DGK Deutsche Geodätische Kommission
der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 710

Alexander Barth

Visual SLAM from image sequences acquired by unmanned aerial vehicles

München 2013

Verlag der Bayerischen Akademie der Wissenschaften
in Kommission beim Verlag C. H. Beck

ISSN 0065-5325 ISBN 978-3-7696-5122-5

Diese Arbeit ist gleichzeitig veröffentlicht in:
Schriftenreihe des Instituts für Geodäsie und Geoinformation
der Rheinischen Friedrich-Wilhelms Universität Bonn
ISSN 1864-1113, Nr. 34, Bonn 2009

DGK Deutsche Geodätische Kommission
der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 710

Visual SLAM from image sequences acquired by unmanned aerial vehicles

Inaugural-Dissertation zur
Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
der Hohen Landwirtschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms Universität
zu Bonn

vorgelegt

am 27.09.2010 von

Alexander Barth

aus Lüdenscheid

München 2013

Verlag der Bayerischen Akademie der Wissenschaften
in Kommission beim Verlag C. H. Beck

ISSN 0065-5325 ISBN 978-3-7696-5122-5

Diese Arbeit ist gleichzeitig veröffentlicht in:
Schriftenreihe des Instituts für Geodäsie und Geoinformation
der Rheinischen Friedrich-Wilhelms Universität Bonn
ISSN 1864-1113, Nr. 34, Bonn 2009

Adresse der Deutschen Geodätischen Kommission:

Deutsche Geodätische Kommission

Alfons-Goppel-Straße 11 ● D – 80 539 München

Telefon +49 – 89 – 23 031 1113 ● Telefax +49 – 89 – 23 031 - 1283 / - 1100
e-mail hornik@dgfi.badw.de ● http://www.dgk.badw.de

Diese Publikation ist als pdf-Dokument veröffentlicht im Internet unter der Adresse /
This volume is published in the internet

<http://dgk.badw.de> ● <http://hss.ulb.uni-bonn.de/2010/2356/2356.htm>

Prüfungskommission

Referent:	Prof. Dr.-Ing. Wolfgang Förstner
1. Korreferent:	Prof. Dr. techn. Wolf–Dieter Schuh

Tag der mündlichen Prüfung: 26.11.2010

ISSN 0065-5325 ISBN 978-3-7696-5122-5

Deutsche Zusammenfassung

In dieser Dissertation wird ein Ansatz zur Trajektorienschätzung von Straßenfahrzeugen (PKW, Lieferwagen, Motorräder,...) anhand von Stereo-Bildfolgen vorgestellt. Bewegte Objekte werden in Echtzeit aus einem fahrenden Auto heraus automatisch detektiert, vermessen und deren Bewegungszustand relativ zum eigenen Fahrzeug zuverlässig bestimmt. Die gewonnenen Informationen liefern einen entscheidenden Grundstein für zukünftige Fahrerassistenz- und Sicherheitssysteme im Automobilbereich, beispielsweise zur Kollisionsprädiktion.

Während der Großteil der existierenden Literatur das Detektieren und Verfolgen vorausfahrender Fahrzeuge in Autobahnszenarien adressiert, setzt diese Arbeit einen Schwerpunkt auf den Gegenverkehr, speziell an städtischen Kreuzungen. Der Ansatz ist jedoch grundsätzlich generisch und skalierbar für eine Vielzahl an Verkehrssituationen (Innenstadt, Landstraße, Autobahn).

Die zu schätzenden Parameter beinhalten die räumliche Lage des anderen Fahrzeugs relativ zum eigenen Fahrzeug, die Objekt-Geschwindigkeit und -Längsbeschleunigung, sowie die Rotationsgeschwindigkeit (Gierrate) des beobachteten Objektes. Zusätzlich werden die Objektabmaße sowie die Objektform rekonstruiert.

Die Grundidee ist es, diese Parameter anhand der Transformation von beobachteten 3D Punkten, welche eine ortsfeste Position auf der Objektoberfläche besitzen, mittels eines rekursiven Schätzers (Kalman Filter) zu bestimmen. Ein wesentlicher Beitrag dieser Arbeit liegt in der Kombination des Starrkörpermodells der Punktewolke mit einem Fahrzeugbewegungsmodell.

An Kreuzungen können sehr unterschiedliche Dynamiken auftreten, von einer Geradeausfahrt mit konstanter Geschwindigkeit bis hin zum raschen Abbiegen. Um eine manuelle Parameteradaption abhängig von der jeweiligen Szene zu vermeiden, werden drei verschiedene Ansätze zur automatisierten Anpassung der Filterparameter an die vorliegende Situation vorgestellt und verglichen. Dies stellt den zweiten Hauptbeitrag der Arbeit dar.

Weitere wichtige Beiträge sind zwei alternative Initialisierungsmethoden, eine robuste Ausreißerbehandlung, ein probabilistischer Ansatz zur Zuordnung neuer Objektpunkte, sowie die Fusion des bildbasierten Verfahrens mit einem Radar-Sensor.

Das Gesamtsystem wird im Rahmen dieser Arbeit systematisch anhand von simulierten und realen Straßenverkehrsszenen evaluiert. Die Ergebnisse zeigen, dass das vorgestellte Verfahren in der Lage ist, die unbekannten Objektparameter auch unter schwierigen Umgebungsbedingungen, beispielsweise bei Nacht, schnellen Abbiegemanövern oder unter Teilverdeckungen, sehr präzise zu schätzen. Die Grenzen des Systems werden ebenfalls sorgfältig untersucht.

Abstract

In this dissertation, a novel approach for estimating trajectories of road vehicles such as cars, vans, or motorbikes, based on stereo image sequences is presented. Moving objects are detected and reliably tracked in real-time from within a moving car. The resulting information on the pose and motion state of other moving objects with respect to the own vehicle is an essential basis for future driver assistance and safety systems, e.g., for collision prediction.

The focus of this contribution is on oncoming traffic, while most existing work in the literature addresses tracking the lead vehicle. The overall approach is generic and scalable to a variety of traffic scenes including inner city, country road, and highway scenarios. A considerable part of this thesis addresses oncoming traffic at urban intersections.

The parameters to be estimated include the 3D position and orientation of an object relative to the ego-vehicle, as well as the object's shape, dimension, velocity, acceleration and the rotational velocity (*yaw rate*).

The key idea is to derive these parameters from a set of tracked 3D points on the object's surface, which are registered to a time-consistent object coordinate system, by means of an extended Kalman filter. Combining the rigid 3D point cloud model with the dynamic model of a vehicle is one main contribution of this thesis.

Vehicle tracking at intersections requires covering a wide range of different object dynamics, since vehicles can turn quickly. Three different approaches for tracking objects during highly dynamic turn maneuvers up to extreme maneuvers such as skidding are presented and compared. These approaches allow for an online adaptation of the filter parameter values, overcoming manual parameter tuning depending on the dynamics of the tracked object in the scene. This is the second main contribution.

Further issues include the introduction of two initialization methods, a robust outlier handling, a probabilistic approach for assigning new points to a tracked object, as well as mid-level fusion of the vision-based approach with a radar sensor.

The overall system is systematically evaluated both on simulated and real-world data. The experimental results show the proposed system is able to accurately estimate the object pose and motion parameters in a variety of challenging situations, including night scenes, quick turn maneuvers, and partial occlusions. The limits of the system are also carefully investigated.

Acknowledgements

Writing a dissertation involves the efforts and generous support from many people.

First, I would like to thank my doctoral advisor Prof. Dr. Wolfgang Förstner (University of Bonn) and Dr. Uwe Franke (Daimler AG) for mentoring my PhD work and giving me valuable support throughout all stages of my research. Discussing my ideas with these excellent scientists has provided me with an optimal balance between theoretical aspects and practical engineering expertise.

The warm, creative, and productive atmosphere at the Daimler Environment Perception group in Böblingen, Germany, has provided me with a perfect working environment, helping my ideas flourish. My gratitude goes to all colleagues, co-authors, students, and friends at Daimler for supporting my work with fruitful discussions and the state-of-the-art technological platform on which I could base my work.

In addition, I would like to particularly thank Dr. Stefan Gehrig, Prof. Dr. Christian Wöhler, Clemens Rabe, Dr. Andreas Wedel, Heidi Loose, David Pfeiffer, Thomas Müller, Friedrich Erbs, and Annemarie Meissner.

My special thank goes to Clemens for his amazing software tools and to Heidi for keeping up my health with citrus fruits.

Although mainly located in Böblingen, I have always felt very welcome at the image processing group at the University of Bonn, Department of Photogrammetry, which I gratefully appreciate. Special thanks to Jan Siegemund and Dr. Richard Steffen for their patience and imagination, while discussing mathematical equations and ideas on endless *just-one-minute* phone calls.

Special credits also to Tobi Vaudrey and Sanaz Jahanbakhsh for proof-reading my thesis and papers, to my family and friends for constantly supporting me during this time, and to my lovely fiancée. Thank you, Steffi!

I finally want to express my gratitude to the members of the committee, Prof. Dr. Wolf-Dieter Schuh, Prof. Dr. Jürgen Kusche, and Prof. Dr. Lutz Plümer.

Contents

Notation

Symbol	Description
a_q	centripetal acceleration
A	Kalman filter system matrix
b	stereo base line
C	center of mass
C	covariance matrix
$\mathsf{C}_{\widehat{xx}}$	covariance matrix of estimated parameters
C_{rr}	covariance matrix of the Kalman filter *measurement/prediction residual*
C_{vv}	Kalman filter measurement noise matrix
C_{ww}	Kalman filter system noise matrix
d	stereo disparity
f	principal distance
f_x	scaled principal distance in x-direction
f_y	scaled principal distance in y-direction
H	Kalman filter measurement matrix
k	discrete time index
K	Kalman gain matrix
L	distance between frontal and rear axle
m_g	object mass
${}^{a}\mathsf{M}$	homogeneous coordinate transformation over two discrete time steps $k-1 \rightarrow k$ w.r.t. frame a (motion matrix)
M	number of point measurements
$\boldsymbol{O}_a$	coordinate origin of system S_a
$\boldsymbol{O}$	object representation
$\boldsymbol{\Omega}$	object pose parameters
$\boldsymbol{\Phi}$	object motion parameters
$\boldsymbol{\Theta}$	object shape parameters
$\boldsymbol{p}$	pixel coordinate $[u, v]^{\mathsf{T}}$
$\mathcal{P}$	name of a particular 3D point
${}^{a}\boldsymbol{P}$	3D point $\mathcal{P}$ in frame a, ${}^{a}\boldsymbol{P} = [{}^{a}X, {}^{a}Y, {}^{a}Z]^{\mathsf{T}}$
${}^{a}\mathbf{P}$	point ${}^{a}\boldsymbol{P}$ in homogeneous coordinates, ${}^{a}\mathbf{P} = [{}^{a}X, {}^{a}Y, {}^{a}Z, 1]^{\mathsf{T}}$
$\mathcal{P}({}^{a}\boldsymbol{P})$	meaning: $\mathcal{P}$ *is represented by* ${}^{a}\boldsymbol{P}$
${}^{c}\boldsymbol{P}$	3D point in camera coordinate system
${}^{e}\boldsymbol{P}$	3D point in ego-vehicle coordinate system
${}^{o}\boldsymbol{P}$	3D point in object coordinate system
${}^{v}\boldsymbol{P}$	3D point in vehicle coordinate system
$\boldsymbol{r}$	residual between measurements and prediction
R	radius

$\mathsf{R}(\boldsymbol{p}_{\mathrm{rot}})$	rotation matrix with parameter vector $\boldsymbol{p}_{\mathrm{rot}}$
$\mathsf{R}_y(\psi)$	rotation matrix around Y-axis about Euler angle ψ
s	arc length
S_a	coordinate system with name a
t	continuous time
Δt	discrete time interval between two images
T	threshold
$\boldsymbol{T}$	translation vector $\boldsymbol{T} = [\,X,\, Y,\, Z]^{\mathsf{T}}$
$\mathbf{T}$	homogeneous translation vector $\mathbf{T} = [\,X,\, Y,\, Z, 1]^{\mathsf{T}}$
${}^{b}\boldsymbol{T}_{a}$	translation vector from $\boldsymbol{O}_a$ to $\boldsymbol{O}_b$
u	horizontal pixel coordinate
v	vertical pixel coordinate
$\boldsymbol{w}$	additive Gaussian system noise vector
${}^{b}\mathsf{W}_{a}$	homogeneous coordinate transformation from frame S_a to frame S_b at one time step k
${}^{b}\mathsf{W}_{a}^{-1}$	homogeneous coordinate transformation from frame S_b to frame S_a at one time step k, i.e., ${}^{b}\mathsf{W}_{a}^{-1} = {}^{a}\mathsf{W}_{b}$
$\boldsymbol{x}$	Kalman filter state vector
$\boldsymbol{x}_{\mathrm{full}}$	Kalman filter state vector including object point coordinates
$\boldsymbol{x}_{\mathrm{red}}$	reduced Kalman filter state vector (pose and motion parameters only)
$\tilde{\boldsymbol{x}}$	true value of $\boldsymbol{x}$
$\widehat{\boldsymbol{x}}$	estimated value of $\boldsymbol{x}$
$\widehat{\boldsymbol{x}}^{+}$	a posteriori state estimate
$\widehat{\boldsymbol{x}}^{-}$	a priori state estimate
$\boldsymbol{z}$	measurement vector
$\widehat{\boldsymbol{z}}$	predicted measurement vector
ρ	distance from object rear to rear axle
β	angle between vehicle and object system
ψ	yaw angle (rotation about height axis)
ϕ	pitch angle (rotation about lateral axis)
θ	roll angle (rotation about longitudinal axis)
$\dot{\psi}$	yaw rate
$\ddot{\psi}$	yaw acceleration
v	velocity in longitudinal direction
$\dot{v}$	acceleration in longitudinal direction
χ	moving direction
δ_s	steering angle
ϱ	wheel specific coefficient
π_{img}	image plane

1. Introduction

Making cars *perceive* and *understand* their environment is a challenging task that has been investigated by many researchers in the past two decades. The idea of *intelligent vehicles*, though, is no longer a vision as there are many commercial driver assistance systems available these days, allowing for autonomous parking, adaptive cruise control, lane departure warning, traffic sign recognition, or obstacle detection.

Future driver assistance and safety systems aim to support the driver in more and more complex driving situations, allowing for safe and stress-free driving. Fast and reliable knowledge of moving objects' location as well as their movement patterns relative to the ego-vehicle, which is the main contribution of this thesis, will be an essential basis for such systems.

1.1. Motivation

The number of traffic deaths in Germany has been significantly decreased by about two third since 1997 [Statistisches Bundesamt Wiesbaden 2009], which is related to improvements in the infrastructure, more restrictive laws, e.g. drink-drive-limit, and an increase in active and passive safety of todays vehicles. However, the total number of traffic accidents remains approximately constant. In 2008, 15.7% of all road accidents in Germany with damage to persons happened at intersections and during turning maneuvers. Intersections are accident hot spots and, thus, of special interest for future driver assistance systems.

Following a study on left turn across path crashes of the U.S. department for transportation from 1994, at 49% of all accidents of left turning vehicles at intersections, the drivers were unaware of the oncoming vehicle, and 30% were caused by drivers who saw but misjudged the velocity or gap of the oncoming vehicle [Chovan et al. 1994]. Most accidents occur at daylight conditions (74%). At more than 80% of all occurrences, the pavement has been dry and there were no adverse weather conditions.

Driver assistance systems, that can help the driver to prevent such collisions, have to be able to answer the question, where are other traffic participants and how do they move? There are two main strategies to answer this question: Telematics and environment perception by sensing.

In telematics, vehicles and infrastructure units send and receive information on their position, motion state (e.g., velocity), or status (braking, break-down, red traffic light,...), which is referred to as *car-to-car* or *car-to-infrastructure* communication. The advantage of such systems is that instances (cars, traffic lights,...) typically know their own state quite accurately and can communicate the information without requiring intervisibility. The drawback is that objects not equipped with communication hardware such as pedestrians, obstacles on the road, or old cars, are not *visible* for such systems. Another issue is data integrity and communication security.

The alternative are vehicles that are equipped with sensors, e.g., video cameras, radar sensors, or laser scanners, to perceive the surrounding environment. Such vehicles are not restricted to perceive what other objects are broadcasting. However, extracting the same information reliably and accurate from a complex environment is a challenging task. These systems have to deal with a limited field of view, (partial) occlusions, and sensor specific problems, e.g., at bad weather conditions such as heavy rain or fog.

This thesis addresses the detection and tracking of other traffic participants, using a stereo vision sensor, with a special focus on oncoming traffic at intersections.

1.2. State Of The Art

Vehicle detection and tracking will play a key role in future driver assistance and active safety systems. The following sections give a comprehensive overview on the state of the art in driver assistance systems as well as on published approaches for vehicle tracking that are to some amount related to the present work.

1.2.1. Driver Assistance Systems

Driver assistance systems support the driver in a wide range of situations. They can be categorized into *safety systems* and *comfort systems*.

One of the earliest **safety systems** that have established are the anti-lock braking system (ABS) and the Electronic Stability Program (ESP) system, offering improved vehicle control at critical situations such as heavy braking [Dietsche and Jäger 2003]. These systems are based on inertial sensors.

Lane departure assistance systems are an example of systems that require environment perception, e.g., by sensors such as video cameras. There are different strategies to react to detected lane departure, such as warning the driver acoustically or haptically, or to perform an autonomous intervention that steers the vehicle back into the lane (e.g. Toyota, Honda).

Collision avoidance systems, such as the Mercedes-Benz brake assist, are systems that monitor the environment around the ego-vehicle and predict potential collisions with obstacles in the driving corridor. This includes blind spot monitoring systems that assist the driver at lane changes. If a potential collision is detected, the system warns the driver, or provides support at braking. There exist even systems that perform fully autonomous braking (e.g. Volvo City Safety). Collision avoidance systems have high demands on the environment perception, since traffic situations can be extremely complex [Gehrig and Stein 2007; Polychronopoulos et al. 2007; Kaempchen et al. 2009]. In case of autonomous interaction, any false interpretations may lead to accidents that would not have happened without the system.

Drowsiness detection systems, as proposed in 2009 for the Mercedes-Benz E-class, are able to assess the behavior of the driver and can, for example, suggest a coffee break if the behavior indicates the driver is getting tired.

Comfort systems, such as GPS-based navigation systems, have established as standard equipment in many cars. Such systems incorporate, for example, up-to-date traffic information to suggest alternative routes in case of traffic jams.

One common example for comfort systems requiring active sensing of the environment is the Adaptive Cruise Control (ACC) system. It allows for automated adaptation of the vehicle speed to the lead vehicle, or a preset velocity if there is no lead vehicle in the current traffic situation [Dietsche and Jäger 2003]. These systems have to be able to detect and track the lead vehicle reliable, as well as to handle cut-in traffic.

Other comfort systems that require environment perception include traffic sign recognition/ speed warning, intelligent head lights, parking assistance, or night vision. A comprehensive overview on driver assistance system is given, e.g., in [Färber 2004].

While all the driver assistance systems above help the driver at a particular task, the peak of such driver assistance systems is an autonomous (intelligent) vehicle. A lot of pioneer work in the field of autonomous driving has been done from 1986 to 1995 in the *PROMETHEUS* project ("PROgraMme for a European Traffic of Highest Efficiency and Unprecedented Safety"), an outstanding colaboration of automotive companies and universities in Europe. The promising results have influenced the research and development of the past 15 years. One important contribution was the usage of video cameras and machine vision for lane and obstacle detection tasks. The DARPA *Grand Challenges* in 2004 and 2005 and the *Urban Challenge* in 2007 have been an incentive to continue the development of autonomous vehicles that are able to perform, for example, lane following, obstacle avoidance, precedence evaluation amongst other cars at intersections, or parking [Team MIT 2007; Kammel et al. 2007] without a human driver in the car.

1.2.2. Vehicle Tracking Using Computer Vision

Detecting and tracking vehicles has been explored by many researchers in the computer vision and Intelligent Transportation Systems (ITS) community over the past two decades. This section gives an overview on different approaches that are to some amount related to the presented vehicle tracking approach.

