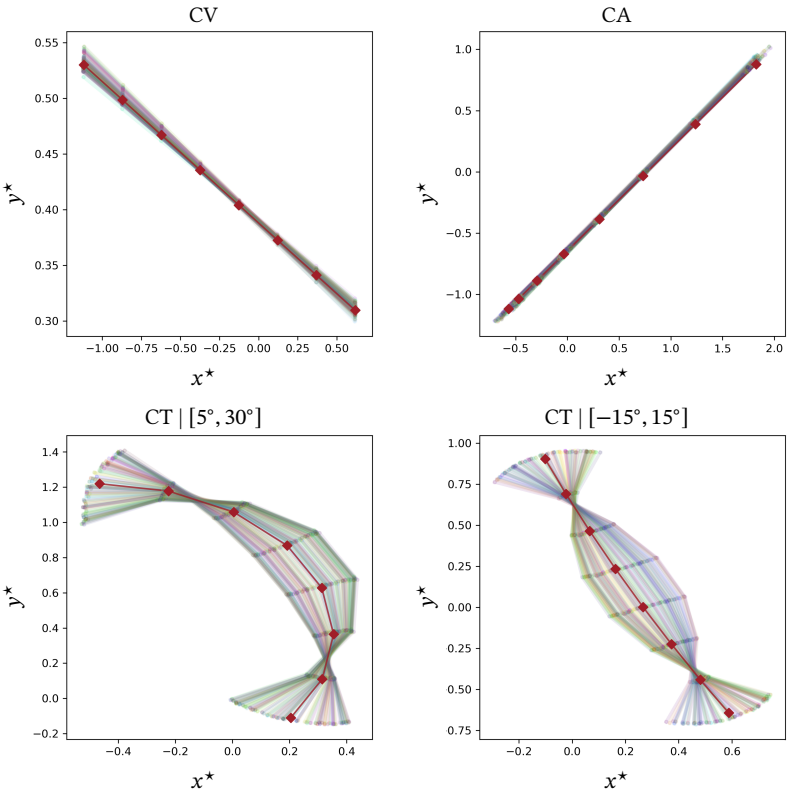

Figure 4.25: Visualization of aligned tracklets for the four different training sets. The prototype tracklet is highlighted in red ▬. The resulting prototype reflects the minimum variation in the dynamical behavior. Tracklets are visualized in an unit-less embedded space.

The synthetically generated trajectories reflect a CV, a CA, and a CT dynamical behavior. Four training sets of 100 noise-free tracklets of 8 steps with a frame rate of 1 fps are generated. All starting positions and heading directions are uniformly sampled from an origin interval of $[-10\ m, 10\ m]$ and from a rotation interval of $[-90.0°, 90.0°]$. For the first set, the agent speed is uniformly sampled from $[5\ m/s, 25\ m/s]$ and is kept fixed. For the second set, an additional acceleration is sampled from the interval of $[5\ m/s^2, 25\ m/s^2]$. Thus, the second set includes only agents according to a CA model. The third and the fourth training set include in addition to the walking speed a turning component uniformly sampled from $[5°, 30°]$ and $[-15°, 15°]$. Thus, the agents of the last sets perform a curvilinear motion where the one set is biased towards one turning direction. The resulting prototype ▬ and the aligned tracklets of the four training sets are visualized in figure 4.25.

The images depict how the learned prototype reflects the underlying dynamics. For the top left image, the prototype is shaped like a straight line with equidistant points according to a CV model. The top right image shows a prototype that reflects the additional acceleration. The images on the bottom correspond to curvilinear motion. For the left image, the bias towards one rotation direction can be seen. For the right image, the prototype is adjusted to reflect the minimum variance of the input tracklets. For all training sets, the variation in translation and rotation is removed. Theoretically, the alignment results are independent of data pre-processing such as position normalization. Nevertheless, data pre-processing also helps to make training more stable. The important difference is that the reference for all tracklets is not arbitrarily chosen but learned. This enables to assess the input tracklet due to their distance to a common reference - the prototype of the bottleneck network.

4.2.3.4 Scenario: Real-World Data

As part of the TrajNet dataset collection, the *BIWI* [Pel09] dataset and the *UCY* [Ler07] dataset are described in section 4.2.1. Although the datasets include pedestrians with varying motion types, most pedestrians walk straight corresponding to a CV model (see section 4.2.1.1).

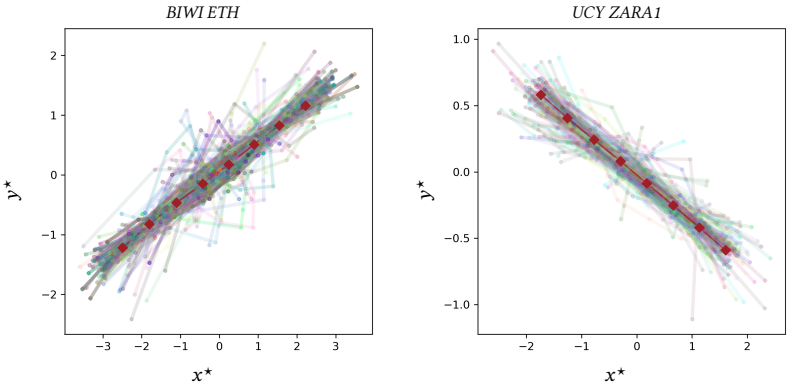

Figure 4.26: Alignment results for the *BIWI-ETH* and *UCY-ZARA1* sequences. The prototype tracklet is highlighted in red ▬. Tracklets are visualized in an unit-less embedded space.

Here, the datasets are used to analyze if the alignment network is able to learn a reasonable prototype from real-world data and to assess the input tracklets. In figure 4.26, the alignment results for input tracklets of length 8 of the *BIWI ETH* and the *UCY ZARA1* dataset are visualized. The resulting prototype ▬ reflects a nearly constant walking behavior. For both datasets, the variation due to affine transformations is compensated by the alignment network.

Visualization of the aligned tracklets for different steps during training for the *BIWI-ETH* dataset are depicted in figure 4.27. The images show how the parameters of the forward network and bottleneck network are jointly adapted

to remove the translation and rotation variation. The effect how the prototype is adjusted over time is also clearly visible.

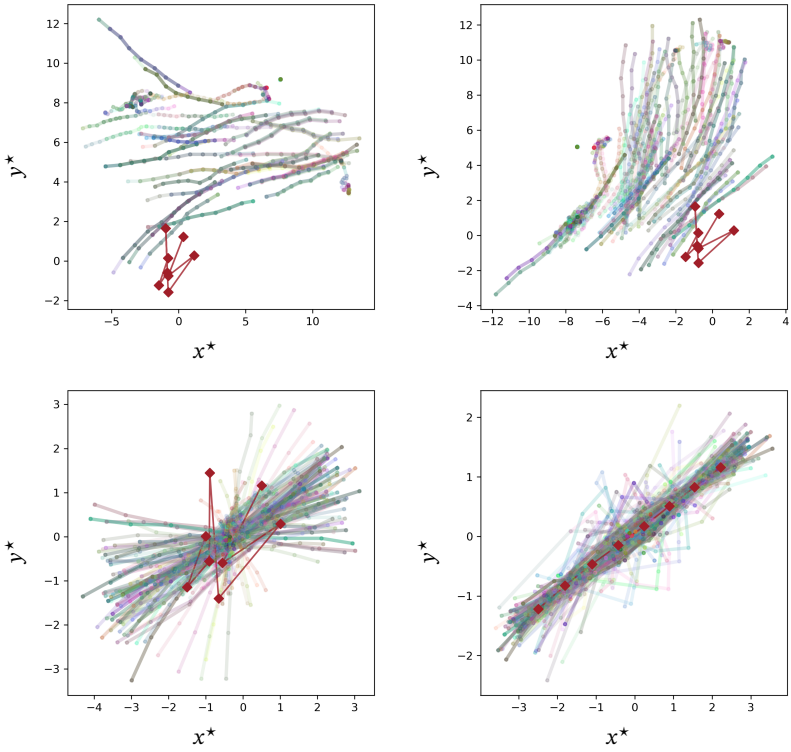

Figure 4.27: Visualization of aligned tracklets from the the *BIWI ETH* dataset for different time steps during training. The prototype tracklet is highlighted in red ▬. The images depict the joint learning of the transformations of the tracklets and adjusting the prototype. Tracklets are visualized in an unit-less embedded space.

The distance of the input tracklets to the learned prototype for the *BIWI ETH* and *UCY ZARA1* sequences are visualized in figure 4.28. For visualization,

the input tracklets are color-coded in accordance with a sequential colormap using the L2 distance to the prototype. The strongest outliers for both datsets correspond to standing or loitering persons. This can be better exposed by using the translation and rotation normalized input tracklets, as shown in figure 4.29. Since the result for the normalized input data is similar to using un-processed tracklets, this also demonstrates the robustness of the alignment network to remove affine transformation variation from the input tracklets. In figure 4.29, the larger distance to the prototype of person tracklets close to the origin is visible. The largest distances correspond to persons walking slowly with some sort of loitering behavior. This complies with the statements given in section 4.2.1.1.

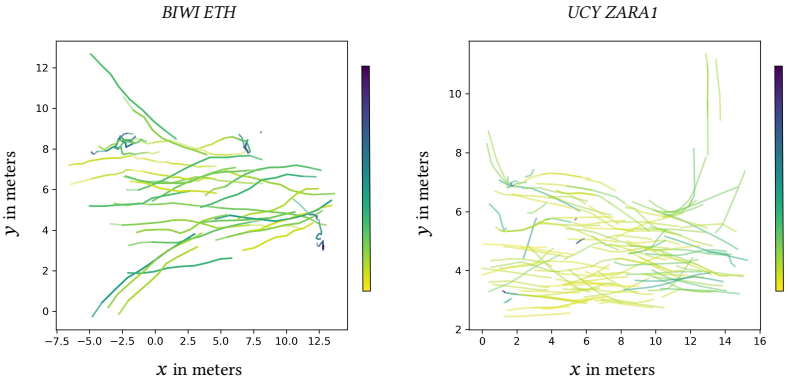

Figure 4.28: Color-coded input tracklets from the *BIWI ETH* and *UCY ZARA1* datasets. The color-coding is done based on the L2 distance (0 ■ → ■ max) between a tracklet and the learned prototype.

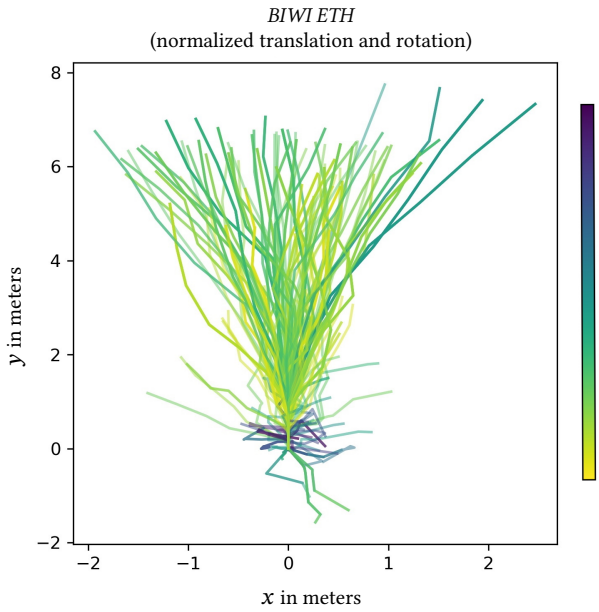

Figure 4.29: Visualization based on the L2 distance (0 → max) between normalized tracklets (translation and rotation) from the *BIWI ETH* dataset and the corresponding learned prototype.

4.2.3.5 Summary

The presented results show that the proposed alignment network offers new possibilities to assess input tracklet data. Due to the fact that trajectory clustering approaches mainly rely on time-varying positions, it is clear that the required information is mostly removed by applying normalization. The prototype provides a reference without shifting the variation along the tracklets. Thereby, the conditions to apply clustering approaches or out-of-distribution detection are improved. Moreover, the prototype is adjusted to match with the minimum variance in the input tracklets. Thus, it reflects the prototypical

dynamical behavior. The proposed alignment network offers a promising direction for future research to better separate the temporal-dependent *motion* cues from the spatial-dependent *environmental* cues.

5 Summary and Concluding Remarks

This thesis addressed the problem of state estimation of maneuvering objects as part of a visual detection-by-tracking system with a focus on applications in the surveillance and intelligent vehicle domains. Object observations provided by an appearance model, describing the object in image space, serve as input for recursive Bayesian filters or respectively for proposed RNN-based alternatives.

After discussing the interconnected Bayesian and deep learning functional viewpoints on state estimation, the IMM filter, as the most common representative based on a Bayesian formulation for dealing with model mismatches or maneuvering objects, was selected as our reference approach. For a model mismatch scenario of directly tracking in image space, this thesis contributes to an improved design of a basic IMM filter as a top-down filtering approach by introducing a state de-coupling and a re-coupling scheme.

The benefit of the suggested de-coupling scheme of an IMM filter was demonstrated for prototypical visual object tracking sequences, where the estimation of the mapping function to a 3D physical reference system is so far an widely unsolved problem. For better dealing with the corresponding observation uncertainties in these conditions, a state re-coupling scheme was introduced. Thereby, an implicit depth prior, which is connected through the object scales, enables a scale-dependent adaptation of the observation noise levels.

In order to reduce the amount of required engineering and to learn an improved process model set structure, the IMM functionality was transferred into a comparable deep learning architecture. Since the IMM filter is in particular designed to deal with the maneuver types of switching noise levels and

switching dynamics, the proposed RNN-based networks were correspondingly analyzed with regards to both maneuver types. This was done for the exemplary tasks of *path prediction* and *intention prediction*. Due to the fact that there exists a public standard benchmark, *path prediction* was mainly used for comparison to related approaches for motion prediction methods. The dataset analysis revealed that the trajectory data reflects the desired properties of observations provided by an underlying visual tracking component with varying noise levels. Despite the simple core architecture of an RNN-encoder with a dense layer for mapping back in the observation space, the proposed RNN network yielded the top-rank on *World H-H TrajNet* challenge, and thus achieved a performance comparable to related current state-of-the-art methods. The presented modification, such as *overshooting*, helped to enhance the prediction performance in the presence of varying noise levels.

The ability of proposed solutions with respect to the switching dynamics of objects was evaluated for *intention prediction* in the application domain of intelligent vehicles. In extensive experiments on synthetic and real-world datasets, the proposed RNN-based IMM filter surrogate (RNN-IMM) obtained a performance boost over existing proposed IMM filter configuration tailored to specific maneuver scenarios. Similar to an IMM filter solution, the presented RNN-IMM assigns a probability value to a dynamics and, based on them, puts out a multi-modal distribution over future object states. The RNN-IMM achieved a better performance for jointly estimating intentions and future paths. Also for a pure intention classification task, the (RNN-IMM) yields a performance boost compared to IMM filters. The amount of filter tuning is reduced due to a direct estimate of the dynamics probability, and thus there is no explicit modeling of the transition probabilities. Although in the analyzed maneuver scenarios several tailored IMM solutions exist, the RNN-IMM captures the switch in dynamics more reliable.

Instead of utilizing the state estimator as a top-down module in a visual tracking system, one direction for future research is the end-to-end reasoning on object motion directly from image sequences. Here, relying on an intermediate object state representation was a design choice to allow, among other

things, a fairer comparison between Bayesian filters and RNN-based alternatives. However, more and more end-to-end formulations of different tracking tasks are introduced. Besides currently achieving lower performance on benchmarks [Lea17], end-to-end solutions are a future direction to overcome requirements on system identification in the form of dynamics or observation models, and on any feature engineering for the appearance model.

As discussed in section 2.3, developing sophisticated methods for motion prediction which go beyond Kalman filtering is a clear trend. Due to the rapidly expanding field and thus the amount of new diverse methods, there is a need for improved standardized prediction benchmarking. Especially for benchmarking prediction with *contextual* cues, a well-defined categorization of the underlying data into specific conditions is crucial. The presented results revealed that it is required to first ensure a meaningful learned representation for single object dynamics rather than just to increase the model complexity by simply adding cues. Nevertheless, more *contextual* cues are undeniably necessary in order to improve object behavior anticipation. More methods, in particular deep learning-based methods, start to include the global structure of the environment and allow better estimates of context-dependent patterns in real-world data. Thus, intelligent autonomous systems require an in-depth semantic scene understanding to predict object motion or to plan and navigate alongside them [Rud19]. Contextual understanding with respect to features of the static environment and dynamic environment offers many options for future research to explore. In order to better separate spatial-independent *motion* cues from the spatial-dependent *environmental* cues, the proposed alignment network offers a promising direction for future research.

Bibliography

[Aba15] Abadi, M. et al.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/ (cit. on pp. 101, 123, 147).

[Aga12] Agamennoni, G.; Nieto, J.I. and Nebot, E.M.: "Estimation of Multivehicle Dynamics by Considering Contextual Information". In: *IEEE Transactions on Robotics* 28.4 (2012), pp. 855–870. DOI: 10.1109/TRO.2012.2195829 (cit. on p. 20).

[Ala16] Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Fei-Fei, L. and Savarese, S.: "Social LSTM: Human Trajectory Prediction in Crowded Spaces". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 961–971 (cit. on pp. 21, 29, 89, 94, 99, 103, 109, 110).

[Ale17] Alexandre, A.; Vignesh, R.; Kratarth, G.; A., Robicquet; A., Sadeghian; L., Fei-Fei and S., Savarese: "Learning to Predict Human Behavior in Crowded Scenes". In: *Group and Crowd Behavior for Computer Vision* (2017), pp. 183–207 (cit. on pp. 21, 29, 94, 103).

[And08] Andriluka, M.; Roth, S. and Schiele, B.: "People-tracking-by-detection and people-detection-by-tracking". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587583 (cit. on p. 7).

[Ave91] Averbuch, A.; Itzikowitz, S. and Kapon, T.: "Radar target tracking-Viterbi versus IMM". In: *IEEE Transactions on Aerospace and Electronic Systems* 27.3 (1991), pp. 550–563. DOI: 10.1109/7.81437 (cit. on p. 43).

[Bai18] Bai, S.; Kolter, J.Z. and Koltun, V.: "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". In: *arXiv preprint* abs/1803.01271 (2018). URL: http://arxiv.org/abs/1803.01271 (cit. on pp. 100, 101).

[Bar02] Bar-Shalom, Y.; Kirubarajan, T. and Li, X.-R.: Estimation with Applications to Tracking and Navigation. New York, NY, USA: John Wiley & Sons, Inc., 2002 (cit. on pp. 20, 26, 38, 39, 42, 43, 45, 48, 54, 55, 119, 124–126, 130).

[Bar08] Barth, A. and Franke, U.: "Where will the oncoming vehicle be the next second?" In: *Intelligent Vehicles Symposium (IV)*. 2008, pp. 1068–1073. DOI: 10.1109/IVS.2008.4621210 (cit. on p. 26).

[Bar12] Barber, D.: Bayesian Reasoning and Machine Learning. Cambridge University Press, 2012 (cit. on p. 21).

[Bat09] Batz, T.; Watson, K. and Beyerer, J.: "Recognition of dangerous situations within a cooperative group of vehicles". In: *Intelligent Vehicles Symposium (IV)*. June 2009, pp. 907–912. DOI: 10.1109/IVS.2009.5164400 (cit. on pp. 26, 134).

[Bec15] Becker, S.; Münch, D.; Kieritz, H.; Hübner, W. and Arens, M.: "Detecting Abandoned Objects using Interacting Multiple Models". In: *Proc. SPIE Volume 9652 Optics and Photonics for Counterterrorism, Crime Fighting, and Defence*. 2015 (cit. on p. 129).