One can distinguish between the general tasks of *object detection*, *object classification*, and *object tracking*. Object detection is related to identifying *where* a particular object of interest is with respect to a given reference system. Object classification corresponds to recognizing *what* has been detected, and object tracking considers observing a detected object over time, e.g., to extract *how* an object moves. The latter corresponds to the main issue of this thesis. In practice, the boundaries between these categories are not strict. Most applications proposed in the literature use combinations of the three categories above, thus, a different categorization is chosen to structure this section.

It is distinguished between approaches with a stationary sensor platform and systems with the sensors mounted to a moving platform. Although the focus of this thesis will be on the latter, some techniques and concepts used with stationary cameras are also applicable to the non-stationary case. Among the moving platform approaches it will be further distinguished between monocular and stereoscopic ones, as well as approaches fusing vision with active sensors. Finally, this literature review focuses on different tracking methods used in the field of vision-based vehicle tracking.

Vehicle Tracking From Stationary Cameras

Many vision-based vehicle tracking systems with application to traffic surveillance have been proposed in the literature. In such system, stationary cameras are usually placed at elevated positions, monitoring for example highway sections or intersections [Koller et al. 1993; Koller et al. 1994; Beymer et al. 1997; Kim and Malik 2003; Atev et al. 2005; Jianguang et al. 2005; Jiang et al. 2005; Kanhere and Birchfield 2008; Morris and Trivedi 2008; Ottlik and Nagel 2008].

In this field, object detection often involves segmenting moving objects from a static [Ebbecke et al. 1997; Kamijo et al. 2000] or adaptive [Karmann and Brandt 1990; Stauffer and Grimson 1999] background model using background subtraction.

Deviations from the background model are often thresholded, yielding a binary image where each pixel represents either foreground or background. Connected foreground pixels are then grouped and further analyzed [Chen et al. 2001; Veeraraghavan and Papanikolopoulos 2004; Jiang et al. 2005; Choi et al. 2006; Xie et al. 2006; Zhi-fang and Zhisheng 2007; Mosabbeb et al. 2007].

In [Koller et al. 1993], moving objects are discriminated from the background on the basis of image flow. In a motion segmentation step, clusters of image positions showing mainly translational displacements between consecutive frames, are assumed to belong to single vehicles. For each cluster, the enclosing rectangle gives a subimage likely to contain the detected vehicle. Then, a parametrized 3D vehicle shape model is projected onto the image plane, and aligned to edges in this subimage for estimation of a vehicle's pose. Finally, detected vehicles are tracked by means of an extended Kalman filter using a 3D vehicle motion model. The state parameters include the pose, i.e., position and orientation, as well as the translational and rotational velocity. Similar approaches can be found, for example, in [Kim and Malik 2003; Jianguang et al. 2005; Buch et al. 2009]

Beymer et al. [Beymer et al. 1997] introduced a model free object representation based on groups of corner features to yield more stable tracking results in dense traffic situations. In this approach, objects are detected based on the *Law of Common Fate* concept of Gestalt psychologists [Goldstein 2001]. The idea is that a group of points moving rigidly together is assumed to belong to the same vehicle. Obviously, this property requires additional processing at dense traffic scenes if more than one object is moving with equal velocity in the same direction side-by-side or bumper-to-bumper. This work has been extended by Kanhere and Birchfield [Kanhere and Birchfield 2008] for applications with low camera angles.

Leotta and Mundy [Leotta and Mundy 2007] track a set of contour segments instead of corner features, to estimate the 3D translational motion of vehicles, also from low camera angle. However, this approach does not work for rotational movements.

Vehicle Tracking From Moving Platforms

If the camera is mounted in a car, many a priori constraints introduced in terms of static camera setups do not hold as the ego-vehicle is driving through an unknown area. Depending on the ground level and driving maneuver, camera movements cover all six degrees of freedom. Thus, image-based methods for compensating the *ego-motion* have been proposed, e.g., [H. Badino 2004; Klappstein 2008], to distinguish static from independently moving points in the scene. Additional robustness is achieved if information on the vehicle speed and yaw rate is provided by inertial sensors.

Monocular Systems: From the early publications to the present, most of the research on vision based vehicle detection and tracking from a moving platform addresses tracking leading vehicles on highways. These approaches consider, for example, image statistics such as edges [Dellaert and Thorpe 1997; Ferryman et al. 2000; Zeng and Ma 2002], grey level or color histograms [She et al. 2004; Liu et al. 2007b], or symmetry [Brauckmann et al. 1994; Toulminet et al. 2006; Liu et al. 2007a]. Others utilize optical flow [Smith and Brady 1995], contours [Dahlkamp et al. 2004], template matching [Richter et al. 2008], classification techniques [Fu et al. 2006; Maehlisch et al. 2007], and combinations of the former [Chen et al. 2007] to detect and track vehicles in an image. A good survey on different vehicle detection methods from a moving platform using optical sensors is given, for example, in [Sun et al. 2004]. Many of these methods, e.g., the symmetry of a vehicle rear side or characteristic shadow edges between the vehicle and the road, are designed for highway scenarios only and can hardly be transferred to a generic solution for vehicle tracking in arbitrary scenarios.

Leibe et al. have proposed an outstanding monocular vehicle detection and tracking approach, which is based on the combination of depth from structure from motion and *appearance* [Leibe et al. 2007]. Local classifiers are trained to detect characteristic objects parts (vehicles and pedestrians) in the 2D image. Each part votes for a given object center position. This method is far from real-time by now.

All these methods above are realized with a single camera. In this case, the distance to an object is often estimated based on the *structure from motion* principle or by detecting an object's base point on the planar ground plane, which is assumed to be known from camera calibration.

Stereoscopic Systems: Alternatively, stereo vision is used for depth estimation. Van der Mark and Gavrila [Mark and Gavrila 2006] provide a good overview on stereo vision in the intelligent vehicle domain, including an extensive evaluation of different real-time stereo implementations.

In many stereo vision based object detection approaches, objects are modeled as upright planes on a ground plane. Such planes can be identified, for example, based on an accumulation of equal distances (or disparities) within an image region as in [Franke and Kutzbach 1996]. The ground plane (road) does not necesseraly have to be flat. A solution for dealing with non-flat roads using so called *v-disparity* images has been proposed in [Labayrade et al. 2002]. Toulminet et al. [Toulminet et al. 2006] present a method combining stereoscopic and monocular cues for vehicle detection.

Given 3D point cloud data from stereo vision, several approaches fit geometrical models to this data that approximate the vehicle shape, e.g., a cuboid [Danescu et al. 2007; Barrois et al. 2009]. Such approaches perform well as long as the model is a sufficient approximation of the real object, and the data is reliable.

In [Hahn et al. 2010], an approach for object tracking and motion estimation based on stereo vision, optical flow, and mean shift clustering techniques has been proposed.

In this work, promising results for estimating the position of oncoming and crossing traffic participants in a roundabout scenario are presented. Further details on this work can be found in [Einhaus 2010].

A stereo sensor cannot distinguish between static obstacles and moving objects without linking the information between consecutive time steps. Furthermore, nearby objects are likely to be merged. Thus, methods fusing the depth information from stereo with motion have been proposed, for example in [Franke and Heinrich 2002; Dang et al. 2002]. Bachmann [Bachmann 2009] has proposed a recursive expectation-maximization (EM) framework that tracks and segments rigidly moving objects based on a 3D motion field.

Sensor Fusion: Beside the approaches using vision sensors for vehicle detection and tracking, there exist many other approaches based on active sensors such as radar (radio detection and ranging) or lidar (light detection and ranging) sensors. A detailed description of these approaches is outside the scope of this thesis. However, recently a lot of work concentrates on sensor fusion of vision-based methods with active sensors.

Radar and vision is fused, for example, in the following publications [Möbus and Kolbe 2004; Haselhoff et al. 2007; Liu et al. 2008], while [Maehlisch et al. 2006; Kaempchen 2007; Wender 2008; Effertz 2008; Schneider et al. 2010] combine lidar and vision information. An example that fuses all three sensors, i.e., radar, lidar and vision has been proposed in [Weiss et al. 2004].

The objective of sensor fusion is to combine the advantages of multiple sensors in a way that the shortcomings are reduced to a minimum. One can distinguish between *low-level fusion* and *high-level fusion* [Wender 2008].

Low-level fusion means fusing the (raw) data of the sensors at a very early processing step. At this stage, the system has access to the complete sensor information, thus, allowing for extremely specialized detection and tracking algorithms.

High-level fusion, on the other hand, combines information that has been processed and reduced at lower levels before. An example is the fusion of object tracks computed from different sensors. The advantage of high-level fusion is a good scalability. As long as a specified interface is served by a sensor, it can be easily replaced by another version or different manufacturer. Furthermore, with respect to product safety, taking a risky decision, such as autonomous braking, becomes more reliable if more than one sensor has detected the same critical object independently.

In this thesis, radar information is optionally fused with stereo vision data, as will be presented in Sec. 3.5.5 and 3.8.2. Since raw vision data is combined with high-level radar objects, this can be seen in between both types of sensor fusion.

Tracking Strategies

In many approaches, *tracking* is related to rediscovering an image region labeled as *vehicle* in the next frame such as in [Rajagopalan and Chellappa 2000; Bertozzi et al. 2000; Li et al. 2004]. Other approaches predict the object position in the image plane based on the optical flow [Liu et al. 2007c].

However, if one wants to be able to predict other traffic participant's trajectories in 3D space, one has to estimate the motion state of an observed object based on a corresponding 3D motion model. There are two main classes of tracking methods that are able to incorporate such motion models: *Kalman filter* based methods and *particle filter* based methods.

The Kalman filter, [Kalman 1960], is the most popular tracking technique in this field. It consists of a prediction and an update step. The former incorporates the motion model, while the latter considers the actual measurements (see Sec. 2.6 for details). There exist several variants and extensions, including the non-linear (extended) Kalman filter, the unscented Kalman filter, or multi-filter approaches. Kalman filters are used, for example, in combination with a linear motion model in [Dang et al. 2002], or with particular vehicle motion models incorporating specific properties of vehicle movements [Dellaert and Thorpe 1997; Zhao and Thorpe 1998; Leibe et al. 2007]. The different motion models used in the context of vehicle tracking will be addressed in detail in Sec. 2.5.

Throughout this thesis, the term **vehicle tracking** is used synonymously for estimating the motion state of a given vehicle in 3D space.

Other than the Kalman filter, the particle filter does not assume a Gaussian probability distribution of the estimated parameters [Thrun et al. 2005]. Instead, the posterior probability density function of a state estimate is represented by a set of (random) sample state vectors drawn from this distribution (*particles*). This allows for modeling more complex distributions than Gaussians as well as nonlinear transformations of random variables. The evolution of the particle set can be steered via a proper motion model. A particle filter is used for tracking the 3D pose of vehicles, for example, with a linear motion model in [Petrovskaya and Thrun 2009; Danescu et al. 2009] or with constant turn models [Catalin and Nedevschi 2008; Hahn et al. 2010].

Although one can observe an increase in publications that utilize a particle filter, the drawback of this filter is that it is non-deterministic and, depending on the problem, computationally much more complex than the Kalman filter, even if capabilities for parallel computing are exploited. In the case of a linear Gaussian system, the particle filter can never yield a better result than the Kalman filter. However, even the sub-optimal extended Kalman filter, which will play a key role in this thesis, yields very promising results in many practical situations at much lower computational costs.

As a conclusion of this review on vehicle tracking systems, to the knowledge of the author, there is no literature, beside the own publications, explicitly addressing motion state estimation of oncoming traffic, including the yaw rate, from a moving platform with a stereo vision sensor yet. Most approaches are designed to work in highly controlled environments, e.g., on highways, and cannot be transferred without larger adaptation to other scenarios. However, with respect to driver assistance and safety systems the knowledge of where an oncoming vehicle will be the next second is highly advantageous, e.g., for collision avoidance in particular on country roads and at intersections.

1.3. Thesis Contributions

In this thesis, a novel real-time approach for estimating trajectories of road vehicles such as cars, vans, or motorbikes, from a moving platform based on stereo image sequences is presented. A *trajectory* consists of a sequence of object pose and motion states. The focus of this contribution will be on oncoming traffic, while most existing work in the literature addresses tracking the lead vehicle.

The overall approach is generic and scalable to a variety of traffic scenes including inner city, country road, and highway scenarios. A considerable part of this thesis addresses oncoming traffic at urban intersections.

The estimated parameters include the 3D position and orientation of an object, relative to the ego-vehicle, as well as the object's shape, boundaries (dimension), velocity, acceleration and yaw rate, e.g., rotational velocity.

The present work combines the advantages of a feature-based object representation and a geometric 3D model. Dense stereo depth maps are efficiently integrated, allowing for more accurate reconstruction of the object boundaries. With respect to real-time demands, the problem of estimating the objects shape and size is separated from estimating the pose and motion parameters.

The *first main contribution* of this dissertation includes the derivation of the object model, motion model, and measurement model of this tracking approach, which enables estimating the yaw rate of other traffic participants reliably from a moving platform using a vision sensor.

The *second main contribution* addresses vehicle tracking a highly dynamic turn maneuvers. Different methods for dealing with the large dynamic range at intersections are proposed and compared, including a multi-filter setup.

Further contributions are two alternative initialization methods, robust outlier handling strategies, a probabilistic approach for assigning new points to a tracked object, or an object verification method.

Author publications: The contents of this thesis have been partly published in the following articles and conference proceedings:

A. Barth, J. Siegemund, A. Meissner, U. Franke, and W. Förstner [2010]. "Probabilistic Multi-Class Scene Flow Segmentation for Traffic Scenes". In: *DAGM Symposium on Pattern Recognition*. LNCS 6376, pp. 513–522

A. Barth and U. Franke [2010]. "Tracking Oncoming and Turning Vehicles at Intersections". In: *Intelligent Transportation Systems, IEEE Conference on*. Madeira Island, Portugal, pp. 861–868

A. Barth, D. Pfeiffer, and U. Franke [Nov. 2009b]. "Vehicle Tracking at Urban Intersections Using Dense Stereo". In: *3rd Workshop on Behaviour Monitoring and Interpretation, BMI*. Ghent, Belgium, pp. 47–58

D. Pfeiffer, A. Barth, and U. Franke [2009]. “Robust and Precise 3D-Modeling of Traffic Scenes based on Dense Stereo Vision”. In: *6th Workshop Fahrerassistenzsysteme.* Löwenstein/ Hößlinsülz, Germany, pp. 11–20

A. Barth, J. Siegemund, U. Franke, and W. Förstner [2009a]. “Simultaneous Estimation of Pose and Motion at Highly Dynamic Turn Maneuvers”. In: *DAGM Symposium on Pattern Recognition*, pp. 262–271

A. Barth and U. Franke [2009]. “Estimating the Driving State of Oncoming Vehicles From a Moving Platform Using Stereo Vision”. In: *Intelligent Transportation Systems, IEEE Transactions on* 10.4, pp. 560–571. issn: 1524-9050

C. Hermes, A. Barth, C. Wöhler, and F. Kummert [2009b]. “Object Motion Analysis and Prediction in Stereo Image Sequences”. In: *Proc. Oldenburger 3D-Tage*, pp. 172–182

A. Barth and U. Franke [2008]. “Where Will the Oncoming Vehicle be the Next Second?” In: *Intelligent Vehicles Symposium, IEEE*, pp. 1068–1073

U. Franke, C. Rabe, S. Gehrig, H. Badino, and A. Barth [2008]. “Dynamic stereo vision for intersection assistance”. In: *FISITA World Automotive Congress, VDI-FVT*

Patent Application:
Inventors: Alexander Barth, Dr. Uwe Franke
Patent-No.: DE 10 2008 025 773 A1
Filing Date: 29.05.2008, Publication Date: 08.01.2009
Title: Method for Estimation of the Pose and Motion State of an Observed Object (German title: Verfahren zur Schätzung eines Orts- und Bewegungszustands eines beobachteten Objekts)

1.4. Problem Statement

This section gives a formal description on the problem at hand to be investigated throughout this thesis. It proposes a general model that will be successively concretized in later sections.

The objective is to derive relevant properties of a rigid object $\boldsymbol{O}$, moving within a three-dimensional unknown environment, from a sequence of stereo images and other sensor inputs. Each rigid object has specific, time-invariant properties (e.g. size, shape, color, weight,...) as well as time-dependent properties, such as the relative position and orientation to another object, a particular configuration of the wheels in case of vehicles, or a given velocity. Among all possible object properties, the motion parameters as well as the object boundaries are of particular interest with respect to driver assistance systems.

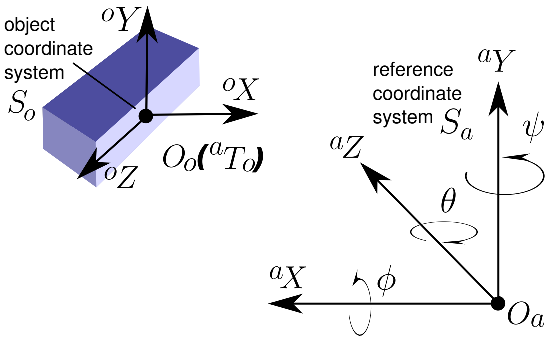


Figure 1.1.: The object pose $\mathbf{\Omega}$ with respect to a reference system S_a is given by the rotation and translation of a tripod, corresponding to a local object coordinate system. The rotation is described, e.g., by the three Euler angles, i.e., pitch angle ϕ, yaw angle ψ, and roll angle θ. The translation corresponds to the coordinates of the object origin in the reference frame.

There are properties that are directly observable by the available sensors, others are hidden and have to be derived based on proper functional models. For example, the object movements, following a certain physical model, are continuous in time. However, they are observed at discrete time steps and have to be reconstructed accordingly from the given *snapshots*. This requires a proper mathematical model, approximating the complex object dynamics.

There exist multiple representations for a given property, e.g., a surrounding cuboid can be described by the coordinates of its corners, or by definition of a local coordinate system at one reference corner in combination with the specification of the length of the cuboid sides. The concrete representation must not influence the general computational theory of the problem at hand. The same holds for the algorithmic realization estimating the unknown object properties. This organization into *computational theory* and *algorithmic realization* is motivated by Marr [Marr 1982]. According to this definition one can distinguish between a general model specifying *what* is the problem and *why*, and a particular approach, that solves a given problem in a specific way (*how*).

Formalization

A generic rigid object description at discrete time k[1] is given by a number of pose parameters $\mathbf{\Omega}$, a motion model f with parameters $\mathbf{\Phi}$, and a set of object specific, time-invariant parameters $\mathbf{\Theta}$ as

$$\boldsymbol{O}(k) \models \{\,^a\mathbf{\Omega}(k), \mathbf{\Phi}(k), \mathbf{\Theta}\}\,. \tag{1.1}$$

Pose Parameters: The object pose parameters define the position and orientation of a local orthonormal object coordinate system, relative to a given reference system S_a, e.g., the camera coordinate system or an $\underline{a}$rbitrary static world system (see Fig. 1.1).