[Bec16] Becker, S.; Kieritz, H.; Hübner, W. and Arens, M.: "On the benefit of state separation for tracking in image space with an Interacting Multiple Model Filter". In: *International Conference on Image and Signal Processing (ICISP)*. Springer International Publishing, 2016, pp. 3–11. DOI: 10.1007/978-3-319-33618-3_1 (cit. on pp. 3, 31).

[Bec18a] Becker, S.; Hübner, W. and Arens, M.: "State estimation for tracking in image space with a de- and re-coupled IMM filter". In: *Multimedia Tools and Applications* 77.15 (2018), pp. 20207–20226. DOI: 10.1007/s11042-017-5324-3 (cit. on pp. 3, 31).

[Bec18b] Becker, S.; Hug, R.; Hübner, W. and Arens, M.: "An Evaluation of Trajectory Prediction Approaches and Notes on the TrajNet Benchmark". In: *arXiv preprint arXiv:1805.07663* (2018) (cit. on pp. 73, 103).

[Bec18c] Becker, S.; Hug, R.; Hübner, W. and Arens, M.: "RED: A simple but effective Baseline Predictor for the TrajNet Benchmark". In: *European Conference on Computer Vision (ECCV) Workshops*. Springer International Publishing, 2018 (cit. on pp. 4, 30, 73, 103).

[Bec19a] Becker, S.: "RNN-based Prediction of Pedestrian Turning Maneuvers". In: *Proceedings of the Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory*. KIT Scientific Publishing, 2019 (cit. on pp. 4, 30, 73).

[Bec19b] Becker, S.; Hug, R.; Hübner, W. and Arens, M.: "An RNN-based IMM Filter Surrogate". In: *Scandinavian Conference on Image Analysis (SCIA)*. Vol. 11482. Springer International Publishing, 2019, pp. 387–398 (cit. on pp. 4, 30, 73).

[Ben13] Bengio, Y.; Boulanger-Lewandowski, N. and Pascanu, R.: "Advances in optimizing recurrent networks". In: *International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 8624–8628. doi: 10.1109/ICASSP.2013.6639349 (cit. on p. 83).

[Ben93] Bengio, Y.; Frasconi, P. and Simard, P.: "The problem of learning long-term dependencies in recurrent networks". In: *IEEE International Conference on Neural Networks*. 1993, 1183–1188 vol.3. doi: 10.1109/ICNN.1993.298725 (cit. on p. 77).

[Ber04] Bertozzi, M.; Broggi, A.; Fascioli, A.; Tibaldi, A.; Chapuis, R. and Chausse, F.: "Pedestrian localization and tracking system with Kalman filtering". In: *Intelligent Vehicles Symposium (IV)*. 2004, pp. 584–589. doi: 10.1109/IVS.2004.1336449 (cit. on pp. 26, 122).

[Bin05] Binelli, E.; Broggi, A.; Fascioli, A.; Ghidoni, S.; Grisleri, P.; Graf, T. and Meinecke, M.: "A modular tracking system for far

infrared pedestrian recognition". In: *Intelligent Vehicles Symposium*. 2005, pp. 759–764. DOI: 10.1109/IVS.2005.1505196 (cit. on pp. 26, 52, 122).

[Bis06] Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Berlin, Heidelberg: Springer-Verlag, 2006 (cit. on pp. 21, 86, 123).

[Bis94] Bishop, C.M.: Mixture Density Networks. Tech. rep. 1994. URL: https://www.microsoft.com/en-us/research/publication/mixture-density-networks/ (cit. on pp. 86, 87).

[Bis95] Bishop, C.M.: Neural Networks for Pattern Recognition. New York, NY, USA: Oxford University Press, Inc., 1995 (cit. on p. 88).

[Bla03] Black, J.; Ellis, T. and Rosin, P.: "A Novel Method for Video Tracking Performance Evaluation". In: *Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*. 2003, pp. 125–132 (cit. on p. 7).

[Bla99] Blackman, S.S. and Popoli, R.: Design and analysis of modern tracking systems. Artech House radar library. Boston, London: Artech House, 1999. URL: http://opac.inria.fr/record=b1097185 (cit. on p. 124).

[Blo88] Blom, H.A.P. and Bar-Shalom, Y.: "The interacting multiple model algorithm for systems with Markovian switching coefficients". In: *Transactions on Automatic Control* 33.8 (1988), pp. 780–783. DOI: 10.1109/9.1299 (cit. on p. 3).

[Blu15] Blundell, C.; Cornebise, J.; Kavukcuoglu, K. and Wierstra, D.: "Weight Uncertainty in Neural Network". In: *International Conference on Machine Learning (ICML)*. Ed. by Bach, Francis and Blei, David. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 1613–1622 (cit. on p. 89).

[Bob96] Bobick, A. and Davis, J.: "An appearance-based representation of action". In: *International Conference on Pattern Recognition (ICPR)*. Vol. 1. 1996, 307–312 vol.1. DOI: 10.1109/ICPR.1996.546039 (cit. on p. 28).

[Bon14] Bonnin, S.; Weisswange, T.H.; Kummert, F. and Schmuederich, J.: "Pedestrian crossing prediction using multiple context-based models". In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2014, pp. 378–385. doi: 10.1109/ITSC.2014.6957720 (cit. on pp. 28, 40).

[Bro16] Brouwer, N.; Kloeden, H. and Stiller, C.: "Comparison and evaluation of pedestrian motion models for vehicle safety systems". In: *International Conference on Intelligent Transportation Systems (ITSC)*. Nov. 2016, pp. 2207–2212. doi: 10.1109/ITSC.2016.7795912 (cit. on p. 17).

[Bro17] Brownlee, J.: Introduction to Time Series Forecasting with Python: How to Prepare Data and Develop Models to Predict the Future. Jason Brownlee, 2017. url: https://books.google.de/books?id=bA5ItAEACAAJ (cit. on p. 95).

[Bro88] Broomhead, D.S. and Lowe, D.: "Multivariable Functional Interpolation and Adaptive Networks". In: *Complex Systems* 2 (1988), pp. 321–355 (cit. on p. 75).

[Ceh14] Cehovin, L.; Kristan, M. and Leonardis, A.: "Is my new tracker really better than yours?" In: *Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2014, pp. 540–547. doi: 10.1109/WACV.2014.6836055 (cit. on p. 57).

[Ceh16] Cehovin, L.; Leonardis, A. and Kristan, M.: "Visual Object Tracking Performance Measures Revisited". In: *IEEE Transactions on Image Processing* 25.3 (2016), pp. 1261–1274. doi: 10.1109/TIP.2016.2520370 (cit. on pp. 51, 52, 55, 56, 61).

[Cha78] Chang, C.B. and Athans, M.: "State Estimation for Discrete Systems with Switching Parameters". In: *IEEE Transactions on Aerospace and Electronic Systems* AES-14.3 (1978), pp. 418–425. doi: 10.1109/TAES.1978.308603 (cit. on p. 43).

[Cho14] Cho, K.; Merrienboer, B. van; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H. and Bengio, Y.: "Learning Phrase

Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1724–1734 (cit. on pp. 77, 101).

[Chu15] Chung, J.; Kastner, K.; Dinh, L.; Goel, K.; Courville, A. and Bengio, Y.: "A Recurrent Latent Variable Model for Sequential Data". In: *Advances in Neural Information Processing Systems (NIPS)*. 2015 (cit. on p. 93).

[Cle16] Clevert, D.-A.; Unterthiner, T. and Hochreiter, S.: "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *International Conference on Learning Representations (ICLR)*. 2016. URL: http://arxiv.org/abs/1511.07289 (cit. on p. 87).

[Deo18] Deo, N. and Trivedi, M. M.: "Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs". In: *Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 1179–1184. DOI: 10.1109/IVS.2018.8500493 (cit. on pp. 30, 116).

[Dol12] Dollar, P.; Wojek, C.; Schiele, B. and Perona, P.: "Pedestrian Detection: An Evaluation of the State of the Art". In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 34.4 (2012). URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5975165 (cit. on pp. 52, 64, 69).

[Don15] Donahue, J.; Hendricks, L.A.; Guadarrama, S.; Rohrbach, M.; Venugopalan, S.; Saenko, K. and Darrell, T.: "Long-term Recurrent Convolutional Networks for Visual Recognition and Description". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2015 (cit. on p. 93).

[Doz16] Dozat, T.: "Incorporating Nesterov Momentum into Adam". In: *International Conference on Learning Representations Workshops (ICLRW)*. 2016 (cit. on p. 85).

[Dra66] Draper, N.R. and Smith, H.: Applied regression analysis. Wiley series in probability and mathematical statistics. New York [u.a.]: Wiley, 1966. IX, 407 (cit. on p. 97).

[Duc11] Duchi, J.; Hazan, E. and Singer, Y.: "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research (JMLR)* 12 (2011), pp. 2121–2159. URL: http://dl.acm.org/citation.cfm?id=1953048.2021068 (cit. on p. 84).

[Ell09] Ellis, D.; Sommerlade, E. and Reid, I.: "Modelling pedestrian trajectory patterns with Gaussian processes". In: *International Conference on Computer Vision Workshops (ICCVW)*. IEEE. 2009, pp. 1229–1234. DOI: 10.1109/ICCVW.2009.5457470 (cit. on p. 110).

[Elm90] Elman, J.L.: "Finding Structure in Time." In: *Cognitive Science* 14.2 (1990), pp. 179–211. URL: http://dblp.uni-trier.de/db/journals/cogsci/cogsci14.html#Elman90 (cit. on pp. 13, 76).

[Eln01] Elnagar, A.: "Prediction of moving objects in dynamic environments using Kalman filters". In: *International Symposium on Computational Intelligence in Robotics and Automation*. IEEE. 2001, pp. 414–419. DOI: 10.1109/CIRA.2001.1013236 (cit. on pp. 20, 26, 52, 122).

[Enz09] Enzweiler, M. and Gavrila, D.M.: "Monocular Pedestrian Detection: Survey and Experiments". In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 31.12 (2009), pp. 2179–2195 (cit. on pp. 51, 52, 63, 64, 66–68).

[Fer09] Ferryman, J. and Shahrokni, A.: "PETS2009: Dataset and challenge". In: *International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*. 2009, pp. 1–6. DOI: 10.1109/PETS-WINTER.2009.5399556 (cit. on pp. 91, 93).