The position is given by a 3D translation vector, $^a\boldsymbol{T}_o = [\,^aX_o, \,^aY_o, \,^aZ_o]^\mathsf{T}$, between the two coordinate system origins, and a 3×3 rotation matrix, $^a\mathsf{R}_o(\boldsymbol{p}_{\mathrm{rot}})$ with rotation

[1] Throughout this thesis k is used for discrete time steps while t indicates continuous time.

parameters $\boldsymbol{p}_{\mathrm{rot}}$. At this point it does not matter, what concrete rotation representation is chosen, for example, Euler angles, a rotation around one arbitrary axis, or quaternions (see, e.g., [McGlone 2004]). The notation $\boldsymbol{O}_o({}^a\boldsymbol{T}_o)$ expresses that the object origin, with name $\boldsymbol{O}_o$, *is represented by* the translation vector ${}^a\boldsymbol{T}_o$. The pose parameters at time k can thus be summarized to

$$
{}^a\boldsymbol{\Omega}(k) = \left[{}^aX_o(k),\ {}^aY_o(k),\ {}^aZ_o(k), \boldsymbol{p}_{\mathrm{rot}}(k)\right]^\mathsf{T}. \tag{1.2}
$$

The object pose parameters define a coordinate transformation of object coordinates to the given reference system. Let ${}^o\boldsymbol{P} = \left[{}^oX,\ {}^oY,\ {}^oZ\right]^\mathsf{T}$ denote the 3D position of a given point with name $\mathcal{P}$ in the object system S_o, i.e., $\mathcal{P}({}^o\boldsymbol{P})$, and ${}^o\mathbf{P} = \left[{}^oX,\ {}^oY,\ {}^oZ, 1\right]^\mathsf{T}$ the same point in homogeneous representation. Then, the 4×4 matrix ${}^a\mathsf{W}_o(k)$ transforms this point ${}^o\mathbf{P}$ from object coordinates to the reference frame S_a at one discrete time step k, i.e., ${}^a\mathbf{P}(k) = {}^a\mathsf{W}_o(k)\,{}^o\mathbf{P}(k)$, with

$$
{}^a\mathsf{W}_o(k) = \left[\begin{array}{cc} {}^aR_o(\boldsymbol{p}_{\mathrm{rot}}(k)) & {}^a\boldsymbol{T}_o(k) \\ \mathbf{0}_3^\mathsf{T} & 1 \end{array}\right]. \tag{1.3}
$$

It is assumed that there exists a functional model h that relates the pose parameters ${}^a\boldsymbol{\Omega}(k)$ to a number of observations, summarized in vector $\boldsymbol{z}(k)$, with

$$
\boldsymbol{z}(k) = h({}^a\boldsymbol{\Omega}(k), \boldsymbol{\Theta}). \tag{1.4}
$$

Depending on the actual model, the time-independent parameters $\boldsymbol{\Theta}$ can also contribute to this so called *measurement model*. In this contribution, the measurements mainly consist of 3D points obtained from a stereo vision system.

Motion Parameters: It is assumed that there exists a functional model f that fully describes the continuous motion of a given object. For real object movements, this model is typically unknown and has to be approximated, e.g., by a set of differential equations and parameters. Examples for different motion models will be given in Sec. 2.5.

In a time-discrete system, the model f relates the pose and motion parameters between two consecutive time steps $k-1$ and k as

$$
\left[{}^a\boldsymbol{\Omega}(k), \boldsymbol{\Phi}(k)\right]^\mathsf{T} = f\left({}^a\boldsymbol{\Omega}(k-1), \boldsymbol{\Phi}(k-1)\right). \tag{1.5}
$$

The resulting object pose ${}^a\mathsf{W}_o(k)$ is related to the previous pose ${}^a\mathsf{W}_o(k-1)$ by a *motion matrix* ${}^a\mathsf{M}(k-1 \rightarrow k)$ as

$$
{}^a\mathsf{W}_o(k) = {}^a\mathsf{M}(k-1 \rightarrow k)\,{}^a\mathsf{W}_o(k-1). \tag{1.6}
$$

with

$$
{}^a\mathsf{M}(k-1 \rightarrow k) = \left[\begin{array}{cc} {}^aR(\boldsymbol{q}_{\mathrm{rot}}(k-1 \rightarrow k)) & {}^a\boldsymbol{T}(k-1 \rightarrow k) \\ \mathbf{0}_3^\mathsf{T} & 1 \end{array}\right]. \tag{1.7}
$$

The rotational parameters $\boldsymbol{q}_{rot}$ and the translation vector ${}^a\boldsymbol{T}$ depend on the motion model f, the motion parameters $\boldsymbol{\Phi}(k)$, for example, the object velocity or acceleration in a certain direction, and the discrete time interval Δt between time step $k-1$ and k.

Throughout this thesis, **coordinate transformations** at one time step k will be denoted as ${}^{b}\mathsf{W}_{a}(k)$, indicating a transformation from S_a to S_b in homogeneous coordinates. The inverse transformation from S_b to S_a is defined as ${}^{b}\mathsf{W}_{a}^{-1}(k) = {}^{a}\mathsf{W}_{b}(k)$.

Motions, i.e., point transformations within one reference frame S_a from one time step $k-1$ to the next discrete time step k are denoted as ${}^{a}\mathsf{M}(k-1 \rightarrow k)$. Equivalently, the abbreviation ${}^{a}\mathsf{M}(k)$ is used.

Time-invariant parameters: The parameter vector $\boldsymbol{\Theta}$ contains a number of time-invariant parameters, that mainly depend on the given application. The object dimension or the coordinates of object points within the object coordinate system are examples for time-invariant parameters.

Objective: Based on the definitions above, the following objective can be summarized:

> Given a sequence of, to some amount uncertain, observations $\boldsymbol{z}(1), \ldots, \boldsymbol{z}(K)$, estimate the unknown parameters $\boldsymbol{\Omega}(k)$, $\boldsymbol{\Phi}(k)$, and $\boldsymbol{\Theta}$, which are constrained by the measurement model h and the object motion model f in (1.4) and (1.5) respectively.

For this particular task one has to define a proper motion model f, that well-describes the transformation of the observed object poses. In addition, an algorithmic approach for estimating the unknown parameters based on a number of uncertain observations, that are related to the unknown parameters through a mathematical model h is required. Then, different state estimation techniques can be applied to solve for the unknown parameters, such as maximum likelihood estimation or Kalman filtering, which will be presented in Sec. 2.6.

1.5. Organization of the Thesis

The remainder of this thesis will be organized as follows. Chapter 2 gives a comprehensive introduction to the technical background that is required for the proposed vehicle tracking approach. This includes a brief introduction on the used sensor inputs, as well as the fundamental concepts of image formation, stereo vision, and motion estimation from image sequences. The chapter further proposes different motion models used for vehicle tracking, and introduces the theory and notation of state estimation techniques to be applied in later sections.

The actual vehicle tracking approach is presented in Chapter 3. It contains details on the used object model, the measurement model, and the stochastical model, as well as practical issues such as initialization, outlier detection, and object verification. In addition, three extensions of the basic approach are presented, allowing for tracking of vehicles at highly dynamic turn maneuvers.

The proposed system is systematically evaluated in Chapter 4 both on simulated and real world data. An outlook on future research as well as the conclusions of this contribution are given in Chapter 5.

2. Technical Background

The proposed vehicle tracking approach requires accurate knowledge both about depth and motion in the scene. This chapter first briefly introduces the different sensors used in the remainder, and then focuses on how depth and motion information can be derived from stereo image sequences. This includes a comprehensive overview on existing methods in the literature.

The second objective of this chapter is to introduce the notation, parameters and concepts of the image formation process, different object motion models, as well as state estimation techniques, that will be applied in later sections.

Since back references are provided, this background chapter might be skipped at first reading.

2.1. Sensors and Sensor Data

There are a variety of different sensors providing information on the ego-vehicle and the environment. For a good overview on these sensors see, for example, [Färber 2004]. Here, only the sensors used in this thesis are briefly introduced, namely a stereo vision sensor, a far-range radar sensor, as well as inertial sensors. These sensors are assumed to have the following properties:

The **stereo vision sensor** consists of two cameras, mounted behind the windshield of the ego-vehicle, that capture synchronized sequences of intensity images. It is assumed that the sensor further provides stereo *disparity maps*, from which distance information can be derived (see Sec. 2.3 for details).

The **radar sensor** provides a *target list*, where each target is represented by a 2D point (lateral and longitudinal position with respect to the ego-vehicle) as well as the relative velocity of that point target in direction of the radar beam.

Inertial sensors provide information on the current driving state that can be used, for example, to predict the motion of the ego-vehicle (*ego-motion*). In this thesis, the following information is utilized:

- ego-vehicle speed
- ego-vehicle yaw rate
- ego-vehicle GPS-position

The GPS (global positioning system)-position allows to reference the current position of the ego-vehicle within a global coordinate system.

Details on the vision and radar sensor, including the advantages and drawbacks, are summarized below.

	Stereo Camera	Radar Sensor
Output	stereoscopic intensity image sequence, dense stereo disparity maps	point target list + relative velocities
Further processing	motion estimation/ feature tracking (see Sec. 2.4)	-
Max. distance	≈ 50 m	200 m
Field of View (FOV)	42°	17°
Advantages	large FOV, good angular resolution (pixel/degree), enables reconstruction of 3D motion field	velocity accuracy, distance range, all weather-capable
Drawbacks	stereo uncertainty increases quadratically with distance, sensitive to illumination and weather conditions (heavy rain, fog)	narrow FOV, poor lateral resolution

As can be seen, both sensors show almost orthogonal advantages and drawbacks, which makes them well-suited for sensor fusion. The stereo vision sensor provides a very accurate angular resolution in contrast to the radar sensor, which, on the opposite, yields very precise range measurements even at large distances, whereas the range uncertainty of the vision sensor increases quadratically with distance.

Note that the actual maximum distance or field of view may vary between different sensors. The given values are taken from the actual demonstrator car configuration used for the experimental results (cf. Sec. 4.4.1).

2.2. Geometric Image Formation and Camera Models

Cameras project the three-dimensional world onto a two-dimensional *image plane*. This section introduces the fundamental geometric relationships and equations of the image formation process to be used in later chapters. More detailed information can be found in many books or textbooks on Computer Vision, for example [Faugeras 1993; Trucco and Verri 1998; Forsyth and Ponce 2003; Hartley and Zisserman 2003].

2.2.1. Finite Perspective Camera

The finite perspective camera model is a specialization of the ideal perspective camera or *pinhole* camera model [Hartley and Zisserman 2003]. It is based on the fact that the image plane of real cameras is finite and, for digital cameras, consists of a discrete sensor grid.

Throughout this thesis, the pixel coordinate system S_p is defined as a right-handed system at the bottom-left corner of the image, with x corresponding to the horizontal axis, pointing from left to right, and y the vertical axis, pointing from bottom to top (cf. Fig. 2.1). Instead of referring to pixel coordinates as $({}^px, {}^py)$ based on the common

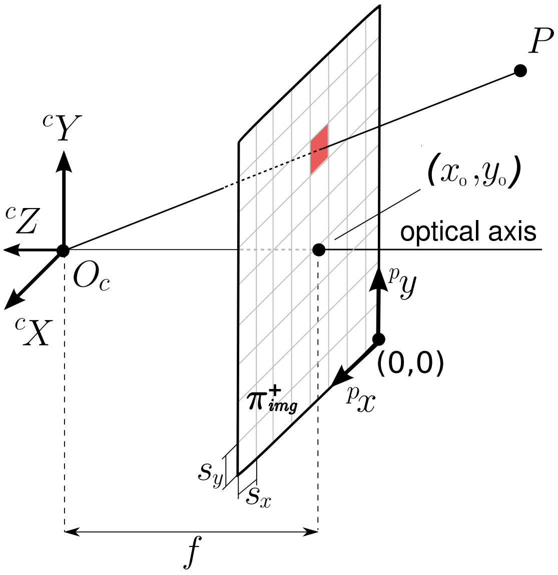


Figure 2.1.: Ideal perspective camera model for a finite discrete image plane. The pixel coordinate system is located at the bottom-left corner of the image.

notation, the short form (u, v) will be used for a pixel coordinate in the following. Alternatively, the vector notation $\boldsymbol{p} = [u, v]^\mathsf{T}$ is used. The term *image coordinate system* will be used synonymously for the pixel coordinate system, if not other stated.

The pixel coordinates (u, v) of the 3D point ${}^c\boldsymbol{P} = [{}^cX, {}^cY, {}^cZ]^\mathsf{T}$ in camera coordinates can be computed as

$$u = x/z \tag{2.1}$$

$$v = y/z \tag{2.2}$$

with

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f_x & f s_\alpha & x_0 & 0 \\ 0 & f_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^cX \\ {}^cY \\ {}^cZ \\ 1 \end{bmatrix} \tag{2.3}$$

where (x_0, y_0) denotes pixel position of the principal point, i.e., the intersection of the optical axis with the image plane, and $f_x = f/s_x$ and $f_y = f/s_y$ denote the principal distance f, scaled by the effective pixels size s_x in horizontal and s_y in vertical direction, respectively. The skewness parameter $s_\alpha = \tan\alpha$ allows for shearing the pixel coordinate system to compensate for manufacturing inaccuracies of the sensor grid. A shear angle $\alpha \neq 0$ corresponds to non perpendicular coordinate axis. However, with modern cameras, the skewness parameter typically can be neglected, i.e., $s_\alpha = 0$.

2.2.2. General Projective Camera

So far an ideal pinhole or ideal thin lens camera model has been assumed. In practice, more sophisticated models for the optical system have to be applied, to allow for a realistic modeling of the image formation process.

The thin lens model does not consider several aberrations that come with real lenses. This includes lens distortions, defocussing of rays that are neither parallel nor go through the focus (spherical aberration), different refraction based on the wavelength or color of light rays entering the lens (chromatic aberration), or focusing of objects at different depths.

Ideally, a 3D point ${}^{c}\boldsymbol{P}$, the corresponding image point $\boldsymbol{p}$ and the optical center $\boldsymbol{O}_c$ are collinear, and straight world lines are imaged as straight lines [Hartley and Zisserman 2003]. For real cameras this model does not hold. Especially at the image boundaries, straight lines appear curved (radially distorted).

To compensate for projection errors, additional intrinsic parameters, for example lens distortion coefficients, have to be introduced. This leads to a general non-projective mapping [McGlone 2004]. If the parameters of the image distortions are known (for example from camera calibration), it is possible to compensate for this errors, yielding a line-preserving perspective image representation of the world. This step is referred to as *rectification* in the literature.

2.2.3. Camera Calibration

Calibrating monocular cameras mainly involves the estimation of the intrinsic camera parameters. In addition, it is possible to compute the extrinsic parameters with respect to a given reference frame.

> Throughout this thesis, **calibrated cameras** are assumed, i.e., the intrinsic and extrinsic camera parameters are known, independent of the actual calibration method. All input images are **rectified** to straight-line-preserving images based on the intrinsic parameters in a way that the ideal perspective camera model can be applied to these images.

There exist a large number of camera calibration techniques. Many of these originate from the field of photogrammetry and have been proposed in the 60s and 70s. Based on the collinearity constraint of an ideal perspective camera and a set of control points on a known calibration object, the unknown intrinsic camera parameters are derived. A minimum number of five control points is required to solve for the principal point and the principal distance [McGlone 2004].

A common and often cited computer vision approach for camera calibration has been proposed by Tsai [Tsai 1986; Tsai 1987]. This model includes the intrinsic parameters principal distance, principal point, pixel size ratio, radial distortion coefficients and center, as well as the extrinsic parameters. It can be seen as the base model which has been extended and modified by many authors, for example [Heikkila and Silven 1997; Zhang 2000].

A well-known tool for camera calibration (an implementation of the Zhang method [Zhang 2000]) has been proposed by Jean-Yves Bouguet [Bouguet 2007]. Here, a planar calibration rig in form of a chessboard is used. This toolbox can also be used for stereo calibration.

Other than the extrinsic parameters, the intrinsic parameters ideally are camera specific constants that can be estimated once and could be provided, for example, by the camera manufacturer. However, in practice, the intrinsic parameters are sensitive to changes in temperature and other external influences. Long term studies and experiments show a drift of the intrinsic camera parameters. Especially in the automotive sector, where cars (and cameras) are exposed to varying climatic conditions and temperatures, reaching from −50°C to 50°C, online calibration techniques become important. A survey on camera self-calibration can be found, for example, in [Hemayed 2003].

2.3. Stereo Vision

Stereoscopic vision or short *stereo vision* refers to the ability of inferring information on the 3D structure of a scene from two or more images taken from different viewpoints [Trucco and Verri 1998]. Humans perceive depth based on this principle. Scene irradiance reaches the retina of the left and right eye from different angles, leading to two slightly different images. The human brain is able to find correspondences between both images and interprets the *disparities*, i.e., the displacement of corresponding image points on the retina, as a measure of scene distance [Goldstein 2001].

It is possible to transfer the idea of human stereo vision to the field of computer vision. The fundamental geometric concepts and ideas for an ideal stereo system are briefly summarized in the following sections. A detailed introduction as well as an overview on the general so called *epipolar* geometry can be found, for example, in [Hartley and Zisserman 2003].

2.3.1. Ideal Stereo Configuration

An ideal stereo configuration is shown in Fig. 2.2(a). The image planes of both cameras are parallel and displaced only by a translational component in a single direction, typically the X-axis. As a result, corresponding points can be found within the same image row.

The displacement vector of the image coordinate in the left and right image reduces to a scalar, the *disparity* d, which indicates the displacement of the horizontal image coordinate u_l and u_r, with

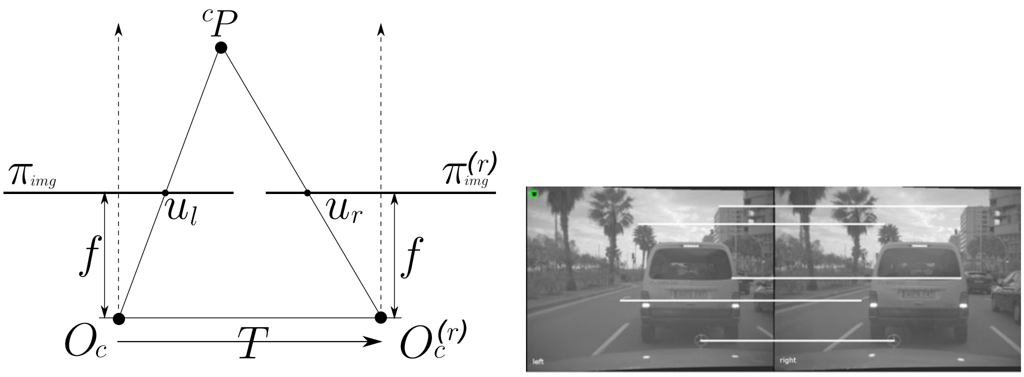
$$d = u_l - u_r. \tag{2.4}$$

If the cameras are displaced both by a known translation and rotation, it is possible to compute an ideal stereo configuration by warping both images. This procedure is called *stereo rectification*. An example can be found in Fig. 2.2(b).

The disparity is a nonlinear function of distance:

$$d = \frac{f_x \, b}{{}^{c}Z}. \tag{2.5}$$

This equation will be required in later sections for definition of the measurement model.



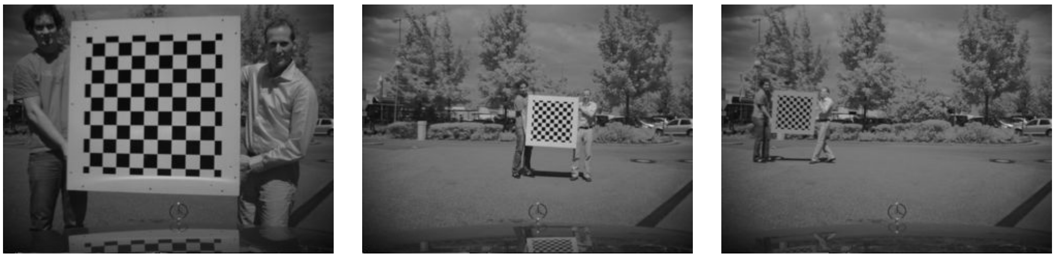
(a)

(b)

Figure 2.2.: (a) Ideal stereo configuration. The image planes are aligned in a way that they are transformed by a translation in a single direction only. (b) Example of rectified image pair, in which the epipolar lines correspond to the image rows.

2.3.2. Stereo Calibration

Stereo calibration mainly involves estimating the transformation between both cameras, i.e., the extrinsic parameters of the stereo system. The intrinsic camera parameters can be determined independently for both cameras (cf. Sec. 2.2.3) or within one step together with the extrinsic parameters. The former is related to estimating the essential matrix, while the latter corresponds to an estimation of the fundamental matrix.



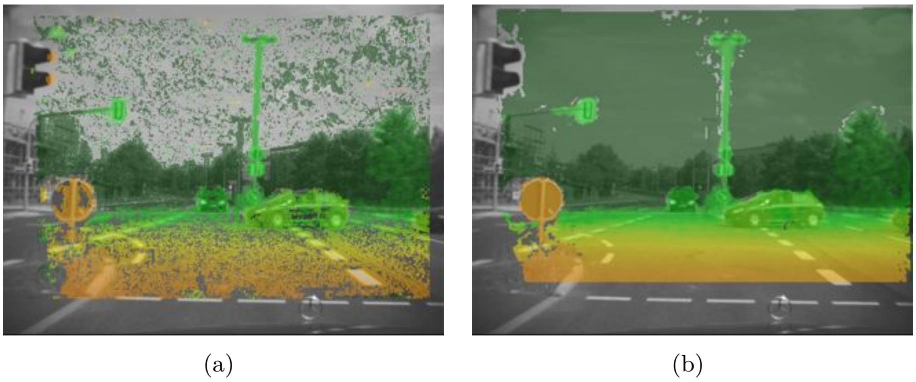
(a)

(b)

(c)

Figure 2.3.: Example images used for calibrating the intrinsic and extrinsic parameters of the stereo system. The checkerboard is captured at several poses and distances all over the viewing field of the cameras.

We use a stereo calibration algorithm that is similar to the one proposed by Bouguet [Bouguet 2007]. It requires a chessboard calibration rig of known size to be captured at different positions and poses within the field of view of the camera. Example calibration images are shown in Fig. 2.3.



(a) (b)

Figure 2.4.: Example disparity map computed by (a) block matching (ZSSD) and (b) SGM. The color encodes the disparity (red=large, green=small).

2.3.3. Review on Stereo Algorithms

Stereo vision algorithms have to solve the correspondence problem between points in the left and right image, originated by the same physical point in space. This problem is not trivial in general, since correspondences can be ambiguous or not observable. There are several surveys on stereo vision algorithms, for example [Lane and Thacker 1996; Brown et al. 2003]. A very comprehensive one up to the year 2003 is given by Brown et al [Brown et al. 2003]. Following this article, stereo algorithms can be categorized into local and global methods.

Local methods include block or feature matching, i.e., local image statistics are matched along the epipolar lines based on a similarity or distance measure. Common measures are, for example, cross-correlation, (zero-mean) sum of squared differences, (Z)SSD, (zero-mean) sum of absolute distances, (Z)SAD, or signature-based distance metrics such as census transform [Zinner et al. 2008]. Such algorithms typically work on rectified images, to reduce image lookup calls and cash misses while traversing the epipolar line. Local matching algorithms result in sparse, but quite robust disparity maps. Sparse data is obtained, since no minimum can be found on, e.g., texture-less areas. Hence, all local methods either work only on data with sufficient structure or perform post-processing steps. Fig. 2.4(a) shows an example disparity map computed with a local block matching algorithm (ZSSD). The color encodes the disparity: Red corresponds to larger disparities originated from near objects, while dark green corresponds to small disparities indicating points at far distance. This color encoding will be used throughout this thesis.

Global methods aim to compute dense disparity maps, assuming a pixel-wise smooth 3D world. These algorithms take into consideration not only local statistics, but also constraints defined over larger regions or the whole image. Such constraints are, for example, smoothness or ordering constraints. The common task is to optimize a global cost function using techniques such as dynamic programming [Ohta and Kanade 1985], graph cuts [Hong and Chen 2004], belief propagation [Yu et al. 2007], or nonlinear diffusion [Scharstein and Szeliski 1998].

The optimality comes at a price of a significant increase in computation time and memory requirements, compared to local methods. Van der Mark and Gavrila [Mark and Gavrila 2006] provide a survey on different stereo algorithms with focus on automotive applications. Due to limited computational capacities and memory resources, early automotive stereo systems have been restricted to quite sparse disparity maps. Block matching methods have been used, for example, in [Gennery 197; Saneyoshi 1994; Franke and Joos 2000]. Franke and Kutzbach present a signature-based real-time stereo algorithm in [Franke and Kutzbach 1996].

In 2005, Hirschmüller [Hirschmüller 2005] proposed a stereo matching algorithm denoted as *Semi-Global Matching* (SGM) approach, utilizing global constraints only along a finite set of scanlines, yielding dense disparity maps at a significant speed up in computation time. Further details can be found in [Hirschmüller 2008]. Recently, this approach has been extended by different matching cost functions, for example, using the census transform [Hirschmüller and Gehrig 2009], to improve the robustness of the algorithm with respect to errors in the camera calibration.

An example of a disparity map computed by the SGM algorithm is shown in Fig. 2.4(b). The SGM algorithm can be ported to dedicated hardware such as field-programmable gate arrays (FPGA). A real-time FPGA-based solution, as proposed in [Gehrig et al. 2009], is available in the demonstrator car. Although the algorithms proposed in later sections of this thesis are designed to work with arbitrary disparity maps, the SGM algorithms has been the practical basis for all experiments if not otherwise stated.

Since 2002, Scharstein and Szeliski [Scharstein and Szeliski 2002] provide a worldwide platform for evaluation of stereo algorithms at their website[1]. The test set has been continuously extended during this time. The group of Klette at Auckland University, New Zealand, has set up a testbed[2] for evaluation of stereo algorithms in automotive environments, i.e., traffic scenes, in cooperation with Daimler AG. Several articles on stereo evaluation based on this data base have been published, for example, [Vaudrey et al. 2008; Morales and Klette 2009].

2.3.4. Stixel World

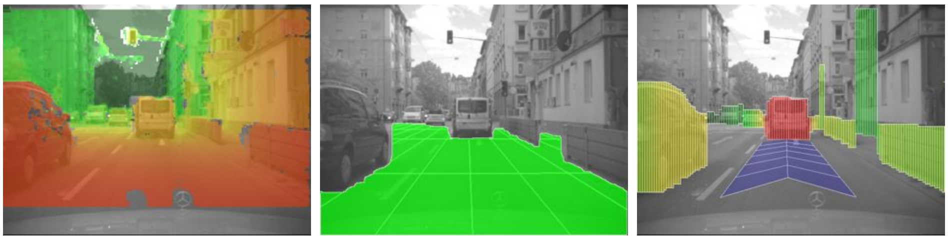
Real-time implementations of dense stereo algorithms provide significantly more information on the 3D environment compared to sparse stereo methods. The gain in information and precision allows for improved scene reconstruction and object modeling. However, more information means there is also more data to process.

The *Stixel World*, as proposed in [Badino et al. 2009], is a much more comprehensive representation of the dense stereo data, containing both information on freespace and obstacles over ground. Instead of evaluating a large number of stereo disparities, only a few *stixels* have to be considered, that locally integrate the stereo information. Each stixel, standing vertically on the ground plane, represents the limit of the drivable freespace at a certain viewing angle from the perspective of the camera. It further contains information on the distance and height of the obstacle that limits the freespace at this position. See Fig. 2.5 for an example.

In Sec. 3.5.4, the stixel representation will be utilized to derive measurements for the object pose and dimension.

[1] http://vision.middlebury.edu/stereo/

[2] http://www.mi.auckland.ac.nz/EISATS



(a) Dense disparity image (b) Free space (c) Stixel representation

Figure 2.5.: Stixel World example. (a) Input SGM disparity map (red=close, green=far). Note that SGM yields measurements even for most pixels on the road. (b) Drivable freespace based on the disparity map. (c) Resulting Stixel World representation comprising the information of (a) and (b).

2.4. Vision-based Motion Estimation

Motion is one of the most dominant monocular visual cues humans and many animals use to perceive the environment. Motion triggers attention and groups information. According to the Gestalt psychologists law of *common fate*, stimulus elements are likely to be perceived as a unit if they move together [Goldstein 2001]. These properties are also of interest in computer vision.

Estimating object motion from image sequences is directly related to estimating motion in the image plane between consecutive frames. This section summarizes the main principles of vision-based motion estimation and tracking.

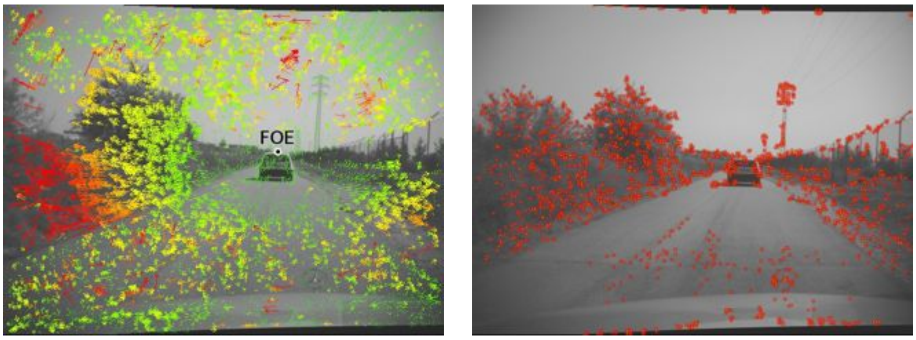
2.4.1. Optical Flow

The *optical flow* is the projection of a three-dimensional motion field onto the image plane, originating both by camera movements and independent movements of other objects in the scene (see Fig. 2.6). Analog to the correspondence problem of stereo vision (cf. Sec. 2.3), the optical flow is related to finding corresponding points in two images captured at different time steps. For introductory texts on optical flow it is referred to standard books and textbooks on computer vision, e.g. [Jähne 1995; Trucco and Verri 1998; Forsyth and Ponce 2003].

Algorithms for optical flow computation rely on the principle of *brightness constancy*, i.e., the assumption that the image intensity of a particular point at one time step does not change between two consecutive images. Formally, this can be stated as

$$I(u, v, t) = I(u + d_u, v + d_v, t + \Delta t) \tag{2.6}$$

where $I(u, v, t)$ denotes the image intensity at pixel position (u, v) at time t, and $[d_u, d_v]^\mathsf{T}$ the unknown image displacement vector on the image plane. This is the fundamental optical flow constraint. In practice, the brightness constancy assumption is often violated, e.g., due to illumination changes in the scene. However, it provides a valuable basis for a large number of optical flow algorithms. As a practical consequence, the time interval Δt between the two images has to be relatively short.



(a)

(b)

Figure 2.6.: (a) Example of the (noisy) optical flow resulting from the camera motion through a mainly stationary environment. All points in the scene beside the *focus of expansion* (FOE) are moving on the image plane. (b) Selected image features to be tracked over time to approximate the optical flow.

There is no unique solution to the optical flow problem, since there are two unknowns and only one constraint. Thus, additional constraints have to be introduced to solve the optical flow problem. Examples will be presented below. It is possible to categorize optical flow algorithms into sparse and dense reconstruction methods.

Sparse Optical Flow

Sparse methods approximate the optical flow constraint locally at a finite number of selected features. The same algorithms used for sparse stereo vision can also be applied to solve for image displacements between two consecutive frames in time, for example block matching techniques.

Classical approaches to feature selection include gray value corners, e.g. [Förstner and Gülch 1987; Harris and Stephens 1988], or small patches with high spatial frequency to overcome the well-known *aperture problem*. The idea of such interest operators is to select only image features that are likely to be retrieved in the next image. Shi and Tomasi [Shi and Tomasi 1994] have published an often cited article on *good features to track*, i.e., features selected based on optimal characteristics for the tracking algorithm used. An example of good features to track is shown in Fig. 2.6(b). The selected features are marked by a surrounding box.

The feature tracking algorithm used in [Shi and Tomasi 1994] has been developed by Tomasi and Kanade [Tomasi and Kanade 1991] and goes back to early work of Lucas and Kanade [Lucas and Kanade 1981]. The implementation is commonly known as *KLT tracker* and has become a standard method for feature tracking. However, the maximum feature displacement between two frames is limited due to processing time and reliability constraints.

If the camera is mounted in a vehicle, image displacements quickly increase as the vehicle is driving faster. Multi-resolution approaches or predictive elements, e.g. Kalman filtering, help to increase the maximum displacement in practice. Alternatively, signature based optical flow algorithms have been proposed, for example by Stein [Stein

2004]. Such methods find correspondences very efficiently using table based indexing mechanisms instead of block matching and can deal with, theoretically, arbitrary displacements on the image plane.

In the remainder of this thesis, the term **feature tracking** will be used, when a feature will be retrieved over multiple frames, while **optical flow** or **image displacement** refers to the motion between a pair of images. Thus, features tracking involves reassigning image displacements of a single feature instance over multiple frames.

Dense Optical Flow

Dense optical flow methods aim to compute a displacement vector at every position on the image plane (pixel-wise or continuously). This requires integrating some sort of smoothing or regularization to overcome the under-determined equation system.

Horn and Schunck [Horn and Schunck 1981] tackled to solve for (2.6) using the calculus of variation, with an additional smoothness constraint on the resulting vector field. Several extensions of this approach have been published, being less sensitive to outliers in the data, e.g., [Mémin and Pérez 1998], or yielding improved results at discontinuities in the flow field [Brox et al. 2004; Zach et al. 2007].

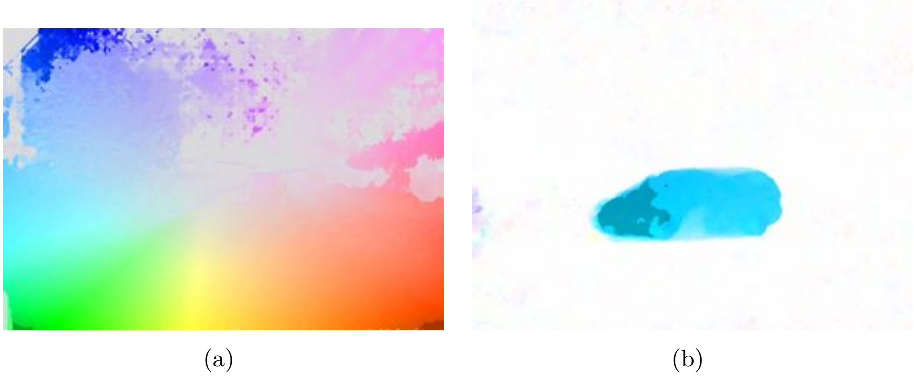

(a) (b)

Figure 2.7.: Examples of dense optical flow estimates. (a) Moving camera through a mainly stationary environment (scene as in Fig. 2.6). (a) Stationary camera with a vehicle moving from left to right. The color hue encodes the motion direction, while the saturation represents the length of a motion vector (white means stationary).

Zach et al. [Zach et al. 2007] have shown how to efficiently apply numerical energy minimization techniques for an optical energy function that contains L^1-norm terms, opposed to L^2-norm terms as widely used in other approaches. This approach runs in real-time (30 fps) on graphics card hardware for 320 × 240 images and is more robust compared to L^2-norm formulations of the energy functional.

Wedel et al. [Wedel et al. 2009a] have extended this approach by an adaptive regularization term, depending on rigid body motion and the 3D structure in the scene. At

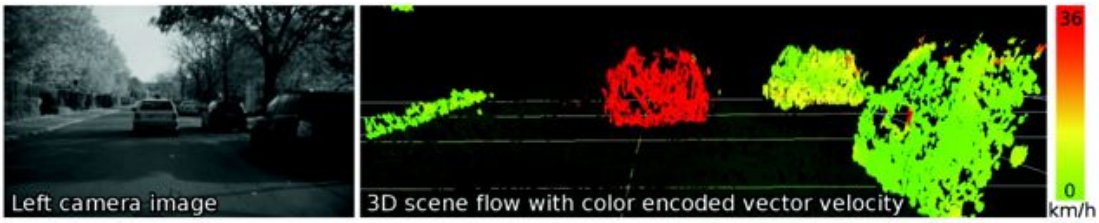


Figure 2.8.: Example scene flow result. The color encodes the absolute velocity of a given point in the world. By courtesy of A. Wedel.

the time of publication, the results of this approach have been outperforming all other methods on public test sets (Middlebury benchmark).

Dense optical flow fields provide valuable information for environment perception. However, in the field of driver assistance systems, all currently existing methods still have problems with really large image displacements as occurring if the camera is mounted in a fast driving car (e.g. highway scenario).

Due to this fact, the vehicle tracking approach proposed in later sections will be based on sparse methods.

2.4.2. Scene Flow

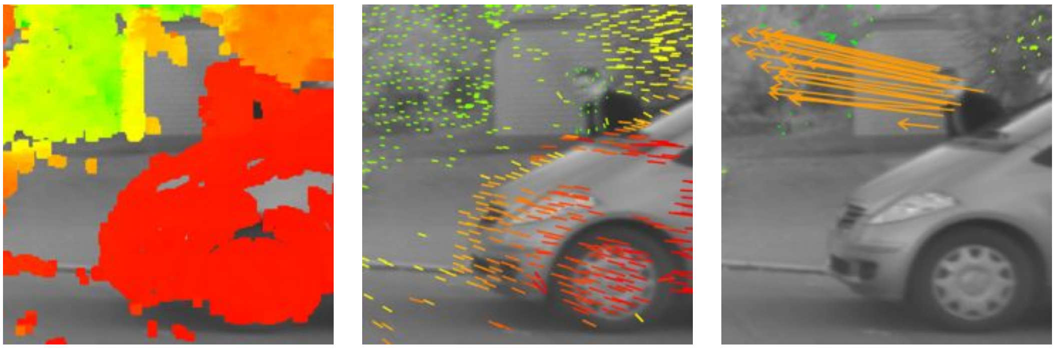
While optical flow refers to the two-dimensional motion field in the image plane, *scene flow* represents the corresponding three-dimensional motion field. Depth information from stereo system helps to solve for ambiguities in the flow field that cannot be reconstructed from monocular vision, for example, the rate of change in depth. Instead of analyzing pairs of images, i.e., two consecutive frames of the same camera, the scene flow is computed over four images (two stereo images at two consecutive time steps). Thus, the scene flow is additionally constrained.

The resulting 3D motion field represents the fundamental input to the vehicle tracking approach in this thesis and will be further the basis for object detection. As for the optical flow there exist dense and sparse scene flow variants.

Dense Scene Flow

The four-image configuration allows for a joint estimation of motion and disparity at sparse image positions [Patras et al. 1997] or for almost all visible points in the scene using variational methods as in [Vedula et al. 2005; Min and Sohn 2006; Huguet and Devernay 2007].

A real-time version of the scene flow has been proposed by Wedel et al. [Wedel et al. 2008b]. This approach first decouples the estimation of depth and motion, and then solves for the globally optimal scene flow (including dense optical flow and disparity rate) by minimization of an energy function that combines depth and motion constraints in a variational framework. The resulting reconstruction of the scene flow is reported to be about 500 times faster compared to other existing techniques, while the accuracy on test sequences, including the Middlebury benchmark, is comparable with the most accurate methods. An example result of this approach is shown in Fig. 2.8. More details



(a) stereo (b) optical flow (c) 6D-Vision

Figure 2.9.: Stereo and motion alone are not sufficient in many situations to separate an object from the background. However, fusing the information over multiple time steps, as in the 6D-Vision method, enables fast and reliable object detection.

on this approach and the background on scene flow can be found in [Wedel 2009]. A GPU scene flow implementation is described, for example, in [Rannacher 2009].

Although a significant speed up has been achieved, the dense scene flow computation is still computationally demanding, e.g., the whole available computation power is required for this task leaving not much capacities for further processing steps such as object tracking. In addition, the current dense scene flow implementations have the same difficulties with large displacements as dense optical flow methods. Thus, sparse scene flow approximations as proposed below are used in this approach to estimate vehicle motions. The integration of dense scene flow data into the vehicle tracking system is part of future work.

Sparse Scene Flow/ 6D-Vision

In the following, a sparse real-time scene flow variant, denoted as *6D-Vision*, is sketched. It has been proposed in [Franke et al. 2005] and will be used in Sec. 3.8.1 for initializing new object tracks.

In this approach, the 3D motion field is reconstructed for a finite set of tracked points in the scene (see Fig. 2.9). The points are tracked on the image plane using a feature tracker. At each time step, the corresponding 3D position follows from the stereo system. This results in a sequence of 3D point positions. A Kalman filter is used to estimate a 3D velocity vector based on the measured 3D points. Accordingly the scene flow estimate is stabilized by integrating the measured point positions over time.

For each tracked point, a state vector of the form

$$\boldsymbol{x} = \left[\underbrace{X, Y, Z}_{\text{position}}, \underbrace{v_x, v_y, v_z}_{\text{velocity}} \right]^\mathsf{T} \tag{2.7}$$

is estimated. Following from the dimensionality of the state vector, the method is referred to as *6D-Vision*. Each point is observed in terms of a measured image position (u, v) and stereo disparity d at the current time step. The nonlinear measurement model, relating these measurements to the unknown state parameters, is based on the perspective camera model (cf. Sec. 2.2).