[For17] Fortunato, M.; Blundell, C. and Vinyals, O.: "Bayesian Recurrent Neural Networks". In: *arXiv preprint arxiv:1704.02798* (2017) (cit. on p. 88).

[Fre97] Freund, Yoav and Schapire, Robert E: "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139 (cit. on p. 64).

[Gal16] Gal, Y. and Ghahramani, Z.: "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *International Conference on Machine Learning (ICML)*. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1050–1059. URL: http://proceedings.mlr.press/v48/gal16.html (cit. on p. 89).

[Gav99] Gavrila, D.M.: "The Visual Analysis of Human Movement: A Survey". In: *Computer Vision and Image Understanding* 73.1 (1999), pp. 82–98. DOI: https://doi.org/10.1006/cviu.1998.0716 (cit. on pp. 15, 25).

[Glo11] Glorot, X.; Bordes, A. and Bengio, Y.: "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, 2011, pp. 315–323 (cit. on p. 87).

[Gol14] Goldhammer, M.; Doll, K.; Brunsmann, U.; Gensler, A. and Sick, B.: "Pedestrian's Trajectory Forecast in Public Traffic with Artificial Neural Networks". In: *2014 22nd International Conference on Pattern Recognition*. 2014, pp. 4110–4115. DOI: 10.1109/ICPR.2014.704 (cit. on pp. 21, 29, 52, 125).

[Gol15] Goldhammer, M.; Köhler, S.; Doll, K. and Sick, B.: "Camera based pedestrian path prediction by means of polynomial least-squares approximation and multilayer perceptron neural networks". In: *SAI Intelligent Systems Conference (IntelliSys)*. 2015, pp. 390–399. DOI: 10.1109/IntelliSys.2015.7361171 (cit. on p. 29).

[Gol17] Goldhammer, M.: "Selbstlernende Algorithmen zur videobasierten Absichtserkennung von Fußgängern". PhD thesis. University of Kassel, 2017 (cit. on p. 128).

[Goo16] Goodfellow, I.J.; Bengio, Y. and Courville, A.: Deep Learning. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016 (cit. on pp. 11–13, 30, 74, 77, 80, 82, 84).

[Gra13a] Graves, A.: "Generating Sequences With Recurrent Neural Networks". In: *arXiv preprint arXiv:1308.0850* (2013) (cit. on pp. 13, 86–89).

[Gra13b] Graves, A.; Mohamed, A. and Hinton, G.: "Speech recognition with deep recurrent neural networks". In: *International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 6645–6649. doi: 10.1109/ICASSP.2013.6638947 (cit. on p. 80).

[Gra13c] Graves, A.; Mohamed, A.R and Hinton, G.: "Speech recognition with deep recurrent neural networks". In: *International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 6645–6649. doi: 10.1109/ICASSP.2013.6638947 (cit. on p. 93).

[Gre17] Greff, K.; Srivastava, R. K.; Koutník, J.; Steunebrink, B. R. and Schmidhuber, J.: "LSTM: A Search Space Odyssey". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.10 (2017), pp. 2222–2232. doi: 10.1109/TNNLS.2016.2582924 (cit. on pp. 77, 80).

[Gri18] Grinberg, M.: "Feature-Based Probabilistic Data Association for Video-Based Multi-Object Tracking". PhD thesis. Karlsruher Institut für Technologie (KIT), 2018. 256 pp. doi: 10.5445/KSP/1000081430 (cit. on pp. 26, 32).

[Gup18] Gupta, A.; Johnson, J.; Fei-Fei, L.; Savarese, S. and Alahi, A.: "Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2018 (cit. on pp. 94, 99, 110, 111).

[Ha18] Ha, D. and Eck, D.: "A Neural Representation of Sketch Drawings". In: *International Conference on Learning Representations (ICLR)*. 2018. url: https://openreview.net/forum?id=Hy6GHpkCW (cit. on p. 89).

[Ham11] Hampel, F.R.; Ronchetti, E.M.; Rousseeuw, P.J. and Stahel, W.A.: Robust statistics: the approach based on influence functions. Vol. 196. John Wiley & Sons, 2011 (cit. on p. 146).

[Has15a] Hashimoto, Y.; Gu, Y.; Hsu, L. and Kamijo, S.: "Probability estimation for pedestrian crossing intention at signalized crosswalks". In: *International Conference on Vehicular Electronics and Safety (ICVES)*. 2015, pp. 114–119. doi: 10.1109/ICVES.2015.7396904 (cit. on pp. 28, 40).

[Has15b] Hashimoto, Y.; Yanlei, G.; Hsu, L. and Shunsuke, K.: "A Probabilistic Model for the Estimation of Pedestrian Crossing Behavior at Signalized Intersections". In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2015, pp. 1520–1526. doi: 10.1109/ITSC.2015.248 (cit. on pp. 28, 40, 116).

[Has18] Hasan, I.; Setti, F.; Tsesmelis, T.; Bue, A. Del; Galasso, F. and Cristani, M.: "MX-LSTM: mixing tracklets and vislets to jointly forecast trajectories and head poses". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2018 (cit. on pp. 94, 95, 99, 101, 103, 110).

[He16] He, K.; Zhang, X.; Ren, S. and Sun, J.: "Deep Residual Learning for Image Recognition". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90 (cit. on p. 94).

[Hel95] Helbing, D. and Molnár, P.: "Social force model for pedestrian dynamics". In: *Phys. Rev. E* 51 (5 1995), pp. 4282–4286. doi: 10.1103/PhysRevE.51.4282 (cit. on p. 109).

[Her15] Hernandez-Lobato, J.M. and Adams, R.: "Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks". In: *International Conference on Machine Learning (ICML)*. Ed. by Bach, Francis and Blei, David. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 1861–1869 (cit. on p. 89).

[Hir08] Hirschmüller, H.: "Stereo Processing by Semi-Global Matching and Mutual Information". In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 30.2 (2008), pp. 328–341. doi: 10.1109/TPAMI.2007.1166 (cit. on pp. 123, 142).

[Hir18] Hirakawa, T.; Yamashita, T.; Tamaki, T. and Fujiyoshi, H.: "Survey on Vision-Based Path Prediction". In: *Distributed, Ambient and Pervasive Interactions: Technologies and Contexts*. Cham: Springer International Publishing, 2018, pp. 48–64 (cit. on pp. 17, 29).

[Hoc97] Hochreiter, S. and Schmidhuber, J.: "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. doi: 10.1162/neco.1997.9.8.1735 (cit. on pp. 77, 101, 123).

[Hof04] Hofbaur, M.W. and Williams, B.C.: "Hybrid estimation of complex systems". In: *Transactions on Systems, Man, and Cybernetics* 34.5 (2004), pp. 2178–2191. doi: 10.1109/TSMCB.2004.835009 (cit. on p. 27).

[Hub15] Huber, M.: "Nonlinear Gaussian Filtering : Theory, Algorithms, and Applications". Habilitation. 2015. 270 pp. doi: 10.5445/KSP/1000045491 (cit. on pp. 14, 31, 32, 37).

[Hub64] Huber, P.J.: "Robust estimation of a location parameter". In: *Annals of Mathematical Statistics* 35.1 (1964), pp. 73–101. url: http://dx.doi.org/10.1214/aoms/1177703732 (cit. on p. 146).

[Hug17] Hug, R.; Becker, S.; Hübner, W. and Arens, M.: "On the reliability of LSTM-MDL models for pedestrian trajectory prediction". In: *Representations, Analysis and Recognition of Shape and Motion from Imaging Data (RFMI)*. Savoie, France, 2017 (cit. on pp. 94, 104, 117).

[Hug18] Hug, R.; Becker, S.; Hübner, W. and Arens, M.: "Particle-based Pedestrian Path Prediction using LSTM-MDL Models". In: *21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2684–2691. doi: 10.1109/ITSC.2018.8569478 (cit. on p. 94).

[Hug19] Hug, R.; Hübner, W. and Arens, M.: "Modeling continuous-time stochastic processes using $\mathcal{N}$-Curve mixtures". In: *arXiv preprint arXiv:1908.04030* (2019) (cit. on p. 89).

[Jac08] Jaccard, P.: "Nouvelles recherches sur la distribution florale". In: *Bulletin de la Societe Vaudoise de Sciences Naturelles* 44 (1908), pp. 223–270 (cit. on p. 59).

[Jac88] Jacobs, R.A.: "Increased rates of convergence through learning rate adaptation". In: *Neural Networks* 1.4 (1988), pp. 295–307. doi: https://doi.org/10.1016/0893-6080(88)90003-2 (cit. on p. 84).

[Kae04] Kaempchen, N.; Weiss, K.; Schaefer, M. and Dietmayer, K. C. J.: "IMM object tracking for high dynamic driving maneuvers". In: *Intelligent Vehicles Symposium (IV)*. 2004, pp. 825–830. doi: 10.1109/IVS.2004.1336491 (cit. on p. 28).

[Kal60] Kalman, R.E.: "A new approach to linear filtering and prediction problems". In: *Journal of basic Engineering* 82.1 (1960), pp. 35–45 (cit. on pp. 15, 26, 32).

[Kar11] Karaman, S. and Frazzoli, E.: "Sampling-based Algorithms for Optimal Motion Planning". In: *International Journal of Robotics Research* 30.7 (2011), pp. 846–894. doi: 10.1177/0278364911406761 (cit. on p. 22).

[Kar16a] Karasev, V.; Ayvaci, A.; Heisele, B. and Soatto, S.: "Intent-aware long-term prediction of pedestrian motion". In: *International Conference on Robotics and Automation (ICRA)*. 2016, pp. 2543–2549. doi: 10.1109/ICRA.2016.7487409 (cit. on pp. 22, 27).

[Kar16b] Karpathy, A.: "Connecting Images and Natural Language". PhD thesis. Stanford University, 2016 (cit. on p. 81).