Temporal fusion of depth and motion cues overcomes several problems of purely stereo or flow based object detection methods. For example, the pedestrian behind the parking car in Fig. 2.9 can be hardly detected in a stereo disparity map, since the measured 3D points on the pedestrian show almost the same disparity as points on the car. However, depth information helps to eliminate ambiguities in the flow field. The small feature displacements on the pedestrian, superimposed in Fig. 2.9(b), cannot be distinguished from displacements of far distant points (the color encodes the amount of displacement from green=small to red=large optical flow). The combination of depth, optical flow, and ego-motion information indicates that the points on the pedestrian are actually moving, while the car and the background are stationary. Filtering helps to stabilize the motion estimate over time. Detecting the pedestrian based on the 6D-Vision result becomes much more easy compared to the original data.

It is possible to track up to $10,000$ 6D-Vision vectors in real-time (25 fps) on GPU architectures. Given dense disparity maps and dense optical flow, it is even possible to estimate the position and velocity at almost every pixel in the image plane [Rabe et al. 2010].

While the stationary points provide useful information to reconstruct the ego-motion of the vehicle [Rabe et al. 2007], moving points are a strong indicator for objects in the scene. With the 6D-Vision approach it is possible to track the motion of objects before an object is actually being *detected* or further segmented.

As will be seen later, the 6D-Vision principle is closely related to the proposed vehicle tracking approach. The main difference is that the latter does not consider points independently. Instead the information of many points that follow the same rigid body motion is grouped. The advantage of this approach is that not only translational, but also rotational movements can be reconstructed.

2.5. Motion Models

A motion model represents a mathematical description of movements within a particular environment (cf. Sec. 1.4). It will be required in later sections to predict an object state a particular time interval ahead. As these movements can be very complex and highly non-linear, sufficient approximations have to be found in practice.

It can be distinguished between *kinematics* and *kinetics*. Kinematics describe object motion in a geometrical way without considering the actual cause of the motion, while kinetics relate movements to their physical origin in terms of forces and masses [Martin 1907]. Both kinematics and kinetics are branches of classical mechanics.

It can be further distinguished between rigid body and non-rigid body motion. An example for the latter is the movement of a human, where different body parts show different motion patterns. However, the focus of this section will be on rigid objects.

Table 2.1 gives one possible categorization of different motion models used for vehicle tracking in the literature. A first basic separation can be done with respect to the

Domain	Category	Examples
Image	Affine transformation	translation only [Rajagopalan and Chellappa 2000; Zeng and Ma 2002; Li et al. 2004; Narayana and Haverkamp 2007; Zhi-fang and Zhisheng 2007], translation and scale [Koller et al. 1994], translation, rotation, and scale [Chateau and Lapreste 2004]
3D	Generic rigid body motion	linear motion/constant velocity [Danescu et al. 2007; Franke et al. 2005], linear motion/constant acceleration [Kaempchen et al. 2004], 3D rotation and 3D translation [Siegemund 2008; Bachmann 2009]
	Vehicle motion (variants of the *bicycle model*)	constant velocity/ constant orientation [Maybank et al. 1996], constant velocity/ constant yaw rate [Koller et al. 1993; Dellaert and Thorpe 1997; Leibe et al. 2007], constantly accelerated velocity/ constant yaw rate [Zhao and Thorpe 1998; Kaempchen et al. 2004], constant velocity/ constant steering angle [Dahlkamp et al. 2004]

Table 2.1.: Categorization of different motion models used in the field of vehicle tracking.

domain in which the motion is modeled. There are several approaches that model object motion in the image plane (2D). Others describe movements in the three-dimensional Euclidean space, which can be further separated into generic rigid body movements (rotation and translation) and into vehicle specific motion models.

The following sections present the basic ideas of each motion model class.

2.5.1. Affine Motion Models

Most tracking approaches that work in the image domain rely on the idea that the displacement of an object will be small if the time interval between two consecutive images is short. They try to retrieve the translation of an image region, including a detected object, within a local neighborhood of the previous location. Examples for such *search based* approaches are [Rajagopalan and Chellappa 2000; Zeng and Ma 2002; Li et al. 2004; Narayana and Haverkamp 2007; Zhi-fang and Zhisheng 2007]. There are only a few approaches that explicitly model the translation and scaling of an object on the image plane, e.g., [Koller et al. 1994; Chateau and Lapreste 2004], or translation, scaling, and rotation [Chateau and Lapreste 2004]. All these approaches consider tracking of a leading vehicle with relative little changes between two images. Affine motion models are commonly used in the field of MPEG encoding of image sequences [Gahlot et al. 2003].

It can be benefical to track an object directly in the domain of the measurements, since the uncertainties of the sensor are usually known very accurately. Nonlinear transformations of the measurements into a different domain also affect the distribution of the measurement noise. A drawback of such approaches is that constant velocities in the image plane do not correspond to constant velocities in 3D space in general due to the nonlinear projection.

2.5.2. Generic Rigid Body Motion

The simplest rigid body motion model assumes linear motion of constant velocity [Danescu et al. 2007; Franke et al. 2005; Petrovskaya and Thrun 2009] or constant acceleration [Singer 1970; Kaempchen et al. 2004]. In most approaches, the object movements are constrained to movements on a planar ground, ignoring the height component of the movement. This is typically referred to as *ground plane assumption* or *planar world assumption*.

There are other approaches that also incorporate rotational movements, e.g., [Siegemund 2008]. A constant translational and rotational velocity model is used in [H. Badino 2008] for representing the movements of the ego-vehicle in 3D space.

Mehrotra and Mahapatra [Mehrotra and Mahapatra 1997] have proposed a system modeling higher order terms such as the jerk (change of acceleration) in the field of tracking maneuvering targets, e.g., manned aircrafts.

The generic motion models are not restricted to movements in a given direction, i.e., vehicles could also move sidewards with such models.

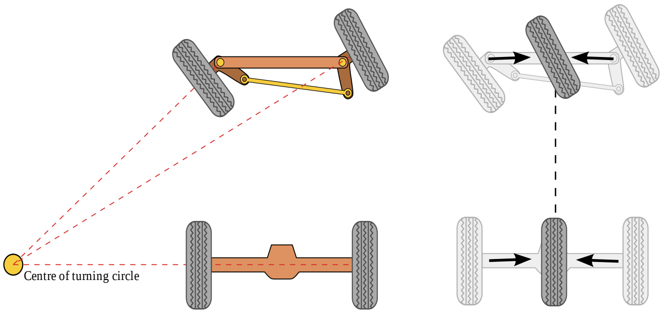


Figure 2.10.: Left: Ackermann steering geometry with four wheels. Right: The left and right wheel at one axle are merged in the bicycle model.

2.5.3. Vehicle Motion Models

In the following, motion models considering specific properties of road vehicles with four wheels and two axles are presented.

Ackermann Steering Geometry

A typical car has four wheels, where only the two wheels on the front axle can be steered (see Fig. 2.10). The configuration with two independently steerable wheels on the front axle goes back to the German wagon-maker Georg Lankensperger in 1816. It is usually denoted as *A-Steering* or *Ackermann Steering Geometry*, named after the British Rudolph Ackermann, who filed a patent for the new steering geometry in England on behalf of Lankensperger. The A-steering geometry replaced the center pivot plate steering, where the whole front axle is turned in curves while the wheels are fixated to that axle.

One important characteristic of this geometry is that the rays perpendicular to each wheel's moving direction intersect in a single point, the *turning center* of a circle. Thus, each wheel moves on a circle with different radius, but with the same center of rotation.

Two-lane models include all four wheels and allow for sophisticated consideration of kinetic parameters at circular path motion. For example, centrifugal forces and wheel loads that impact the wheels differently on the inner and outer lane throughout a curve.

Single-lane models represent a significant simplification of the two-lane model. They are commonly used to approximate the fundamental driving characteristics under normal conditions. If a car is moving on a planar dry surface (no pitch or roll movements) and the center of mass is lying on the ground, the different wheel loads or centrifugal forces between inner and outer lane can be neglected [Zomotor 1987], and the left and right wheel on each axle can be merged at the axle's center (see Fig. 2.11). The resulting two wheel model corresponds to the steering geometry of a motorbike or bicycle. It is thus usually referred to as *bicycle model* in the literature.

Bicycle Model

From a kinematic perspective, the radius R_r of the rear wheel's circular path is completely defined by the steering angle of the front wheel δ_s and the distance L between

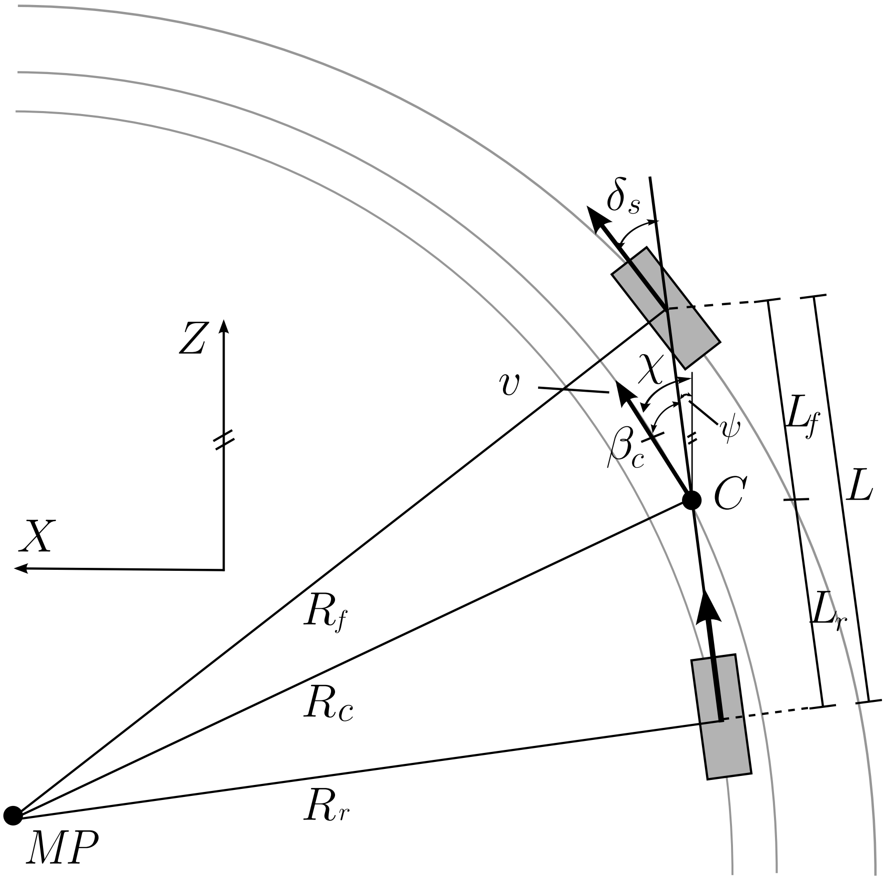


Figure 2.11.: Simple bicycle model for low velocities

the front and the rear wheel:

$$R_r = \frac{L}{\tan \delta_s}. \tag{2.8}$$

For small steering angles one yields $R_r = L/\delta_s$. Accordingly, for $\delta_s = 0$, the rear wheel moves on a circle with infinite radius, i.e., on a straight line. The radius of the front wheel also follows directly from the Ackermann geometry, i.e.

$$R_f = \sqrt{R_r^2 + L^2} \tag{2.9}$$

The radius R_C at the center of mass C is defined as

$$R_C = \frac{L_r}{\tan \beta_C} \tag{2.10}$$

where L_r indicates the distance between rear wheel and center of mass, and β_C denotes the *side slip angle*, i.e., the angle between the longitudinal axis of the vehicle and the velocity vector at C, tangential to the curvature of the circle. At small curvatures, i.e., $R_r \gg L$, the influence of the distance between front and rear wheels becomes neglectable, thus

$$R_f \approx R_C \approx R_r = R. \tag{2.11}$$

The moving direction χ with respect to a global world coordinate system is composed of the yaw angle ψ and the side slip angle β_C as

$$\chi = \psi + \beta_C \quad . \tag{2.12}$$

For small angles and very low velocities, the side slip angle can be approximated by $\beta_C = L_r/R_C$. In general, β_C depends on the velocity and the object's mass. One can show that the following relationship holds [Zomotor 1987]:

$$\beta_C = \frac{L_r}{R} - \frac{m_g}{\varrho} \cdot \frac{L_f}{L} \cdot \frac{v^2}{R} \tag{2.13}$$

where m_g represents the object's mass, L_f the distance between C and the front wheel, v the tangential velocity, and ϱ a linearized rear wheel specific coefficient. Thus, the side slip angle is a function of the centripetal acceleration $a_q = v^2/R$.

Furthermore one can express the curvature $1/R$ of the driving path in terms of the change in orientation of the moving direction as

$$\frac{1}{R} = \frac{d\chi}{ds} = \frac{d(\psi + \beta_C)}{ds} \tag{2.14}$$

where s represents the arc length. This term can be transformed as follows:

$$\frac{d(\psi + \beta_C)}{ds} = \frac{d(\psi + \beta_C)}{dt} \cdot \frac{dt}{ds} = \frac{\dot{\psi} + \dot{\beta}_C}{v} \tag{2.15}$$

with velocity $v = \frac{ds}{dt}$. Accordingly, the centripetal acceleration a_q of the circle can be expressed in terms of the yaw rate $\dot{\psi}$, side slip rate $\dot{\beta}_C$, and velocity v as

$$a_q = \frac{v^2}{R} = v(\dot{\psi} + \dot{\beta}_C). \tag{2.16}$$

A stationary movement on a circular path is reached, if $\dot{v} = \dot{\beta}_C = 0$. The in-plane translation of the center of mass with respect to a static world coordinate system results from the following observations

$$\begin{aligned} \frac{dX_C}{ds} &= \sin(\psi + \beta_C) \\ \frac{dZ_C}{ds} &= \cos(\psi + \beta_C) \end{aligned} \tag{2.17}$$

and

$$\begin{aligned} \frac{dX_C}{ds} &= \frac{dX_C}{dt} \cdot \frac{dt}{ds} = \frac{\dot{X}_C}{v} \\ \frac{dZ_C}{ds} &= \frac{dZ_C}{dt} \cdot \frac{dt}{ds} = \frac{\dot{Z}_C}{v} \end{aligned} \tag{2.18}$$

follows

$$\begin{aligned} \dot{X}_C &= v \sin(\psi + \beta_C) \\ \dot{Z}_C &= v \cos(\psi + \beta_C). \end{aligned} \tag{2.19}$$

The explicit integration of the side slip angle incorporates vehicle specific properties as derived from the A-steering geometry. At normal conditions, the side slip angle becomes negligible at the center of the rear wheel. Thus, the center rear axle is the most stable point at circular path motion. Due to (2.11) it is possible to define the driving path based on the center rear axle instead of the center of gravity. This reduces the influence of a side slip angle to a minimum and (2.19) becomes

$$\begin{aligned} \dot{X}_r &= v\sin(\psi) \\ \dot{Z}_r &= v\cos(\psi). \end{aligned} \tag{2.20}$$

Only the special case of stationary movements on a circular path with constant velocity has been considered above. Many other parameters such as acceleration, suspension, camber change, roll effects, nonlinear tire behavior, aerodynamics, etc. have been ignored. Such complex models are beyond the scope of this thesis, since most of the parameters require sophisticated knowledge about the characteristics of a particular vehicle and can hardly be estimated for an unknown vehicle just by looking at it with a vision sensor. A more detailed introduction into vehicle dynamics can be found, for example, in [Zomotor 1987; Alexander and Maddocks 1988; Mitschke 1990; Gillespie 1992].

Variants of the Bicycle Model

The bicycle model has been developed to describe the motion characteristics of a vehicle, for which several parameters, such as the distance between the front and rear axis or the vehicle mass, have to be known in advance, others must be provided by inertial sensors, e.g., the current steering angle of the front wheels. Since these parameters can hardly be estimated for another vehicle, further simplified variants of the bicycle model are commonly used in the literature for vehicle tracking.

The basis for these models are a set of differential equations, including (2.20) as well as the derivatives of the velocity and orientation. They differ in the level of complexity, i.e., the order up to which the derivatives are non-zero. Common used motion models are given in Table 2.2. Non-modeled higher-order terms are typically represented as zero-mean white Gaussian noise processes.

Further details on the different models can be found, for example, in the PhD thesis of Cramer [Cramer 2004], or in the papers of Bühren and Yang [Bühren and Yang 2007b] and Schubert et al. [Schubert et al. 2008] on vehicle motion models.

The common constant yaw rate model has the problem that it does not incorporate the correlation between velocity and yaw rate, i.e., in place rotations become possible (non-zero yaw rate even if the object has zero velocity). A solution is to replace the constant yaw rate model by a constant steering angle assumption. This is addressed, for example, in [Dahlkamp et al. 2004]. However, deriving the yaw rate from the steering angle and the velocity requires knowledge on the distance between the axles.

2.6. State Estimation

This sections introduces the basic concepts of state estimation and presents different state estimation techniques that will be used in later sections for estimating the un-

Model	Differential Equations
Constant velocity, constant orientation (CVCO)	$\dot{v} = 0$ $\dot{\psi} = 0$
Constant velocity, constant yaw rate (CVCY)	$\dot{v} = 0$ $\dot{\psi} = \text{const}$ $\ddot{\psi} = 0$
Accelerated velocity, constant yaw rate (AVCY)	$\dot{v} = \text{const}$ $\dot{\psi} = \text{const}$ $\ddot{v} = 0$ $\ddot{\psi} = 0$
Accelerated velocity, accelerated yaw rate (AVAY)	$\dot{v} = \text{const}$ $\ddot{\psi} = \text{const}$ $\ddot{v} = 0$ $\dddot{\psi} = 0$

Table 2.2.: Variants of the bicycle model. All higher order derivatives not listed in the table are zero.

known object parameters, such as pose and motion. The most essential technique in this context is the Kalman filter, which goes back to Rudolf E. Kalman in 1960 [Kalman 1960]. It has been established as standard method for a large variety of tracking applications. There exist a large number of variants and extensions, however, only those required in later sections are briefly presented below. For more details on estimation theory it is referred to the literature, e.g. [Maybeck 1979; McGlone 2004]. A very good tutorial-like introduction on Kalman filtering is given by Welch and Bishop [Welch and Bishop 2006]. The main references providing the basis for this section include the comprehensive book on optimal state estimation by Simon [Simon 2006] and the book on probabilistic robotics by Thrun et al. [Thrun et al. 2005]. The theory on multi-filter approaches is mainly originated from [Bar-Shalom et al. 2001].

2.6.1. Least Squares Estimation

The objective is to estimate a number of unknown parameters $\boldsymbol{x} \in \mathbb{R}^{\mathrm{N}}$ based on a set of (noisy) observations $\boldsymbol{z} \in \mathbb{R}^{\mathrm{M}}$, with $\boldsymbol{z} = \tilde{\boldsymbol{z}} + \boldsymbol{\epsilon}_z$, i.e., the ideal (true) observations $\tilde{\boldsymbol{z}}$ are assumed to be disturbed by white, zero-mean, uncorrelated Gaussian noise $\boldsymbol{\epsilon}_z$ with known covariance matrix $\boldsymbol{C}_{zz}$. These observations are related to the unknown parameters by a functional model (also referred to as *measurement model*). In case of a linear system, the functional model is given by the matrix $\boldsymbol{H}$, with

$$\boldsymbol{z} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{v}. \tag{2.21}$$

It is further assumed, that the additive noise term $\boldsymbol{v}$ originates only from errors in the measurements, i.e., $\boldsymbol{v} = \boldsymbol{\epsilon}_z \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}_{zz})$. This is also referred to as the *stochastical model*.

In general, one can formulate the relationship between measurements and estimated state parameters as

$$\boldsymbol{z} = h(\boldsymbol{x}, \boldsymbol{v}) \tag{2.22}$$

with $h : \mathbb{R}^{\mathrm{N}} \times \mathbb{R}^{\mathrm{N}} \rightarrow \mathbb{R}^{\mathrm{M}}$ an arbitrary (nonlinear) function with $E(\boldsymbol{z}) = h(\boldsymbol{x}, \mathbf{0})$. The unknown parameters are also modeled as random variables following a normal distribution, i.e., the state estimate is given by the mean and covariance of the parameters.

There exist a variety of approaches to solve this problem in a statistically optimal sense. The most common approach is to alter $\boldsymbol{x}$ by least-squares minimization of the residual $\boldsymbol{r}$ between the actual measurements $\boldsymbol{z}$ and the predicted measurements $\widehat{\boldsymbol{z}} = H\widehat{\boldsymbol{x}}$, i.e.

$$\boldsymbol{r}^{\mathsf{T}}\boldsymbol{r} = (\boldsymbol{z} - \widehat{\boldsymbol{z}})^{\mathsf{T}} (\boldsymbol{z} - \widehat{\boldsymbol{z}}) \rightarrow \text{min.} \tag{2.23}$$

The uncertainties of the measurements are integrated into the state estimation by minimizing the following equation:

$$\boldsymbol{r}^{\mathsf{T}} C_{zz}^{-1} \boldsymbol{r} \rightarrow \text{min.} \tag{2.24}$$

One can show that, for the linear case, the following equation yields a maximum likelihood estimate of the unknown parameters, assuming the measurement and stochastical model to be correct [McGlone 2004]:

$$\widehat{\boldsymbol{x}} = \underbrace{\left(H^{\mathsf{T}} C_{zz}^{-1} H\right)^{-1}}_{C_{\widehat{x}\widehat{x}}} H^{\mathsf{T}} C_{zz}^{-1} \boldsymbol{z} \tag{2.25}$$

with $C_{\widehat{x}\widehat{x}}$ corresponding to the covariance matrix of the maximum likelihood estimate $\widehat{\boldsymbol{x}}$. This equation requires the covariance matrix of the residuals to be nonsingular, i.e., the residuals must be corrupted by at least some noise [Simon 2006], which is acceptable in practical applications. If the measurement model is nonlinear, h has to be linearized at an expectation value for $\boldsymbol{x}$, e.g., by Taylor expansion. Then, a correction vector for the parameters is computed. This procedure is typically iterated a few times to converge to a local minimum.

2.6.2. Recursive State Estimation

The state estimation procedure presented above does assume that all measurements are available at one time to compute the maximum likelihood estimate for the unknown parameters. In many practical applications, however, the measurements are obtained sequentially over time, and it is desirable to compute a state estimate at runtime at short time intervals.

The task is to integrate new measurements recursively into the state estimation process, without recomputing the state estimate every time step based on the current measurements and all previous measurements. The idea is to define the estimation process as first order hidden Markov model [Rabiner 1989], where the current discrete state estimate $\widehat{\boldsymbol{x}}(k)$ depends only on the previous state estimate $\widehat{\boldsymbol{x}}(k-1)$, i.e.,

$$\widehat{\boldsymbol{x}}(k) = \widehat{\boldsymbol{x}}(k-1) + K(k)\left(\boldsymbol{z}(k) - H(k)\widehat{\boldsymbol{x}}(k-1)\right). \tag{2.26}$$

The *gain matrix* K steers the amount of how much the new measurements are incorporated into the current state estimate. All previous measurements $\boldsymbol{z}(0), \boldsymbol{z}(1), \ldots, \boldsymbol{z}(k-1)$ are included into the estimated parameter vector at discrete time step $k-1$. With the right choice of K, the result is still a maximum likelihood estimate (see, e.g., [Simon 2006]):

$$K(k) = C_{\widehat{xx}}(k-1) H(k)^{\mathsf{T}} \left[H(k) C_{\widehat{xx}}(k-1) H(k)^{\mathsf{T}} + C_{zz}(k) \right]^{-1} \tag{2.27}$$

where $C_{\widehat{xx}}(k-1)$ denotes the covariance matrix of the estimated state vector at time step $k-1$. It is also recursively updated as follows:

$$\begin{aligned} C_{\widehat{xx}}(k) \quad = \quad & [I - K(k)H(k)]\, C_{\widehat{xx}}(k-1)\, [I - K(k)H(k)]^{\mathsf{T}} \\ & + K(k) C_{zz}(k) K(k)^{\mathsf{T}}. \end{aligned} \tag{2.28}$$

The recursive formulation provides also a significant speed up in computation time, since less measurements have to be processed at one time step. An example of a recursive state estimator is the Kalman filter.

2.6.3. The Linear Kalman Filter

In many systems, the unknown state parameters are not time-invariant, but can be described in terms of a linear discrete-time system:

$$\boldsymbol{x}(k) = A\boldsymbol{x}(k-1) + B\boldsymbol{u}(k) + \boldsymbol{w}(k), \tag{2.29}$$

i.e., the state at time k can be predicted based on the *system matrix* A and the previous discrete-time state. The (optional) control input vector $\boldsymbol{u}$ is assumed to be known and is used to incorporate additional information into the state estimation process that is independent from the previous state. The matrix B is denoted as *control matrix*, and $\boldsymbol{w}$ represents Gaussian zero-mean white noise with covariance $C_{ww}(k)$. The measurement model is defined as in (2.21).

The Kalman filter is a recursive estimator that consists of a prediction and a correction (update) step. First, the state estimate and state covariance are predicted into the current time step using the (linear) system model. The prediction is then corrected by incorporating the current measurements based on the given measurement model.

Prediction: The expectation value of (2.29) yields an *a priori* estimate for the current state:

$$\begin{aligned} \widehat{\boldsymbol{x}}^{-}(k) \quad &= \quad A\widehat{\boldsymbol{x}}^{+}(k-1) + B\boldsymbol{u}(k) && (2.30) \\ &= \quad E\left[\boldsymbol{x}(k) | \boldsymbol{z}(1), \ldots, \boldsymbol{z}(k-1)\right] && (2.31) \end{aligned}$$

This equation depends on the previous state and the control input only. The term $\widehat{\boldsymbol{x}}^{+}(k) = E\left[\boldsymbol{x}(k) | \boldsymbol{z}(1), \ldots, \boldsymbol{z}(k)\right]$ denotes the *a posteriori* state estimate, i.e., the state estimate after incorporating the measurements up to time k.

The state covariance matrix $C_{\widehat{xx}}$ is also propagated through the linear system as

$$C^{-}_{\widehat{xx}}(k) = A C^{+}_{\widehat{xx}}(k-1) A^{\mathsf{T}} + C_{ww}(k) \tag{2.32}$$

where $C^{+}_{\widehat{xx}}(k-1)$ denotes the *a posteriori* state covariance matrix at previous time step.

Correction: In (2.26), the recursive update of a state estimate based on the measurements $\boldsymbol{z}(k)$ is given. This equation is the basis for the Kalman filter correction step:

$$\widehat{\boldsymbol{x}}^{+}(k) = \widehat{\boldsymbol{x}}^{-}(k) + \mathsf{K}(k)\left(\boldsymbol{z}(k) - \mathsf{H}(k)\widehat{\boldsymbol{x}}^{-}(k)\right) \tag{2.33}$$

The state estimate of the previous discrete time step is replaced by the *a priori* state estimate at current time step. Accordingly, the *Kalman gain matrix* is computed as follows:

$$\mathsf{K}(k) = \mathsf{C}^{-}_{\widehat{x}\widehat{x}}(k)\mathsf{H}(k)^{\mathsf{T}}\left[\mathsf{H}(k)\mathsf{C}^{-}_{\widehat{x}\widehat{x}}(k)\mathsf{H}(k)^{\mathsf{T}} + \mathsf{C}_{zz}(k)\right]^{-1} \tag{2.34}$$

The Kalman gain controls the influence of the system model and the measurements. Finally, the *a posteriori* state covariance is computed as

$$\mathsf{C}^{+}_{\widehat{x}\widehat{x}}(k) = \left[\mathsf{I} - \mathsf{K}(k)\mathsf{H}(k)\right]\mathsf{C}^{-}_{\widehat{x}\widehat{x}}(k). \tag{2.35}$$

If the system noise $\boldsymbol{v}$ and the measurement noise $\boldsymbol{w}$ are Gaussian, zero-mean, uncorrelated, and white, then the Kalman filter is the best linear estimator minimizing the squared residual error [Simon 2006].

2.6.4. The Extended Kalman Filter

There exist several modifications of the linear Kalman filter to deal also with nonlinear system or measurement models. Most real systems are (highly) nonlinear in practice. The idea is to linearize the nonlinear model at a given prior, assuming the model is sufficient linear at the linearization point. The most common approach to adapt the Kalman filter framework to nonlinear systems by linearization is known as *extended Kalman filter* (EKF). It is assumed, that the following model holds:

$$\boldsymbol{x}(k) = f\left(\boldsymbol{x}(k-1), \boldsymbol{u}(k), \boldsymbol{w}(k)\right) \quad \textit{system model} \tag{2.36}$$

$$\boldsymbol{z}(k) = h\left(\boldsymbol{x}(k), \boldsymbol{v}(k)\right) \quad \textit{measurement model} \tag{2.37}$$

with $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \mathsf{C}_{ww})$, and $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, \mathsf{C}_{vv})$ Gaussian noise terms. The *a priori* state estimate is obtained by applying the nonlinear system model f to the *a posteriori* state estimate of previous discrete time step:

$$\widehat{\boldsymbol{x}}^{-}(k) = f\left(\widehat{\boldsymbol{x}}^{+}(k-1), \boldsymbol{u}(k), \boldsymbol{0}\right). \tag{2.38}$$

The corresponding *a priori* covariance matrix is computed as follows

$$\begin{aligned}\mathsf{C}^{-}_{\widehat{x}\widehat{x}}(k) = {} & \mathsf{A}(k-1)\mathsf{C}^{+}_{\widehat{x}\widehat{x}}(k-1)\mathsf{A}(k-1)^{\mathsf{T}} \\ & + \mathsf{W}(k-1)\mathsf{C}_{ww}(k)\mathsf{W}(k-1)^{\mathsf{T}}\end{aligned} \tag{2.39}$$

where A and W denote the Jacobian matrices of f with respect to $\boldsymbol{x}$ and $\boldsymbol{w}$, respectively, i.e.,

$$\mathsf{A}(k) = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\widehat{\boldsymbol{x}}^{+}(k-1)} \tag{2.40}$$

$$\mathsf{W}(k) = \left.\frac{\partial f}{\partial \boldsymbol{w}}\right|_{\boldsymbol{x}=\widehat{\boldsymbol{x}}^{+}(k-1)}. \tag{2.41}$$

The correction step requires linearization of the measurement model at the currently best available prior for the unknown parameters. In this case that is $\widehat{\boldsymbol{x}}^-$. In the following, an iterative version of the extended Kalman filter correction step is given. The idea is as follows. After incorporating the measurement vectors, the *a posteriori* state estimate should give a better estimate of the unknown parameters. This information can be used to relinearize the measurement model at the updated state estimate to get an even better estimate. One can repeat the correction step multiple times until convergence. Without additional iterations the iterated extended Kalman filter (IEKF) equals the standard EKF.

The *a posteriori* estimate at iteration $\iota + 1$ is computed recursively as

$$\begin{aligned} \widehat{\boldsymbol{x}}^+_{\iota+1}(k) = \widehat{\boldsymbol{x}}^-(k) + K_\iota(k) \Big[& \boldsymbol{z}(k) - h(\widehat{\boldsymbol{x}}^+_\iota(k)) \\ & - H_\iota(k) \left(\widehat{\boldsymbol{x}}^-(k) - \widehat{\boldsymbol{x}}^+_\iota(k) \right) \Big] \end{aligned} \tag{2.42}$$

with $\widehat{\boldsymbol{x}}^+_0(k) = \widehat{\boldsymbol{x}}^-(k)$ and

$$K_\iota(k) = C^-_{\widetilde{x}\widetilde{x}}(k) H_\iota(k)^\mathsf{T} C^{-1}_{rr,\iota}(k) \tag{2.43}$$

$$C_{rr,\iota}(k) = H_\iota(k) C^-_{\widetilde{x}\widetilde{x}}(k) H_\iota(k)^\mathsf{T} + V_\iota(k) C_{vv}(k) V_\iota(k)^\mathsf{T} \tag{2.44}$$

$$H_\iota(k) = \left. \frac{\partial h}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x} = \widehat{\boldsymbol{x}}^+_\iota(k)} \tag{2.45}$$

$$V_\iota(k) = \left. \frac{\partial h}{\partial \boldsymbol{v}} \right|_{\boldsymbol{x} = \widehat{\boldsymbol{x}}^+_\iota(k)} . \tag{2.46}$$

Accordingly, the covariance matrix is adapted as follows

$$C^+_{xx,\iota+1}(k) = \left[I - K_\iota(k) H_\iota(k) \right] C^-_{\widetilde{x}\widetilde{x}}(k).$$

The matrix C_{rr} denotes the *residual covariance matrix* or the covariance of the Kalman filter innovation, i.e., $\boldsymbol{r} = \boldsymbol{z} - h(\boldsymbol{x})$.

One common measure to evaluate the quality or *confidence* of the current state estimate is the normalized innovation squared (NIS), defined as

$$\mathrm{NIS}(k) = \left(\boldsymbol{z}(k) - h(\widehat{\boldsymbol{x}}^-(k)) \right)^\mathsf{T} C^{-1}_{rr}(k) \left(\boldsymbol{z}(k) - h(\widehat{\boldsymbol{x}}^-(k)) \right) . \tag{2.47}$$

This measure expresses the squared Mahalanobis distance, [Mahalanobis 1936], of the residual or *innovation* between predicted and actual measurements. A small NIS indicates that the measurement model and the current state estimate, including the state covariance matrix, fit the current observations well, while larger NIS values reveal a significant violation of the model.

2.6.5. The Unscented Kalman Filter

The extended Kalman filter propagates the state covariances, i.e., the probability density function of the estimated state parameters, by linearization of the nonlinear system and measurement model. This assures that the resulting probability distributions remain Gaussian. However, the computed covariance matrices can significantly differ from the result obtained when applying the exact nonlinear transform.

Thus, an alternative formulation of the Kalman filter for highly nonlinear systems has been proposed by Julier and Uhlman in [Julier and Uhlmann 1997], that includes a different strategy to propagate the mean and covariance matrices of the parameters. The idea is to resample the probability density function from a number of individual points in state space for which the nonlinear system is directly evaluated. A minimum set of $2N + 1$ deterministic samples is required, where N is the dimension of the state vector. The samples, also referred to as *sigma points*, are derived from the currently best state estimate and covariance matrix. The so called *unscented Kalman filter* (UKF) is summarized in Appendix A.1.

In Sec. 4.2, the tracking performance of the UKF is compared to EKF variants with respect to the proposed vehicle tracking approach.

2.6.6. Maneuvering Targets

The system model of the Kalman filter allows for modeling a particular expected dynamic behavior of a tracked instance. This model is typically only an approximation of the complex dynamics an object can have. However, deviations from that model are assumed to be represented sufficiently well by Gaussian noise.

A particular problem often addressed in the literature, are maneuvering targets. A *maneuver* is defined as highly dynamic process in this context, opposed to stationary processes, e.g., straight-line motion of an object with constant velocity. A system model designed for mainly stationary processes may be too slow to follow a maneuvering target. Thus, several extensions have been proposed including input estimation [Chan et al. 1979], adaptive system noise control [Maybank et al. 1996], and multi-filter approaches [Bar-Shalom et al. 1989].

Input Estimation

Larger deviations from the system models can be incorporated via the control input vector $\boldsymbol{u}$, cf. (2.29) and (2.36).

Chan et al. [Chan et al. 1979] introduced an input estimation scheme that computes the maximum likelihood (constant) acceleration of a moving target over a sliding window of observations, using least squares estimation techniques. The estimated acceleration is used to update the constant velocity model. The drawback of this approach is that the control vector has a direct influence on the state estimate. Thus, errors in the input data directly affect the state estimate.

To overcome this problem, the Kalman filter is typically run twice. Once without control input (non-maneuvering mode), once with the estimated control input (maneuvering mode). The state estimate with control input is only selected if the estimated input is statistically significant, i.e., large enough compared to its standard deviation [Bar-Shalom et al. 2001].

Adaptive System Noise

The characteristics of the system model in a Kalman filter are mainly determined by the system noise, i.e., the covariance matrix C_{ww}. The lower the system noise level, the more the filter sticks to the system model. The idea is to adapt the system noise in a way that it is increased at maneuvers, so that the filter is more sensitive to the measurements. On the other hand, the system noise is lowered at mainly stationary processes to yield smooth estimates. An example for a continuous noise level adjustment based on the NIS measure is given in [Bar-Shalom et al. 2001], i.e., the difference between prediction and measurements weighted by its covariance, is selected as maneuver detector, cf. (2.47). It is assumed that larger deviations indicate a maneuver. To be less sensitive to outliers in the measurements, a sliding window approach can be applied, i.e., the average normalized error over a couple of previous time steps is considered. Maybank et al. [Maybank et al. 1996] have proposed a noise control mechanism with respect to vehicle tracking, that incorporates knowledge on the correlation between velocity and turn rate to be able to track turning vehicles. This idea will be addressed in more detail in Sec. 3.11.

Multi Filter Approaches

Instead of trying to cover all possible dynamics by a single system model, multi-filter approaches combine multiple system models by running different Kalman filters, e.g., with different system models, in parallel. The resulting state estimates are rated with respect to how likely a given model is at a given time step. The output state can be either the currently most probable state or a (weighted) combination of all estimates. There are several variants of multi-filter setups published (see [Bar-Shalom et al. 2001] for an overview). At this point, only the most common technique, known as *Interacting Multiple Models* (IMM), is presented, which will be required in later sections again.

A set of I filters builds the basis of the IMM framework. Each filter represents a certain *mode*, e.g. non-maneuvering or maneuvering. One IMM filter cycle consists of three main parts: First, the I *a posteriori* state estimates and covariance matrices of the previous discrete time step are *mixed* using a weighted sum (interaction step). Then, the filters are updated based on a common measurement vector (filtering step). Finally, a mode probability is computed for each filter (mode probability update step). The normalized residual, i.e., the deviation between prediction and measurements in terms of standard deviations, is used as indicator for mode likelihood. Furthermore, *a priori* mode transition probabilities must be provided. The weighting coefficients in the mixing step are derived from the mode likelihood and mode transition probability.