[Kel11] Keller, C.G.; Hermes, C. and Gavrila, D.M.: "Will the Pedestrian Cross? Probabilistic Path Prediction Based on Learned Motion Features". In: *Pattern Recognition*. Vol. 6835. Lecture Notes in Computer Science. Springer, 2011, pp. 386–395. doi: 10.1007/978-3-642-23123-0_39 (cit. on pp. 28, 29, 125, 126).

[Kel14] Keller, C.G. and Gavrila, D.M.: "Will the Pedestrian Cross? A Study on Pedestrian Path Prediction". In: vol. 15. 2. 2014, pp. 494–506. doi: 10.1109/TITS.2013.2280766 (cit. on pp. 21, 128).

[Kie16] Kieritz, H.; Becker, S.; Hübner, W. and Arens, M.: "Online Multi-Person Tracking using Integral Channel Features". In: *Conference on Advanced Video and Signal-based Surveillance (AVSS)*. 2016 (cit. on pp. 63, 64).

[Kin15] Kingma, D.P. and Ba, J.: "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations (ICLR)*. 2015 (cit. on pp. 84, 101, 123, 147).

[Kit12] Kitani, K.M.; Ziebart, B.D.; Bagnell, J.A. and Hebert, M.: "Activity Forecasting". In: *European Conference on Computer Vision (ECCV)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 201–214 (cit. on pp. 22, 27).

[Kol09] Koller, D. and Friedman, N.: Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning. The MIT Press, 2009 (cit. on pp. 10, 28).

[Koo14] Kooij, J.F.P.; Schneider, N.; Flohr, F. and Gavrila, D.M.: "Context-Based Pedestrian Path Prediction". In: *European Conference on Computer Vision (ECCV)*. Cham: Springer International Publishing, 2014, pp. 618–633 (cit. on pp. 28, 121, 122, 125, 134, 135, 137).

[Koo16] Kooij, J.F.P.; Englebienne, G. and Gavrila, D.M.: "Mixture of Switching Linear Dynamics to Discover Behavior Patterns in Object Tracks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.2 (2016), pp. 322–334. doi: 10.1109/TPAMI.2015.2443801 (cit. on p. 20).

[Koo19] Kooij, J.F.P.; Flohr, F.; Pool, E.A.I. and Gavrila, D.M.: "Context-Based Path Prediction for Targets with Switching Dynamics". In: *International Journal of Computer Vision* 127.3 (2019), pp. 239–262. doi: 10.1007/s11263-018-1104-4 (cit. on pp. 20, 28, 40, 122, 128, 134).

[Kre17] Krebs, S.; Duraisamy, B. and Flohr, F.: "A survey on leveraging deep neural networks for object tracking". In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2017, pp. 411–418. doi: 10.1109/ITSC.2017.8317904 (cit. on p. 9).

[Kri14] Kristan, M. et al.: "The Visual Object Tracking VOT2014 challenge results". In: *Proceedings, European Conference on Computer Vision (ECCV) Visual Object Tracking Challenge Workshop*. Vol. 8926. Lecture Notes in Computer Science. Zurich, Switzerland: Springer International Publishing, 2014, pp. 191–217. doi: 10.1007/978-3-319-16181-5_14 (cit. on pp. 51, 55, 56, 61).

[Kru13] Kruse, T.; Pandey, A.K.; Alami, R. and Kirsch, A.: "Human-aware robot navigation: A survey". In: *Robotics and Autonomous Systems* 61.12 (2013), pp. 1726–1743. doi: https://doi.org/10.1016/j.robot.2013.05.007. url: http://www.sciencedirect.com/science/article/pii/S0921889013001048 (cit. on p. 17).

[Kuc17] Kucner, T.P.; Magnusson, M.; Schaffernicht, E.; Bennetts, V.H. and Lilienthal, A.J.: "Enabling Flow Awareness for Mobile Robots in Partially Observable Environments". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 1093–1100. doi: 10.1109/LRA.2017.2660060 (cit. on p. 21).

[Kuh15] Kuhnt, F.; Kohlhaas, R.; Schamm, T. and Zöllner, J. M.: "Towards a unified traffic situation estimation model — Street-dependent behaviour and motion models". In: *International Conference on Information Fusion (Fusion)*. 2015, pp. 1223–1229 (cit. on p. 28).

[Lab14] Labbe, R.: Kalman and Bayesian Filters in Python. https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python. 2014 (cit. on pp. 39, 48).

[Laf01] Lafferty, J.D.; McCallum, A. and Pereira, F.C.N.: "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *International Conference on Machine Learning (ICML)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289. url: http://dl.acm.org/citation.cfm?id=645530.655813 (cit. on p. 28).

[Las17] Lasota, P.A.; Fong, T. and Shah, J.A.: "A Survey of Methods for Safe Human-Robot Interaction". In: *Foundations and Trends® in*

Robotics 5.4 (2017), pp. 261–349. DOI: 10.1561/2300000052. URL: http://dx.doi.org/10.1561/2300000052 (cit. on p. 17).

[Lea17] LEAL-TAIXÉ, L.; MILAN, A.; SCHINDLER, K.; CREMERS, D.; REID, I.D. and ROTH, S.: "Tracking the Trackers: An Analysis of the State of the Art in Multiple Object Tracking". In: *arXiv preprint arXiv:1704.02781* (2017) (cit. on p. 157).

[Lef14] LEFÈVRE, Stéphanie; VASQUEZ, Dizan and LAUGIER, Christian: "A survey on motion prediction and risk assessment for intelligent vehicles". In: *ROBOMECH Journal* 1.1 (July 2014), p. 1. DOI: 10.1186/s40648-014-0001-z. URL: https://doi.org/10.1186/s40648-014-0001-z (cit. on p. 17).

[Ler07] LERNER, A.; CHRYSANTHOU, Y. and LISCHINSKI, D.: "Crowds by Example". In: *Computer Graphic Forum* 26.3 (2007), pp. 655–664 (cit. on pp. 91, 93, 144, 147, 150).

[Li03] LI, X.R. and JILKOV, V.P.: "Survey of maneuvering target tracking. Part I. Dynamic models". In: *IEEE Transactions on Aerospace and Electronic Systems* 39.4 (2003), pp. 1333–1364. DOI: 10.1109/TAES.2003.1261132 (cit. on pp. 26, 134).

[Li05] LI, X.R. and JILKOV, V.P.: "Survey of maneuvering target tracking. Part V. Multiple-model methods". In: *IEEE Transactions on Aerospace and Electronic Systems* 41.4 (2005), pp. 1255–1321. DOI: 10.1109/TAES.2005.1561886 (cit. on p. 53).

[Li10] LI, X.R. and JILKOV, V.P.: "Survey of Maneuvering Target Tracking. Part II: Motion Models of Ballistic and Space Targets". In: *IEEE Transactions on Aerospace and Electronic Systems* 46.1 (2010), pp. 96–119. DOI: 10.1109/TAES.2010.5417150 (cit. on pp. 27, 39, 42).

[Lin16] LINDER, T.; BREUERS, S.; LEIBE, B. and ARRAS, K.O.: "On multi-modal people tracking from mobile platforms in very crowded and dynamic environments". In: *International Conference on Robotics and Automation (ICRA)*. 2016, pp. 5512–5519. DOI: 10.1109/ICRA.2016.7487766 (cit. on p. 28).

[Lub12] Luber, M.; Spinello, L.; Silva, J. and Arras, K. O.: "Socially-aware robot navigation: A learning approach". In: *International Conference on Intelligent Robots and Systems*. 2012, pp. 902–907. doi: 10.1109/IROS.2012.6385716 (cit. on p. 21).

[Mac92] MacKay, D.J.C.: "A Practical Bayesian Framework for Backpropagation Networks". In: *Neural Computation* 4.3 (1992), pp. 448–472. doi: 10.1162/neco.1992.4.3.448. url: http://dx.doi.org/10.1162/neco.1992.4.3.448 (cit. on p. 88).

[Mad13] Madrigal, F. and Hayet, J.-B.: "Evaluation of multiple motion models for multiple pedestrian visual tracking". In: *International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE. 2013, pp. 31–36. doi: 10.1109/AVSS.2013.6636612 (cit. on p. 28).

[Mag11] Maggio, E. and Cavallaro, A.: Video Tracking - Theory and Practice. Wiley, 2011, pp. I–XXV, 1–266 (cit. on p. 7).

[Mar17] Martinez, Julieta; Black, Michael J. and Romero, Javier: "On Human Motion Prediction Using Recurrent Neural Networks". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 4674–4683. doi: 10.1109/CVPR.2017.497. url: https://doi.org/10.1109/CVPR.2017.497 (cit. on pp. 93, 103).

[Maz98] Mazor, E.; Averbuch, A.; Bar-Shalom, Y. and Dayan, J.: "Interacting multiple model methods in target tracking: a survey". In: *Transactions on Aerospace and Electronic Systems* 34.1 (1998), pp. 103–123. doi: 10.1109/7.640267 (cit. on p. 28).

[Meu08] Meuter, M.; Iurgel, U.; Park, S. and Kummert, A.: "The unscented Kalman filter for pedestrian tracking from a moving host". In: *Intelligent Vehicles Symposium (IV)*. 2008, pp. 37–42. doi: 10.1109/IVS.2008.4621191 (cit. on pp. 26, 122).

[Mil16] Milan, A.; Leal-Taixé, L.; Reid, I.D.; Roth, S. and Schindler, K.: "MOT16: A Benchmark for Multi-Object Tracking". In: *arXiv preprint arXiv:1603.00831* (2016) (cit. on p. 64).

[Mil19] Miller, J. and Hardt, M.: "Stable Recurrent Models". In: *International Conference on Learning Representations (ICLR)*. 2019. URL: https://openreview.net/forum?id=Hygxb2CqKm (cit. on pp. 100, 103).

[Møg15] Møgelmose, A.; Trivedi, M.M. and Moeslund, T.B.: "Trajectory analysis and prediction for improved pedestrian safety: Integrated framework and evaluations". In: *Intelligent Vehicles Symposium (IV)*. 2015, pp. 330–335. DOI: 10.1109/IVS.2015.7225707 (cit. on pp. 26, 52, 122).

[Moh15] Mohamed, S.: "A Statistical View of Deep Learning". In: 2015 (cit. on pp. 9, 11).

[Mor08] Morris, B.T. and Trivedi, M.M.: "A Survey of Vision-Based Trajectory Learning and Analysis for Surveillance". In: *IEEE Transactions on Circuits and Systems for Video Technology* 18.8 (2008), pp. 1114–1127. DOI: 10.1109/TCSVT.2008.927109 (cit. on pp. 17, 21).

[Mün16a] Münch, D.; Hilsenbeck, B.; Kieritz, H.; Becker, S.; Grosselfinger, A.-K.; Hübner, W. and Arens, M.: "Detection of infrastructure manipulation with knowledge-based video surveillance". In: *Proc. SPIE Volume 9995 Optics and Photonics for Counterterrorism, Crime Fighting, and Defence*. 2016 (cit. on p. 129).

[Mün16b] Münch, D.; Kieritz, H.; Becker, S.; Hübner, W. and Arens, M.: "Video-based log generation for security systems in indoor surveillance scenarios". In: *Proc. of the 11th Future Security Research Conference*. 2016 (cit. on p. 129).

[Mur12] Murphy, K.P.: Machine Learning: A Probabilistic Perspective. The MIT Press, 2012 (cit. on pp. 22, 27, 40).

[Mur17] Murino, V.; Cristani, M.; Shah, S. and Savarese, S., eds.: Group and Crowd Behavior for Computer Vision, 1st Edition. Academic Press, 2017. URL: http://www.sciencedirect.com/science/book/9780128092767 (cit. on p. 17).

[Nes83] Nesterov, Y.E.: "A method for solving the convex programming problem with convergence rate O($1/k^2$)". In: *Doklady Akademii Nauk SSSR* 269 (1983), pp. 543–547. URL: https://ci.nii.ac.jp/naid/10029946121/en/ (cit. on p. 83).

[Nik18] Nikhil, N. and Morris, B.T.: "Convolutional Neural Network for Trajectory Prediction". In: *The European Conference on Computer Vision (ECCV) Workshops*. 2018 (cit. on p. 103).

[Pas13] Pascanu, Razvan; Mikolov, Tomas and Bengio, Yoshua: "On the Difficulty of Training Recurrent Neural Networks". In: *International Conference on Machine Learning (ICML)*. Atlanta, GA, USA: JMLR, 2013, pp. III-1310–III-1318. URL: http://dl.acm.org/citation.cfm?id=3042817.3043083 (cit. on p. 77).

[Pel09] Pellegrini, S.; Ess, A.; Schindler, K. and Gool, L. van: "You'll never walk alone: Modeling social behavior for multi-target tracking". In: *International Conference on Computer Vision (ICCV)*. IEEE. 2009, pp. 261–268. DOI: 10.1109/ICCV.2009.5459260 (cit. on pp. 20, 91, 93, 99, 127, 144, 147, 150).

[Pit05] Pitre, R.R.; Jilkov, V.P. and Li, X.R.: "A comparative study of multiple-model algorithms for maneuvering target tracking". In: vol. 5809. 2005. DOI: 10.1117/12.609681. URL: http://dx.doi.org/10.1117/12.609681 (cit. on pp. 40, 43).

[Pol64] Polyak, B.T.: "Some methods of speeding up the convergence of iteration methods". In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17. DOI: https://doi.org/10.1016/0041-5553(64)90137-5 (cit. on p. 82).

[Poo17] Pool, E.A.I.; Kooij, J.F.P. and Gavrila, D.M.: "Using road topology to improve cyclist path prediction". In: *Intelligent Vehicles Symposium (IV)*. 2017, pp. 289–296. DOI: 10.1109/IVS.2017.7995734 (cit. on pp. 20, 27).

[Pre07] Press, W.H.; Teukolsky, S.A.; Vetterling, W.T. and Flannery, B.P.: Numerical Recipes 3rd Edition: The Art of Scientific Computing. 3rd ed. New York, NY, USA: Cambridge University Press, 2007 (cit. on p. 88).

[Qui15] Quintero Mínguez, R.; Parra Alonso, I.; Fernández-Llorca, D. and Sotelo, M.Á.: "Pedestrian Intention and Pose Prediction through Dynamical Models and Behaviour Classification". In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2015, pp. 83–88. DOI: 10.1109/ITSC.2015.22 (cit. on pp. 28, 29).

[Qui19] Quintero Mínguez, R.; Parra Alonso, I.; Fernández-Llorca, D. and Sotelo, M.Á.: "Pedestrian Path, Pose, and Intention Prediction Through Gaussian Process Dynamical Models and Pedestrian Activity Recognition". In: *Transactions on Intelligent Transportation Systems* 20.5 (2019), pp. 1803–1814. DOI: 10.1109/TITS.2018.2836305 (cit. on p. 29).

[Ras19] Rasouli, A. and Tsotsos, J.K.: "Autonomous Vehicles That Interact With Pedestrians: A Survey of Theory and Practice". In: *IEEE Transactions on Intelligent Transportation Systems* (2019), pp. 1–19. DOI: 10.1109/TITS.2019.2901817 (cit. on pp. 17, 23, 26, 29).

[Red18] Reddi, S.J.; Kale, S. and Kumar, S.: "On the Convergence of Adam and Beyond". In: *International Conference on Learning Representations (ICLR)*. 2018. URL: https://openreview.net/forum?id=ryQu7f-RZ (cit. on p. 85).

[Reh18] Rehder, E.; Wirth, F.; Lauer, M. and Stiller, C.: "Pedestrian Prediction by Planning Using Deep Neural Networks". In: *International Conference on Robotics and Automation (ICRA)*. 2018, pp. 1–5. DOI: 10.1109/ICRA.2018.8460203 (cit. on p. 22).

[Rid18] Ridel, D.; Rehder, E.; Lauer, M.; Stiller, C. and Wolf, D.: "A Literature Review on the Prediction of Pedestrian Behavior in Urban Scenarios". In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 3105–3112. DOI: 10.1109/ITSC.2018.8569415 (cit. on pp. 17, 26, 29, 116).

[Rob16] Robicquet, A.; Sadeghian, A.; Alahi, A. and Savarese, S.: "Learning Social Etiquette: Human Trajectory Understanding In Crowded Scenes". In: *European Conference on Computer Vision (ECCV)*. Springer International Publishing, 2016,

pp. 549–565. DOI: 10.1007/978-3-319-46484-8_33. URL: http://dx.doi.org/10.1007/978-3-319-46484-8_33 (cit. on pp. 91, 93).

[Rös17] Rösmann, C.; Oeljeklaus, M.; Hoffmann, F. and Bertram, T.: "Online trajectory prediction and planning for social robot navigation". In: *International Conference on Advanced Intelligent Mechatronics (AIM)*. 2017, pp. 1255–1260. DOI: 10.1109/AIM.2017.8014190 (cit. on p. 22).

[Ros58] Rosenblatt, F.: "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain". In: *Psychological Review* (1958), pp. 65–386 (cit. on p. 74).

[Rud16] Ruder, S.: "An overview of gradient descent optimization algorithms". In: *arxiv preprint arxiv/1609.04747* (2016) (cit. on p. 85).

[Rud17] Rudenko, A.; Palmieri, L. and Arras, K.O.: "Predictive Planning for a Mobile Robot in Human Environments". In: *Proceedings of the Workshop on AI Planning and Robotics: Challenges and Methods*. 2017 (cit. on p. 22).

[Rud19] Rudenko, A.; Palmieri, L.; Herman, M.; Kitani, K.M.; Gavrila, D.M. and Arras, K.O.: "Human Motion Trajectory Prediction: A Survey". In: *arXiv preprint arXiv:1905.06113* (2019) (cit. on pp. 16, 17, 20, 21, 29, 157).

[Rum86] Rumelhart, D.E.; Hinton, G.E. and Williams, R.J.: "Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1". In: ed. by Rumelhart, David E.; McClelland, James L. and PDP Research Group, CORPORATE. Cambridge, MA, USA: MIT Press, 1986. Chap. Learning Internal Representations by Error Propagation, pp. 318–362. URL: http://dl.acm.org/citation.cfm?id=104279.104293 (cit. on p. 83).

[Rum88] Rumelhart, D.E.; Hinton, G.E. and Williams, R.J.: "Neurocomputing: Foundations of Research". In: ed. by Anderson, James A. and Rosenfeld, Edward. Cambridge, MA, USA: MIT Press, 1988. Chap. Learning Representations by Back-propagating Errors, pp. 696–699 (cit. on pp. 11, 76).

[Sad18] Sadeghian, A.; Kosaraju, V.; Gupta, A.; Savarese, S. and Alahi, A.: "TrajNet: Towards a Benchmark for Human Trajectory Prediction". In: *arXiv preprint* (2018) (cit. on pp. 91, 93, 110).

[Sad19] Sadeghian, A.; Kosaraju, V.; Sadeghian, A.; Hirose, N. and Savarese, S.: "SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints". In: (2019) (cit. on p. 111).

[Sal18] Saleh, K.; Hossny, M. and Nahavandi, S.: "Intent Prediction of Pedestrians via Motion Trajectories Using Stacked Recurrent Neural Networks". In: *IEEE Transactions on Intelligent Vehicles* 3.4 (2018), pp. 414–424. doi: 10.1109/TIV.2018.2873901 (cit. on p. 30).

[Sär13] Särkkä, S.: Bayesian Filtering and Smoothing. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2013. doi: 10.1017/CBO9781139344203 (cit. on pp. 38, 48).

[Sch00] Schuster, M.: "Better Generative Models for Sequential Data Problems: Bidirectional Recurrent Mixture Density Networks". In: *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2000, pp. 589–595. url: http://papers.nips.cc/paper/1778-better-generative-models-for-sequential-data-problems-bidirectional-recurrent-mixture-density-networks.pdf (cit. on p. 86).