To prevent a combinatorial explosion (there are I^k possible filter configurations if all I filters are run with all filter outputs of the previous time step each new discrete time step), the output states of previous time step are probabilistically mixed, guaranteeing a constant number of I filter updates to perform each time step. The I *a posteriori* filter states $\widehat{\boldsymbol{x}}_j^+$ at time $k-1$ are combined into a single input state vector $\widehat{\boldsymbol{x}}_j^*$ and

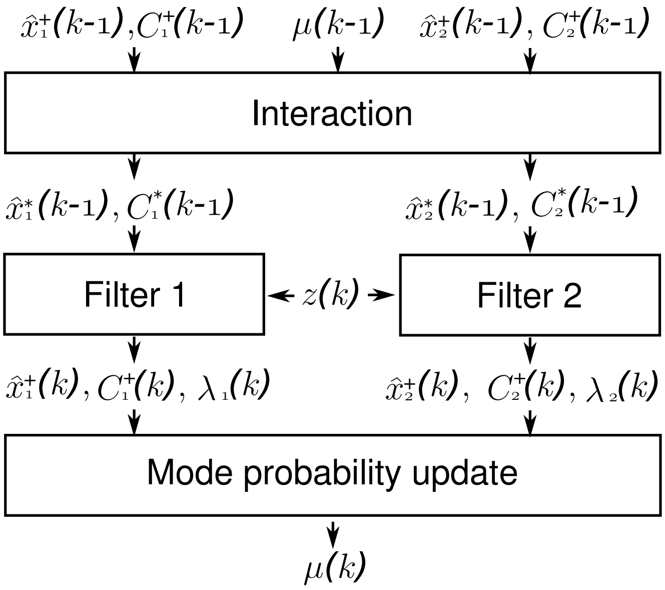

$\hat{x}_1^+(k\text{-}1), C_1^+(k\text{-}1)$
$\mu(k\text{-}1)$
$\hat{x}_2^+(k\text{-}1), C_2^+(k\text{-}1)$
Interaction
$\hat{x}_1^*(k\text{-}1), C_1^*(k\text{-}1)$
$\hat{x}_2^*(k\text{-}1), C_2^*(k\text{-}1)$
Filter 1
$z(k)$
Filter 2
$\hat{x}_1^+(k), C_1^+(k), \lambda_1(k)$
$\hat{x}_2^+(k), C_2^+(k), \lambda_2(k)$
Mode probability update
$\mu(k)$

Figure 2.12.: IMM framework for two filter setup. The two filters can have different system models, e.g. stationary or maneuvering, and are updated in parallel given the same measurements. The resulting *a posteriori* state estimates and covariance matrices are combined in the interaction step based on the mode probabilities $\boldsymbol{\mu}$, before the next filter cycle to prevent a combinatorial explosion.

covariance matrix $\mathsf{C}^*_{xx,j}$ for each filter using a weighted average:

$$\widehat{\boldsymbol{x}}^*_j(k-1) = \sum_{i=1}^{I} \widehat{\boldsymbol{x}}^+_i(k-1)\mu_{i|j}(k-1|k-1) \tag{2.48}$$

$$\mathsf{C}^*_{xx,j}(k-1) = \sum_{i=1}^{I} \mu_{i|j}(k-1|k-1)\left[\mathsf{C}^+_{xx,i}(k-1) + \right. \tag{2.49}$$
$$\left. [\widehat{\boldsymbol{x}}^+_i(k-1) - \widehat{\boldsymbol{x}}^*_j(k-1)][\widehat{\boldsymbol{x}}^+_i(k-1) - \widehat{\boldsymbol{x}}^*_j(k-1)]^\mathsf{T}\right].$$

The weighting coefficients $\mu_{i|j}(k-1|k-1)$ indicate the conditional probability that mode i has been in effect at time $k-1$, if mode j is in effect at the current time step. It is computed as follows:

$$\mu_{i|j}(k-1|k-1) = \frac{p_{ij}\mu_i(k-1|k-1)}{\widehat{\mu}_j(k|k-1)} \tag{2.50}$$

with p_{ij} the transition probability from mode i to mode j, and $\widehat{\mu}_j(k|k-1)$ the predicted probability that mode j is in effect at time k, given the mode probabilities of time $k-1$:

$$\widehat{\mu}_j(k|k-1) = \sum_{i=1}^{I} p_{ij}\mu_i(k-1|k-1). \tag{2.51}$$

The mode probability for mode j at current time step k is defined as

$$\mu_j(k|k) = \frac{\widehat{\mu}_j(k|k-1)\lambda_j(k)}{\sum_{i=1}^{I} \widehat{\mu}_i(k|k-1)\lambda_i(k)} \tag{2.52}$$

where $\lambda_j(k)$ indicates the mode likelihood based on the normalized residual error squared:

$$\lambda_j(k) = \frac{\exp\left(-\frac{1}{2}\boldsymbol{r}_j^\top \boldsymbol{C}_{rr}^{-1} \boldsymbol{r}_j\right)}{|2\pi \boldsymbol{C}_{rr}|^{\frac{1}{2}}}. \tag{2.53}$$

2.6.7. Robust Estimation

State estimation methods, as proposed above, assume mathematical models to hold true, e.g., the measured data is assumed to be disturbed by Gaussian noise only. Any deviations from that model influence the estimation result. In practice, an estimator has to be able to deal with gross errors in the data, that can occur as clear outliers or hidden contamination which usually cannot even be detected [Hampel 1973].

The term *robustness* in this context refers to the property of an estimator to produce accurate and precise results even under the presence of outliers in the data, or smaller deviations from the model assumptions. As will be seen in later chapters, this is a very important requirement in the design of the vision-based vehicle tracking system, intended to run in real world scenarios.

There exist several classes of robust estimators that are able to deal with outliers in the data, including Hough transformation [Jähne 1995], random sample consensus (RANSAC) [M. A. Fischler 1981], least median of squares [Rousseeuw 1987], or M-Estimators. The latter will be used in this approach due to the convenient way to integrate it efficiently into the Kalman filter, as will be summarized below.

M-Estimator

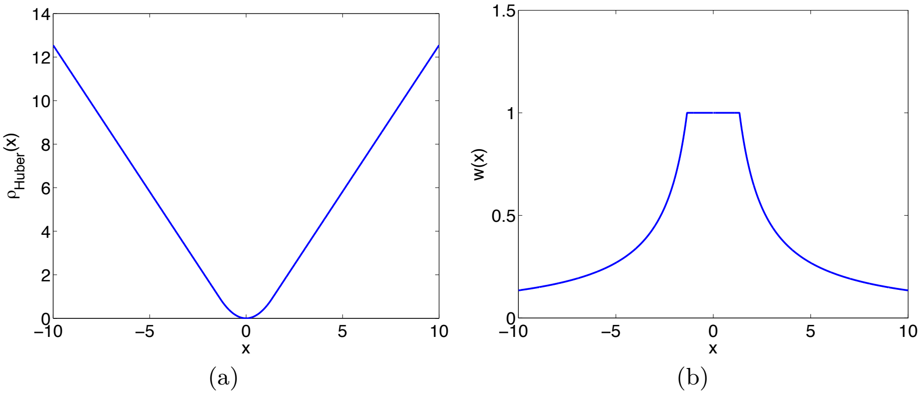
Least-squares approaches, such as the Kalman filter, minimize the squared error of the residuals $\boldsymbol{r} = \boldsymbol{z} - \widehat{\boldsymbol{z}}$ between the actual and the predicted measurements, normalized by the measurement covariance matrix $\boldsymbol{C}_{vv}$. If the M measurements are uncorrelated, i.e., $\boldsymbol{C}_{vv} = \mathrm{Diag}(\sigma_1^2, \ldots, \sigma_M^2)$, the (scaled) objective function in (2.23) becomes

$$\frac{1}{2}\sum_m \left(\frac{r_m}{\sigma_m}\right)^2 \rightarrow \min\,. \tag{2.54}$$

The rapid growth of the quadratic function has the effect that distant outliers have a strong influence, and can draw the cost minimum well away from the desired value [Hartley and Zisserman 2003]. Thus, alternative cost functions with similar properties, i.e., positive definite, symmetric functions with a unique minimum, are applied. Such functions, if chosen properly, yield a maximum-likelihood-type estimate that is close to the maximum-likelihood result, but reduce the influence of outliers significantly.

In general, (2.54), can be written as

$$\sum_m \rho(r_m^*) \rightarrow \min \tag{2.55}$$



(a)

(b)

Figure 2.13.: (a) The Huber cost function combines the L2 and L1 norm to a continuous function. (b) Corresponding weight function. Opposed to least-squares, measurements more distant from the expectation do not gain more influence on the estimation result.

with the special case $\rho(x) = \frac{1}{2}x^2$ for the least-squares method, and $r_m^* = \frac{r_m}{\sigma_m}$.

In [Zhang et al. 1997], it is shown, for example, how an arbitrary cost function can be used with only a marginal modification of the standard least-squares methods, making this approach highly applicable in practice. The quadratic form in (2.54) can be retained, if scaling (or *weighting*) the residuals by $w(x)$, i.e.,

$$\sum_m w(r_m^*) {r_m^*}^2 \to \min \tag{2.56}$$

with

$$w(x) = \frac{\rho'(x)}{x}. \tag{2.57}$$

A variety of maximum-likelihood-type cost functions have been proposed (see [Hartley and Zisserman 2003] for an overview). They can be categorized into non-convex and convex functions, where the latter are preferable due to a single, global minimum. This has the advantage of being able to descend to that minimum of the cost function independent of the initialization.

At this point, only an example is presented, which is used in later chapters. It is going back to the work of Huber [Huber 1964] on robust estimation. The so called *Huber function* is a hybrid between the L2 (least-squares) and the L1 (absolute values) cost function (see Fig. 2.13(a)). It is defined by

$$\rho_{\text{Huber}}(x) = \begin{cases} \frac{1}{2}x^2 & , \quad |x| < C_{\text{Huber}} \\ C_{\text{Huber}}(|x| - \frac{C_{\text{Huber}}}{2}) & , \quad \text{otherwise} \end{cases} . \tag{2.58}$$

where C_{Huber} is a tuning constant indicating the threshold between quadratic and linear weighting. In [Zhang et al. 1997], a value of $C_{\text{Huber}} = 1.345$ is suggested, corresponding to a 95% asymptotic efficiency, i.e., the expected precision of the estimator is 5% below the theoretic minimum precision given by the Cramer-Rao inequality (see, for example, [Kay 1993]). The first derivative of this function is

$$\rho'_{\text{Huber}}(x) = \begin{cases} x & , \quad |x| < C_{\text{Huber}} \\ C_{\text{Huber}} \operatorname{sign}(x) & , \quad \text{otherwise} \end{cases} \tag{2.59}$$

and, thus, the weighting coefficient is $w_{\text{Huber}}(x) = 1$ for residuals below C_{Huber}, and $w_{\text{Huber}}(x) = C_{\text{Huber}}/|x|$ otherwise. Since the Huber function is not bounded by definition, it is common to cut off measurements that are too far away from the expectation in practice by manually setting the weighting coefficient to 0 if the residual exceeds a user-defined threshold. This is equivalent to a n-σ-test based on the Mahalanobis distance [Zhang and Faugeras 1992], i.e., the normalized residuals are rejected as outliers, if differing more than n standard deviations from the expectation (for example $n = 3$, corresponding to the 99.7 percentile of the normal distribution).

The reweighing of measurements is easily integrated into the Kalman filter framework by scaling each row of the measurement covariance matrix by the inverse of the weighting factor of the corresponding residual, i.e.,

$$\mathsf{C}^*_{vv} = \mathsf{Q}\mathsf{C}_{vv}\mathsf{Q}^\mathsf{T} \tag{2.60}$$

with $\mathsf{Q} = \text{Diag}(w(r^*_1)^{-1}, w(r^*_2)^{-1}, \ldots, w(r^*_M)^{-1})^{\frac{1}{2}}$ and C_{vv} denoting the original measurement covariance. The resulting matrix C^*_{vv} then replaces C_{vv} in the Kalman filter equations. This yields a robust version of the Kalman filter that is less sensitive to outliers.

The techniques and models presented in this chapter are the basis for developing the vehicle tracking approach proposed in this thesis, which will be introduced in the following chapter.

3. Vehicle Tracking Approach

In Sec. 1.4 a generic problem description for tracking moving objects in a 3D environment has been introduced. Now, a specific realization for tracking of road vehicles from a moving platform based on stereo image sequences is presented. The different elements of this approach such as the object model, motion model, and measurement model are successively concretized in the following sections.

First, a compact overview on the system is given, including the given inputs, the key idea, and the requirements to make the system practicable for real world application. The overview section further motivates the organization of the remainder of this chapter.

3.1. Overview

Fig. 3.1 gives an overview on the overall vehicle tracking approach.

Inputs: The main input data consists of motion and depth information extracted from stereo image sequences. Based on this information it is possible to track 3D points over time. Inertial sensors provide information on the motion of the ego-vehicle (velocity and yaw rate) and, thus, information on the camera motion. A radar sensor additionally provides information on the velocity of other vehicles in front of the ego-vehicle.

Objective: The objective is to estimate the pose, shape, and motion parameters of an observed vehicle in 3D over time based on these sensor inputs, including its velocity, acceleration, and yaw rate.

The main requirements of this approach are generality, robustness, and real-time computation time. Opposed to many existing approaches, which are restricted to a particular task such as tracking the lead vehicle at highways, this approach should be generic in a sense that it is neither limited to a particular moving direction nor to the vehicle type and shape. All *typical* real world driving situations, including inner city traffic, intersection turn maneuvers, country road traffic, and high way traffic should be covered, with no requirement for situation dependent parameter tuning. The system must be explicitly able to deal with oncoming vehicles, which is in particular challenging due to large relative velocities and the increasing uncertainty of the stereo measurements with distance.

Key idea: The key idea of this approach originates from the fact that observed 3D points on an object's surface do not move randomly and independently of others. Instead they underlie one common rigid body motion.

As the vehicle moves, all points on the object surface move with respect to a fixed reference system, e.g., the world system, while the relative position of a given point

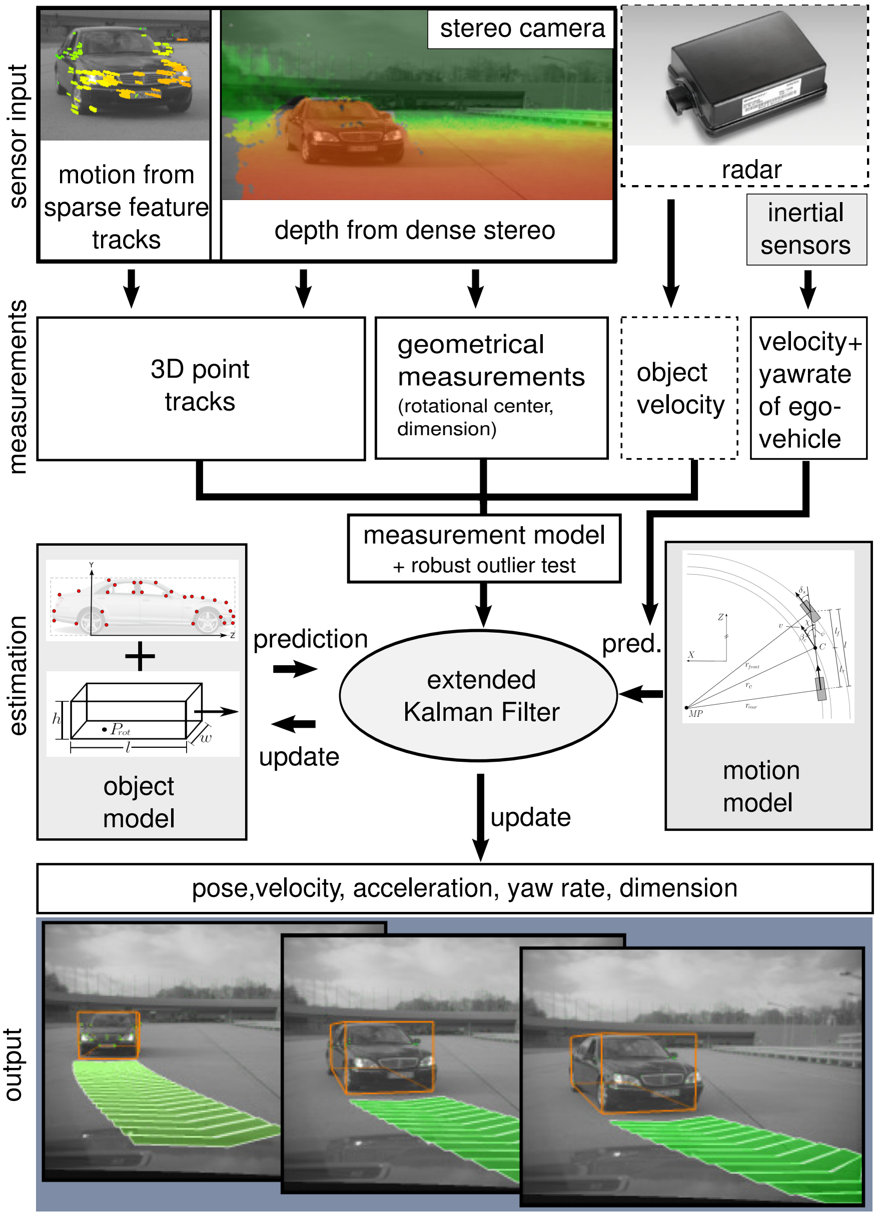

sensor input
stereo camera
motion from sparse feature tracks
depth from dense stereo
radar
inertial sensors
measurements
3D point tracks
geometrical measurements (rotational center, dimension)
object velocity
velocity+ yawrate of ego-vehicle
measurement model + robust outlier test
estimation
prediction
update
extended Kalman Filter
pred.
object model
motion model
update
pose,velocity, acceleration, yaw rate, dimension
output

Figure 3.1.: Overview on the tracking system. Depth and motion information from stereo vision is fused with a 3D object model and a vehicle motion model within an extended Kalman filter approach to estimate the pose and motion parameters of road vehicles. The framework can be easily extended by other sensor inputs, for example, radar velocity measurements.

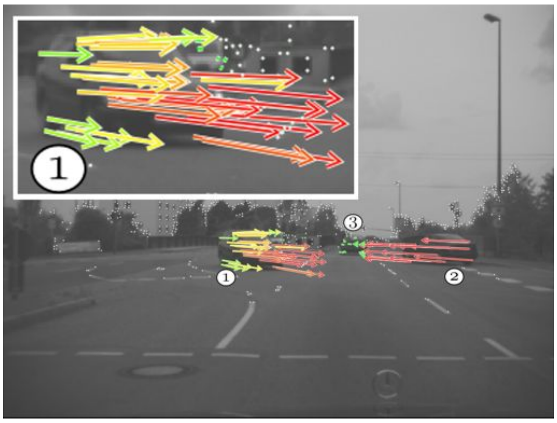


Figure 3.2.: 3D motion of tracked object points. The colors encode the lateral velocity (red fast, green slow). Points at the front of vehicle no. 1 show larger lateral velocities compared to points at the rear as the vehicle is turning; demonstrating the data contains valuable information on the rotational velocity.

with respect to a local object coordinate system does not change over time. The point movements in the world can be observed in terms of image displacements and depth changes in a sequence of stereo images.

Now the task is to solve for the inverse problem. Given a number of image displacements and distance measurements of tracked object points (=sparse *scene flow*, cf. Sec. 2.4.2), find the corresponding object movement in 3D that best matches these observations. For this purpose, an extended Kalman filter is utilized.

The advantage of considering a group of points is that not only translational, but also rotational movements become observable. This is illustrated in Fig. 3.2. Here, the measured lateral velocity of points at the vehicle's front is larger compared to points at the rear, correctly indicating a rotation of the turning vehicle. A 3D point cloud is further a very generic representation of an object, and thus can be used to represent the shape of arbitrary vehicle types, even if partly occluded.

Robustness is mainly obtained by two properties. First, the more 3D points are contributing (redundancy), the less influence get random measurement errors on the estimation result implicitly. In a data association step, new points are successively added to an existing object model at runtime to ensure that there are always sufficient points, since in practice feature tracks might be lost.

Secondly, movements of the point cloud are restricted by a vehicle motion model. The motion model incorporates a prediction on the expected, physically possible vehicle movement. Measured 3D point displacements that differ significantly from the expectation are detected and sorted out from the model. The influence of the remaining points on the estimation result is controlled using the robust Kalman filter approach as proposed in Sec. 2.6.7.

The radar sensor is optional in the system design, i.e., the tracking approach yields also very good results if only the data from the stereo vision sensor is available.

Requirements and Practical Issues: There are several requirements to make the core system, which is based on 3D points displacements only, applicable in practice. First, the 3D points belonging to a given object have to be segmented from the remaining points.

Secondly, a time-consistent coordinate system has to be attached to each object. The transformation between this object coordinate system and the camera coordinate system must be known for one step in time, as well as the position of each point in the object system, up to random errors with known covariance matrix.

The vehicle motion model further assumes that the rotational center is physically located at the center rear axle of the corresponding vehicle. As will be seen later, this position is essential for a reliable prediction of rotational movements and for estimating the yaw rate correctly. However, this position is only weakly observable from the 3D point cloud motion. Additional measurements based on dense stereo data enable to stabilize the rotation point geometrically and to reconstruct the object dimension.

The vehicle tracking approach will be successively introduced in the following sections.

First, the coordinate system definitions are summarized in Sec. 3.2. Then, the object representation and the unknown parameters to be estimated are presented in Sec. 3.3, following the structure of Sec. 1.4.

In the consecutive sections, the different components required for defining the Kalman filter are presented, i.e., the system model in Sec. 3.4, the measurement model in Sec. 3.5, and the stochastic model in Sec. 3.6.

The detection of error-prone measurements that significantly differ from the stochastic model is addressed in Sec. 3.7. Then, two different object detection and state initialization methods are proposed in Sec. 3.8. The probabilistic data association of new points to an existing model is presented in Sec. 3.9. Once an object track has been initialized, there must also be a way to terminate it, for example, if an object leaves the visual field of the camera or the estimated parameters are implausible. This is addressed in Sec. 3.10. Strategies for tracking maneuvering vehicles, e.g., at urban intersections, are given in Sec. 3.11. The chapter concludes with a brief summary in Sec. 3.12.

3.2. Coordinate Systems

In this approach, four different 3D coordinate systems are used, including camera system, ego-vehicle system, object system, and an *object-vehicle* system, respectively. Each coordinate system corresponds to an orthogonal reference frame, or tripod. All systems are right-handed. The Y-axis corresponds to the upwards pointing height axis for all systems. An overview on the different coordinate systems and transformations is given in Fig. 3.3. In addition, the two-dimensional image coordinate system of the left camera is used, following the definition in Sec. 2.2.1.