[Sch13] Schneider, N. and Gavrila, D.M.: "Pedestrian Path Prediction with Recursive Bayesian Filters: A Comparative Study". In: *German Conference on Pattern Recognition (GCPR)*. Springer Berlin Heidelberg, 2013, pp. 174–183. url: https://doi.org/10.1007/978-3-642-40602-7%5C_18 (cit. on pp. 1, 20, 26, 28, 40, 52, 119, 121–126, 132).

[Sch14] Schaul, T.; Antonoglou, I. and Silver, D.: "Unit Tests for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)*. 2014. url: http://arxiv.org/abs/1312.6055 (cit. on p. 85).

[Sch15] Schulz, A.T. and Stiefelhagen, R.: "Pedestrian intention recognition using Latent-dynamic Conditional Random Fields". In: *Intelligent Vehicles Symposium (IV)*. 2015, pp. 622–627. doi: 10.1109/IVS.2015.7225754 (cit. on pp. 28, 40, 52, 122).

[Spi10] Spinello, D. and Stilwell, D. J.: "Nonlinear Estimation With State-Dependent Gaussian Observation Noise". In: *IEEE Transactions on Automatic Control* 55.6 (2010), pp. 1358–1366. doi: 10.1109/TAC.2010.2042006 (cit. on p. 65).

[Sze10] Szeliski, R.: Computer Vision: Algorithms and Applications. 1st. Berlin, Heidelberg: Springer-Verlag, 2010 (cit. on p. 18).

[Tam12] Tamura, Y.; Le, P.D.; Hitomi, K.; Chandrasiri, N.P.; Bando, T.; Yamashita, A. and Asama, H.: "Development of pedestrian behavior model taking account of intention". In: *International Conference on Intelligent Robots and Systems*. 2012, pp. 382–387. doi: 10.1109/IROS.2012.6385599 (cit. on p. 27).

[Tek02] Teknom, K.: "Microscopic Pedestrian Flow Characteristics: Development of an Image Processing Data Collection and Simulation Model". PhD thesis. Tohoku University, 2002 (cit. on pp. 123, 130, 141).

[Thr05] Thrun, S.; Burgard, W. and Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press, 2005 (cit. on pp. 22, 32, 38).

[Tie12] Tieleman, T. and Hinton, G.: Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning. 2012 (cit. on p. 84).

[Tra10] Trautman, P. and Krause, A.: "Unfreezing the robot: Navigation in dense, interacting crowds". In: *International Conference on Intelligent Robots and Systems*. 2010, pp. 797–803. doi: 10.1109/IROS.2010.5654369 (cit. on p. 21).

[Tug82] Tugnait, J.: "Adaptive estimation and identification for discrete systems with Markov jump parameters". In: *IEEE Transactions on Automatic Control* 27.5 (1982), pp. 1054–1065. doi: 10.1109/TAC.1982.1103061 (cit. on p. 43).

[van16] van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.W. and Kavukcuoglu, K.: "WaveNet: A Generative Model for Raw Audio". In: *arXiv preprint arXiv:1609.03499* (2016). arXiv: 1609.03499 (cit. on p. 101).

[Vas16] Vasquez, D.: "Novel planning-based algorithms for human motion prediction". In: *International Conference on Robotics and Automation (ICRA)*. 2016, pp. 3317–3322. doi: 10.1109/ICRA.2016.7487505 (cit. on p. 22).

[Vem18] Vemula, A.; Muelling, K. and Oh, J.: "Social Attention: Modeling Attention in Human Crowds". In: *International Conference on Robotics and Automation (ICRA)*. 2018, pp. 1–7. doi: 10.1109/ICRA.2018.8460504. url: https://doi.org/10.1109/ICRA.2018.8460504 (cit. on pp. 21, 89, 99).

[Völ15] Völz, B.; Mielenz, H.; Agamennoni, G. and Siegwart, R.: "Feature Relevance Estimation for Learning Pedestrian Behavior at Crosswalks". In: *International Conference on Intelligent Transportation Systems (ITSC)*. Sept. 2015, pp. 854–860. doi: 10.1109/ITSC.2015.144 (cit. on p. 29).

[Völ16] Völz, B.; Mielenz, H.; Siegwart, R. and Nieto, J.: "Predicting pedestrian crossing using Quantile Regression forests". In: *Intelligent Vehicles Symposium (IV)*. June 2016, pp. 426–432. doi: 10.1109/IVS.2016.7535421 (cit. on p. 29).

[Völ19] Völz, B.; Mielenz, H.; Gilitschenski, I.; Siegwart, R. and Nieto, J.: "Inferring Pedestrian Motions at Urban Crosswalks". In: *IEEE Transactions on Intelligent Transportation Systems* 20.2 (2019), pp. 544–555. doi: 10.1109/TITS.2018.2827956 (cit. on p. 29).

[Wak04] Wakim, C.F.; Capperon, S. and Oksman, J.: "A Markovian model of pedestrian behavior". In: *International Conference on Systems, Man and Cybernetics*. Vol. 4. 2004, 4028–4033 vol.4. doi: 10.1109/ICSMC.2004.1400974 (cit. on p. 27).

[Wan17] Wang, X.; Takaki, S. and Yamagishi, J.: "An autoregressive recurrent mixture density network for parametric speech synthesis". In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, pp. 4895–4899. doi: 10.1109/ICASSP.2017.7953087 (cit. on p. 89).

[Wan99] Wang, H.; Kirubarajan, T. and Bar-Shalom, Y.: "Precision large scale air traffic surveillance using IMM/assignment estimators". In: *IEEE Transactions on Aerospace and Electronic Systems* 35.1 (1999), pp. 255–266. doi: 10.1109/7.745696 (cit. on p. 54).

[Wen04] Wendel, J.; Metzger, J. and Trommer, G.: "Rapid Transfer Alignment in the Presence of Time Correlated Measurement and System Noise". In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. 2004, pp. 1–12 (cit. on p. 65).

[Wen11] Wendel, J.: Integrierte Navigationssysteme: Sensordatenfusion, GPS und Inertiale Navigation. Oldenbourg, 2011. url: https://books.google.de/books?id=9uSgZwEACAAJ (cit. on p. 43).

[Wil95] Williams, R.J. and Zipser, D.: "Backpropagation". In: ed. by Chauvin, Yves and Rumelhart, David E. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1995. Chap. Gradient-based Learning Algorithms for Recurrent Networks and Their Computational Complexity, pp. 433–486. url: http://dl.acm.org/citation.cfm?id=201784.201801 (cit. on pp. 14, 81).

[Xia15] Xiao, S.; Wang, Z. and Folkesson, J.: "Unsupervised robot learning to predict person motion". In: *International Conference on Robotics and Automation (ICRA)*. 2015, pp. 691–696. doi: 10.1109/ICRA.2015.7139254 (cit. on p. 21).

[Xu15] Xu, K.; Ba, J.; Kiros, R.; K.Cho; Courville, A.; Salakhudinov, R.; Zemel, R. and Bengio, Y.: "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". In: *International Conference on Machine Learning (ICML)*. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 2048–2057 (cit. on p. 93).

[Xue18] Xue, H.; Huynh, D.Q. and Reynolds, M.: "SS-LSTM: A Hierarchical LSTM Model for Pedestrian Trajectory Prediction". In: *Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018 (cit. on pp. 99, 103, 110, 111).

[Xue19] Xue, H.; Huynh, D.Q. and Reynolds, M.: "Location-Velocity Attention for Pedestrian Trajectory Prediction". In: *Winter Conference on Applications of Computer Vision (WACV)*. 2019, pp. 2038–2047. doi: 10.1109/WACV.2019.00221 (cit. on pp. 21, 29, 89, 110).

[Yam11] Yamaguchi, K.; Berg, A. C.; Ortiz, L. E. and Berg, T. L.: "Who are you with and where are you going?" In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011, pp. 1345–1352. doi: 10.1109/CVPR.2011.5995468 (cit. on pp. 20, 109, 110).

[Yed97] Yeddanapudi, M.; Bar-Shalom, Y. and Pattipati, K.: "IMM estimation for multitarget-multisensor air traffic surveillance". In: *Proceedings of the IEEE* 85.1 (1997), pp. 80–96. doi: 10.1109/5.554210 (cit. on p. 54).

[Zar09] Zarchan, P.; Musoff, H.; Lu, F.K.; Aeronautics, American Institute of and Astronautics: Fundamentals of Kalman Filtering: A Practical Approach. Progress in astronautics and aeronautics. American Institute of Aeronautics and Astronautics, 2009. url: https://books.google.de/books?id=8T-hMgEACAAJ (cit. on p. 39).

[Zei12] Zeiler, M.D.: "ADADELTA: An Adaptive Learning Rate Method". In: *arXiv preprint arXiv:1212.5701* (2012) (cit. on p. 84).

[Zer16] Zernetsch, S.; Kohnen, S.; Goldhammer, M.; Doll, K. and Sick, B.: "Trajectory prediction of cyclists using a physical model and an artificial neural network". In: *Intelligent Vehicles Symposium (IV)*. 2016, pp. 833–838. doi: 10.1109/IVS.2016.7535484 (cit. on pp. 20, 26).

[Zha19] Zhang, P.; Ouyang, W.; Zhang, P.; Xue, J. and Zheng, N.: "SR-LSTM: State Refinement for LSTM towards Pedestrian Trajectory Prediction". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 (cit. on pp. 21, 29, 89, 110).

[Zho17] Zhou, Yi; Li, Zimo; Xiao, Shuangjiu; He, Chong; Huang, Zeng and Li, Hao: "Auto-Conditioned Recurrent Networks for Extended Complex Human Motion Synthesis". In: *International Conference on Learning Representations (ICLR)*. 2017 (cit. on p. 103).

[Zie09] Ziebart, B.D.; Ratliff, N.; Gallagher, G.; Mertz, C.; Peterson, K.; Andrew Bagnell, J.; Hebert, M.; Dey, A.K. and Srinivasa, S.: "Planning-based Prediction for Pedestrians". In: *International Conference on Intelligent Robots and Systems (IROS)*. St. Louis, MO, USA: IEEE Press, 2009, pp. 3931–3936. url: http://dl.acm.org/citation.cfm?id=1732643.1732694 (cit. on pp. 22, 27).

[Zim12] Zimmermann, H.-G.; Tietz, C. and Grothmann, R.: "Forecasting with Recurrent Neural Networks: 12 Tricks". In: *Neural Networks: Tricks of the Trade - Second Edition*. 2012, pp. 687–707. doi: 10.1007/978-3-642-35289-8_37. url: https://doi.org/10.1007/978-3-642-35289-8%5C_37 (cit. on pp. 103, 105).

Publications

[1] Becker, S. and Jüngling, K.: "An implicit shape model based approach to identify armed persons". In: *Proc. SPIE 8049 Automatic Target Recognition XXI*. 2011.

[2] Jüngling, K.; Becker, S. and Arens, M.: "Hierarchical Object Detection and Tracking with an Implicit Shape Model". In: *Proc. of the International Conference on Image Processing, Computer Vision, and Pattern Recognition*. Las Vegas, Nevada, 2011, pp. 55–61.

[3] Becker, S.; Hübner, W. and Arens, M.: "IR-videostream rendering based on high-level object information". In: *Proc. SPIE 8541 Electro-Optical and Infrared Systems: Technology and Applications*. 2012.

[4] Münch, D.; Becker, S.; Hübner, W. and Arens, M.: "Towards a Real-time Situational Awareness System for Surveillance Applications in Unconstrained Environments". In: *Proc. of the 7th Future Security Research Conference*. 2012.

[5] Becker, S.; Voelcker, A.; Kieritz, H.; Hübner, W. and Arens, M.: "Automated Generation of High-Quality Training Data for Appearance-based Object Models". In: *Proc. SPIE Volume 8899C Unmanned/Unattended Sensors and Sensor Networks X*. 2013.

[6] Münch, D.; Becker, S.; Hübner, W. and Arens, M.: "Towards situational awareness systems based on semi-stationary multi-camera components". In: *Proc. of the 8th Future Security Research Conference*. 2013.

[7] Becker, S.; Hübner, W. and Arens, M.: "Independent motion detection with a rival penalized adaptive particle filter". In: *Proc. SPIE Volume 9248 Unmanned/Unattended Sensors and Sensor Networks XI*. 2014.

[8] Becker, S.; Hübner, W. and Arens, M.: "Annotation driven MAP Search Space Estimation for Sliding-Window based Person Detection". In: *IAPR International Conference on Machine Vision Applications (MVA)*. 2015.

[9] Becker, S.; Münch, D.; Kieritz, H.; Hübner, W. and Arens, M.: "Detecting Abandoned Objects using Interacting Multiple Models". In: *Proc. SPIE Volume 9652 Optics and Photonics for Counterterrorism, Crime Fighting, and Defence*. 2015.

[10] Becker, S.; Scherer-Negenborn, N.; Thakkar, P.; Hübner, W. and Arens, M.: "An evaluation of background subtraction algorithms on fused infrared-visible video streams". In: *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. 2015.

[11] Felsberg, M. et al.: "The Thermal Infrared Visual Object Tracking VOT-TIR2015 Challenge Results". In: *International Conference on Computer Vision Workshops (ICCVW)*. 2015.

[12] Kristan, M. et al.: "The Visual Object Tracking VOT2015 Challenge Results". In: *International Conference on Computer Vision Workshops (ICCVW)*. 2015.

[13] Münch, D.; Becker, S.; Kieritz, H.; Hübner, W. and Arens, M.: "Knowledge-based situational analysis of unusual events in public places". In: *Proc. of the 10th Future Security Research Conference*. 2015.

[14] Becker, S.; Kieritz, H.; Hübner, W. and Arens, M.: "On the benefit of state separation for tracking in image space with an Interacting Multiple Model Filter". In: *International Conference on Image and Signal Processing (ICISP)*. Springer International Publishing, 2016, pp. 3–11. doi: 10.1007/978-3-319-33618-3_1.

[15] Becker, S.; Krah, S.B.; Hübner, W. and Arens, M.: "MAD for visual tracker fusion". In: *Proc. SPIE Volume 9995 Optics and Photonics for Counterterrorism, Crime Fighting, and Defence*. 2016.

[16] Becker, S.; Scherer-Negenborn, N.; Thakkar, P.; Hübner, W. and Arens, M.: "The effects of camera jitter for background subtraction algorithms on fused infrared-visible video streams". In: *Proc. SPIE Volume 9995 Optics and Photonics for Counterterrorism, Crime Fighting, and Defence*. 2016.

[17] Felsberg, M. et al.: "The Thermal Infrared Visual Object Tracking VOT-TIR2016 Challenge Results". In: *European Conference on Computer Vision Workshops (ECCVW)*. 2016.

[18] Kieritz, H.; Becker, S.; Hübner, W. and Arens, M.: "Online Multi-Person Tracking using Integral Channel Features". In: *Conference on Advanced Video and Signal-based Surveillance (AVSS)*. 2016.

[19] Kristan, M. et al.: "The Visual Object Tracking VOT2016 Challenge Results". In: *European Conference on Computer Vision Workshops (ECCVW)*. 2016.

[20] Münch, D.; Hilsenbeck, B.; Kieritz, H.; Becker, S.; Grosselfinger, A.-K.; Hübner, W. and Arens, M.: "Detection of infrastructure manipulation with knowledge-based video surveillance". In: *Proc. SPIE Volume 9995 Optics and Photonics for Counterterrorism, Crime Fighting, and Defence*. 2016.

[21] Münch, D.; Kieritz, H.; Becker, S.; Hübner, W. and Arens, M.: "Video-based log generation for security systems in indoor surveillance scenarios". In: *Proc. of the 11th Future Security Research Conference*. 2016.

[22] Abel, P.; Kieritz, H.; Becker, S. and Arens, M.: "Robust visual object tracking with interleaved segmentation". In: *Proc. SPIE Volume 10441 Counterterrorism, Crime Fighting,Forensics and Surveillance*. 2017. doi: 10.1117/12.2275259.

[23] Hug, R.; Becker, S.; Hübner, W. and Arens, M.: "On the reliability of LSTM-MDL models for pedestrian trajectory prediction". In: *Representations, Analysis and Recognition of Shape and Motion from Imaging Data (RFMI)*. Savoie, France, 2017.

[24] Becker, S.; Hübner, W. and Arens, M.: "State estimation for tracking in image space with a de- and re-coupled IMM filter". In: *Multimedia Tools and Applications* 77.15 (2018), pp. 20207–20226. doi: 10.1007/s11042-017-5324-3.

[25] Becker, S.; Hug, R.; Hübner, W. and Arens, M.: "An Evaluation of Trajectory Prediction Approaches and Notes on the TrajNet Benchmark". In: *arXiv preprint arXiv:1805.07663* (2018).

[26] Becker, S.; Hug, R.; Hübner, W. and Arens, M.: "RED: A simple but effective Baseline Predictor for the TrajNet Benchmark". In: *European Conference on Computer Vision (ECCV) Workshops*. Springer International Publishing, 2018.

[27] Hug, R.; Becker, S.; Hübner, W. and Arens, M.: "Particle-based Pedestrian Path Prediction using LSTM-MDL Models". In: *21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2684–2691. doi: 10.1109/ITSC.2018.8569478.

[28] Becker, S.: "RNN-based Prediction of Pedestrian Turning Maneuvers". In: *Proceedings of the Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory*. KIT Scientific Publishing, 2019.

[29] Becker, S.; Hug, R.; Hübner, W. and Arens, M.: "An RNN-based IMM Filter Surrogate". In: *Scandinavian Conference on Image Analysis (SCIA)*. Vol. 11482. Springer International Publishing, 2019, pp. 387–398.

[30] Vujasinovic, S.; Becker, S.; Scherer-Negenborn, N. and Arens, M.: "A comparison study of deep visual tracking on infrared imagery in a maritime environment". In: *Proc. SPIE Volume 11166 Counterterrorism, Crime Fighting, Forensics, and Surveillance Technologies*. 2019.

[31] Vujasinovic, S.; Becker, S.; Scherer-Negenborn, N. and Arens, M.: "Impact of Fused Visible-Infrared Video Streams on Visual Tracking". In: *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*. Springer International Publishing, 2019.

Supervised student theses

[1] Voelcker, A.: "Automated Generation of Training Samples for Appearance-Based Person-Models". Master thesis. Karlsruhe Institute of Technology (KIT), 2013.

[2] Abel, P.: "Robust Object Tracking with Interleaved Detection and Segmentation". Master thesis. Karlsruhe Institute of Technology (KIT), 2017.

[3] Kostov, T.: "Integration of Observation Uncertainty into an RNN-based Prediction Network". Master thesis. Karlsruhe Institute of Technology (KIT), 2019.

List of Figures

List of Tables

Acronyms

ADAM	**adaptive moment estimation**
ADE	**average displacement error**
BPTT	**backpropagation through time**
CA	**constant acceleration**
CP	**constant position**
CRF	**conditional random field**
CT	**constant turn**
CV	**constant velocity**
DBN	**dynamic Bayesian network**
DS	**dynamical system**
EKF	**extended Kalman filter**
ELU	**exponential linear unit**
FDE	**final displacement error**

fps	**frames per second**
GPDM	**Gaussian processes with dynamic model**
GRU	**gated recurrent unit**
HMM	**hidden Markov model**
IMM	**interacting multiple-model**
IoU	**intersection-over-union**
JMLS	**jump Markov linear system**
KF	**Kalman filter**
LSTM	**long short-term memory**
MCMC	**Markov-chain-Monte-Carlo**
MDN	**mixture density network**
MLP	**multi-layer perceptron**
pdf	**probability density function**
PF	**particle filter**
pmf	**probability mass function**
RED	**RNN-encoder with a dense MLP stacked on top**
RMSE	**root-mean-squared error**
RNN	**recurrent neural network**

SGD	**stochastic gradient descent**
SLDS	**switching linear dynamical system**
SSSM	**switching state-space model**
SVM	**support-vector-machine**
TCN	**temporal convolutional network**
TPM	**transition probability matrix**
TTE	**time-to-event**
UKF	**unscented Kalman filter**