Ego-Vehicle System S_e Coordinate system attached to the ego-vehicle. The origin $\boldsymbol{O}_e$ is located at the center rear axle on the ground. The ${}^eZ$-axis is aligned to the longitudinal axis of the vehicle and points towards the scene in positive direction.

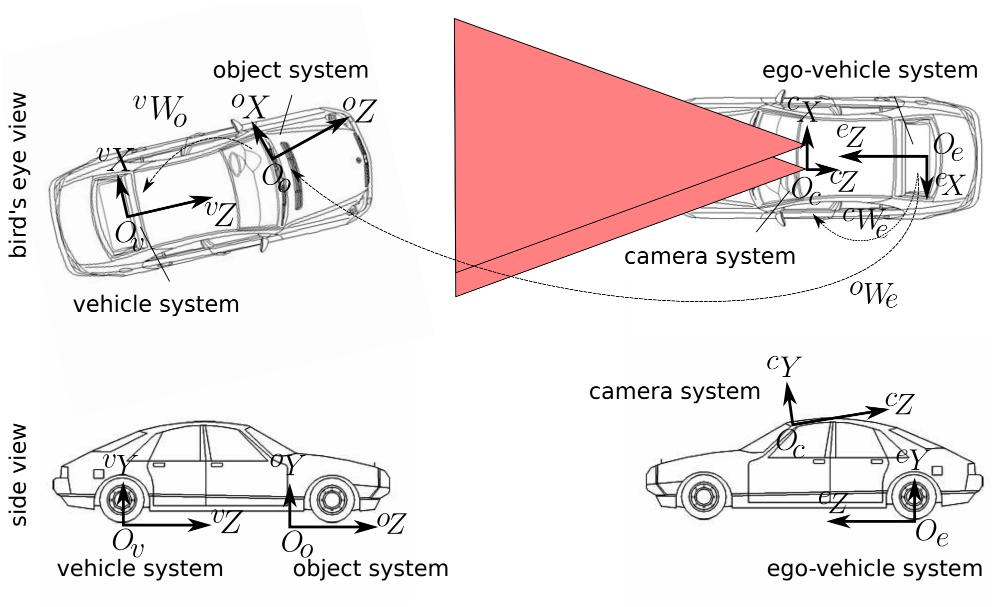


Figure 3.3.: Overview on different coordinate systems and transformations.

The ^eY-axis corresponds to the ground surface normal at the origin. The ego-vehicle system, or short *ego-system*, is attached to a moving platform, thus, it can change its position relative to a static world system over time. Points in the ego-vehicle system are denoted as $^e\boldsymbol{P}$.

Camera System S_c Located at the left camera with the axis defined according to Sec. 2.2.1. Its translation and rotation with respect to the ego-system is assumed to be known a priori from a calibration step. Points in the camera system are denoted as $^c\boldsymbol{P}$.

Object System S_o Coordinate system attached to a given observed rigid object. The origin $\boldsymbol{O}_o$ is located at theoretically arbitrary position on the object, e.g., the centroid of the object point cloud. The orientation of the ^oY-axis is defined as the ground surface normal at the origin. Points in the object system are denoted as $^o\boldsymbol{P}$.

Object-Vehicle System S_v Vehicle specific object coordinate system defined analog to the ego-vehicle system with the origin $\boldsymbol{O}_v$ located at the center rear axle of a given object on the ground. The ^vZ-axis points in the moving direction of the observed vehicle, the ^vY-axis corresponds to the ground surface normal. This definition of the object-vehicle system, which is simply referred to as *vehicle system* in the following, is required, since movements of the rigid body are related to the motion of the center rear axle (cf. Sec. 2.5.3). The pose of the vehicle system with respect to the ego-vehicle system is of particular interest, as it directly represents the pose of the given vehicle to the ego-vehicle.

The definition of two coordinate systems for each observed object is benefical for the following reason. The pose of the vehicle system with respect to the ego-system, which is important for correctly predicting the object motion, is typically not known at the beginning of a new object track and has to be estimated from the observed object motion or/and geometrical properties of the measured 3D points. At this, the object system provides a time and gauge consistent reference frame to which both the 3D points building the object shape model and the vehicle system will be referred to.

3D points can be transformed from one to another coordinate system based on the coordinate transformation matrices ${}^{c}W_e$, ${}^{e}W_o$, and ${}^{o}W_v$, respectively. With these transformation and the inverse transformations it is possible to perform the following coordinate transformation chain for a given point:

$$ {}^{c}\boldsymbol{P} \leftrightarrow {}^{e}\boldsymbol{P} \leftrightarrow {}^{o}\boldsymbol{P} \leftrightarrow {}^{v}\boldsymbol{P} \tag{3.1} $$

These coordinate transformation matrices will be specified in more detail throughout this chapter. Based on the stereo projection equations and the intrinsic and extrinsic parameters of the stereo system, it is further possible to transform a point ${}^{c}\boldsymbol{P}$ in camera coordinates into a pixel position (u, v) and stereo disparity d and vice versa.

3.3. Object Model

In Sec. 1.4, a generic rigid body object model has been defined including pose and motion parameters as well as time-invariant object properties. In the following, a concrete realization of this model for road vehicles is proposed and the different parameters of this model are described.

3.3.1. Pose Parameters

In this approach, the position and orientation of an object is defined with respect to the ego-vehicle system as illustrated in Fig. 3.4. It is fully defined by six parameters. However, since the pitch and roll angle depend mainly on the road surface, these parameters are not included in the vehicle model used throughout this thesis. Only rotations around the height axis by angle ${}^{e}\psi_o$ are modeled, as this is the only rotational parameter that can be controlled by the driver using the steering wheel. In the following, the angle ${}^{e}\psi_o$ will be referred to as yaw angle ψ for simplicity, as it always describes the rotation of a point from the object to the ego-system. This yields the reduced pose parameter vector

$$ {}^{e}\boldsymbol{\Omega} \models [{}^{e}X_o, {}^{e}Y_o, {}^{e}Z_o, \psi]^{\mathsf{T}} \tag{3.2} $$

where ${}^{e}\boldsymbol{T}_o = [{}^{e}X_o, {}^{e}Y_o, {}^{e}Z_o]^{\mathsf{T}}$ defines the translation vector from the object to the ego origin. Alternatively stated, it corresponds to the object origin in ego-coordinates ${}^{e}\boldsymbol{O}_o$.

By definition, the object origin is assumed to lie on the ground. Thus, the relative height ${}^{e}Y_o$ in ego-coordinates depends only on the given road geometry and does not contain additional information on the object. Several methods exist that estimate a vertical road model based on stereo vision, e.g. [Wedel et al. 2008a; Loose and Franke 2010; Siegemund et al. 2010]. With such methods it is possible to derive ${}^{e}Y_o$ as a function of ${}^{e}X_o$ and ${}^{e}Z_o$. Alternatively, the road is often approximated as

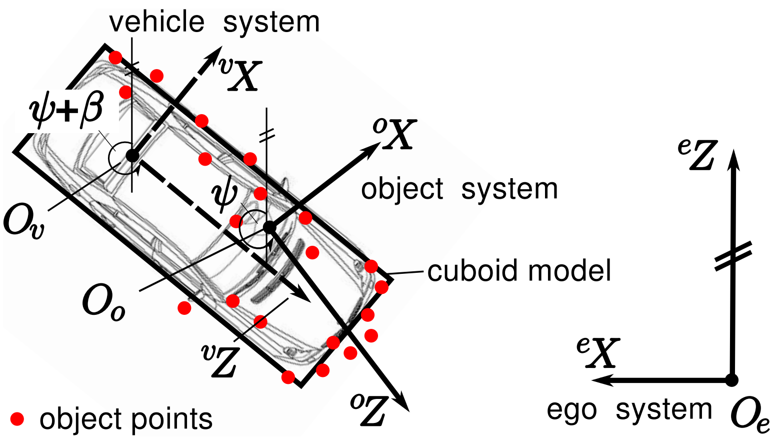


Figure 3.4.: Object model and pose parameters. The object system S_o is defined relative to the ego-system S_e based on a rotation about the height axis by angle ψ as well as translation ${}^e\boldsymbol{T}_o = {}^e\boldsymbol{O}_o$. The vehicle system S_v is defined relative to the object system by rotation β and translation ${}^o\boldsymbol{T}_v = {}^o\boldsymbol{O}_v$.

planar ground, yielding ${}^eY_o = 0$ independent of the position on the plane. This is an acceptable assumption at many road scenes and is used in the further derivation of the object tracking procedure for simplicity.

The parameter vector of the rotation, as introduced in (1.2), has only one element, i.e., $\boldsymbol{p}_{\text{rot}} = \psi$. As long as rotations are restricted to rotations about the height axis, and since the expected rotations between consecutive images are small, the rotational representation by Euler angles is a sufficient and efficient representation, which is commonly used in the automotive field.

Finally, the transformation from object coordinates into ego-coordinates for a 3D point ${}^o\boldsymbol{P}$ is given by

$$\begin{aligned} {}^e\mathbf{P} &= {}^e\mathsf{W}_o\, {}^o\mathbf{P} \\ &= \begin{bmatrix} R_y(\psi) & {}^e\boldsymbol{T}_o \\ \mathbf{0}_3^\top & 1 \end{bmatrix} {}^o\mathbf{P} \end{aligned} \tag{3.3}$$

in homogeneous representation. The 3×3 matrix $R_y(\psi)$ represents a counter-clockwise rotation by ψ around the height axis with

$$R_y(\psi) = \begin{bmatrix} \cos(\psi) & 0 & \sin(\psi) \\ 0 & 1 & 0 \\ -\sin(\psi) & 0 & \cos(\psi) \end{bmatrix}. \tag{3.4}$$

3.3.2. Motion Parameters

Object movements are modeled by the vehicle motion model, as proposed in Sec. 2.5.3. The motion parameters include the velocity v in moving direction, the longitudinal acceleration $\dot{v}$ and the yaw rate $\dot{\psi}$, as well as the pose parameters of the vehicle system to which all motion parameters are related. The coordinate system will thus be skipped in the notation of v, $\dot{v}$, and $\dot{\psi}$.

The total parameter vector of the motion model is

$$
{}^{o}\boldsymbol{\Phi} \models \left[v, \dot{v}, \dot{\psi}, {}^{o}X_v, {}^{o}Z_v, \beta\right]^{\mathsf{T}} \tag{3.5}
$$

where ${}^{o}\boldsymbol{T}_v = [{}^{o}X_v, 0, {}^{o}Z_v]^{\mathsf{T}}$ defines the origin of the vehicle system in object coordinates. It is geometrically located at the position of the center rear axle of the observed vehicle, also denoted as *rotation point*. The rotation of a point from vehicle coordinates to object coordinates about the height axis is given by the angle ${}^{o}\beta_v$. This angle will be also referred to as β for simplicity in the following. The moving direction of the vehicle in ego coordinates is $\chi = \psi + \beta$. If the object ${}^{o}Z$-axis is ideally aligned with the longitudinal axis of the vehicle, β equals the side slip angle.

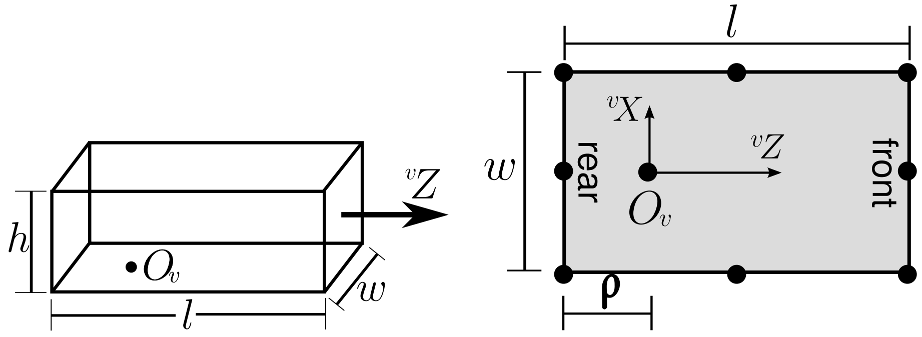
The transformation of a 3D point ${}^{v}\boldsymbol{P}$ in vehicle coordinates into object coordinates is given by

$$
\begin{aligned}
{}^{o}\mathbf{P} &= {}^{o}\mathsf{W}_v\, {}^{v}\mathbf{P} \\
&= \begin{bmatrix} \mathsf{R}_y(\beta) & {}^{o}\boldsymbol{T}_v \\ \mathbf{0} & 1 \end{bmatrix} {}^{v}\mathbf{P}.
\end{aligned} \tag{3.6}
$$

3.3.3. Shape Parameters

In this approach, the time-invariant parameters define the object shape and dimension. The shape parameters correspond to the actual point positions within the object coordinate system as well as the parameters of the surrounding cuboid.

The advantage of the point cloud representation is that a wide range of vehicles, e.g. cars, vans, motorbikes, etc., can be described sufficiently well without any a priori knowledge on the actual shape. The point cloud does not have to cover all parts of a vehicle to be able to estimate the motion parameters. This makes it very flexible and allows for dealing with partial occlusions.



(a) Perspective View

(b) Bird's Eye View

Figure 3.5.: Geometric box model with the rotation point at the center rear axis located at a constant distance ρ from the vehicle's rear. The object corners as well as the side centers have a fixed position with respect to the rotation point describable in terms of w, h, l, and ρ.

The cuboid allows for describing the object dimension and boundaries independently of the currently observed point cloud, which does not have to cover the whole object. This will be addressed in more detail in Sec. 3.5.4.

Now, the shape parameters are summarized as

$$\begin{aligned} {}^{o}\boldsymbol{\Theta} &\models \left[{}^{o}\boldsymbol{\Theta}_{\text{points}}^{\mathsf{T}},\ {}^{o}\boldsymbol{\Theta}_{\text{cube}}^{\mathsf{T}}\right]^{\mathsf{T}} \\ &= \Big[\underbrace{{}^{o}X_1,\ {}^{o}Y_1,\ {}^{o}Z_1, \ldots,\ {}^{o}X_M,\ {}^{o}Y_M,\ {}^{o}Z_M}_{{}^{o}\boldsymbol{\Theta}_{\text{points}}^{\mathsf{T}}},\underbrace{w, l, h, \rho}_{{}^{o}\boldsymbol{\Theta}_{\text{cube}}^{\mathsf{T}}}\Big]^{\mathsf{T}} . \end{aligned} \tag{3.7}$$

Each object point, named $\mathcal{P}_m$, has a time-invariant position ${}^{o}\boldsymbol{P}_m = [\,{}^{o}X_m,\ {}^{o}Y_m,\ {}^{o}Z_m]^{\mathsf{T}}$ within the object coordinate system ($1 \leq m \leq M$), i.e., $\mathcal{P}_m({}^{o}\boldsymbol{P}_m)$. The cuboid dimension is given by width w, length l, and height h, respectively. The object dimension of the cuboid is also assumed to be constant over time. The axis of the cuboid are aligned with the vehicle coordinate system (see Fig. 3.5). The corners of the cuboid have a semantical meaning, e.g., front bottom left or center bottom right, and can be fully described in terms of the vehicle dimension and a constant offset ρ in ${}^{v}Z$-direction between the center rear axle and the vehicle's rear side.

3.3.4. Filter State Representation

In this section, the object model proposed above is mapped into a Kalman filter state representation. Not all parameters are included in the state vector. There are also parameters that are estimated outside the filter as will be motivated below. Two basic models have been investigated. A full model for estimating pose, motion, and shape parameters simultaneously within a single Kalman filter state vector, and a reduced state vector that contains only the pose and motion parameters while the shape model is updated outside the filter for real-time applicability.

Full Model

The following Kalman filter state vector defines the *full model*:

$$\boldsymbol{x}_{\text{full}} = \Big[\underbrace{{}^{e}X_o,\ {}^{e}Z_o, \psi}_{\text{pose}}, \underbrace{v, \dot{v}, \dot{\psi},\ {}^{o}X_v,\ {}^{o}Z_v, \beta}_{\text{motion}},\ \underbrace{{}^{o}\boldsymbol{\Theta}_{\text{points}}^{\mathsf{T}}}_{\text{shape}}\Big]^{\mathsf{T}} \tag{3.8}$$

where the vector ${}^{o}\boldsymbol{\Theta}_{\text{points}}$ contains the object point coordinates as proposed in (3.7).

The dimensional parameters of the cuboid, ${}^{o}\boldsymbol{\Theta}_{\text{cube}}$, are not part of the state vector and have to be estimated separately. If the object point cloud represents the object completely, the cuboid is derived from the point cloud's bounding box. In practice, the point cloud usually covers only the visible parts of an object. Thus, other methods have to be applied as will be proposed in Sec. 3.5.4..

External Point Update Model

The full model estimates the point cloud model representing the object's shape simultaneously with the pose and motion parameters in an optimal sense. However, with an expected number of 50-300 points for one object, the filter state becomes extremely large.

The large filter state significantly increases the overall computation time and leads to numerical problems if the number of points gets too large. In addition, a varying number of points leads to a changing state vector size, which also alters the covariance matrix size. Accordingly, a dynamic memory handling is required, producing a significant overhead compared to constant size Kalman filter implementations.

For real-time applicability, the problem of shape reconstruction is *decoupled* from the problem of pose and motion estimation. Soatto et al. [Soatto et al. 1994] have motivated this, in the context of estimating the motion of the camera from a number of image feature correspondences between consecutive time steps, by the fact that the dynamics of the (static) points are significantly lower compared to the camera motion.

The points in the object model have a fixed position due to the rigid body assumption, i.e., no dynamics at all, while the object pose with respect to the ego-vehicle can change dynamically.

The reduced filter state is given by

$$\boldsymbol{x}_{\text{red}} = \left[\underbrace{{}^eX_o,\ {}^eZ_o, \psi}_{\text{pose}}, \underbrace{v, \dot{v}, \dot{\psi},\ {}^oX_v,\ {}^oZ_v, \beta}_{\text{motion}} \right]^{\mathsf{T}} . \tag{3.9}$$

This formulation assumes that the object point cloud model is perfectly known, i.e., the 3D position of each point in the object model is a constant in the measurement model. In practice the object shape is unknown and has to be obtained from the (noisy) observations. The point update mechanism outside the Kalman filter is introduced in Sec. 3.5.3.

In the following the additional state vector descriptors *full* and *red* will be skipped if it is clear which model is used or if the context applies to both models.

3.4. System Model

The system model describes an expectation on the temporal development of the object state entries and is used for state prediction. It consists of two parts. First, the movement of the tracked object which is modeled as circular path motion based on the simplified bicycle model as proposed in Sec. 2.5.3. This is also referred to as the *dynamic model* of the observed object. Secondly, since the ego-vehicle can also move, the absolute ego-motion has to be considered at state prediction as well. Finally, both absolute motions have to be combined to a relative motion.

As proposed in Sec. 2.6.4, the system model is a mathematical description of the transformation of the time-discrete state $\boldsymbol{x}(k-1)$ to $\boldsymbol{x}(k)$. Let t denote the continuous time at discrete step $k-1$, then the continuous time at step k is equivalent to $t + \Delta t$, for a given time interval Δt. This yields an alternative formulation of (2.36) as

$$\boldsymbol{x}(t+\Delta t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t+\Delta t), \boldsymbol{w}(t+\Delta t)) \tag{3.10}$$

or, in short form, by indicating the time parameters t with a prime and $t + \Delta t$ with a double prime:

$$\boldsymbol{x}'' = f(\boldsymbol{x}', \boldsymbol{u}'', \boldsymbol{w}''). \tag{3.11}$$

with $\boldsymbol{u}$ a known control input including information on the ego-motion, and $\boldsymbol{w}$ a noise term. The system model is used in the Kalman filter prediction step, i.e., $\widehat{\boldsymbol{x}}''^{-} = f(\widehat{\boldsymbol{x}}'^{+}, \boldsymbol{u}'', \mathbf{0})$. It is successively derived in the following.

3.4.1. Motion of Observed Vehicle

Assuming a stationary ego-vehicle system, the object dynamics are modeled by the following nonlinear differential equations:

$$
\begin{aligned}
{}^{e}\dot{X}_v &= \underline{v}\sin(\underline{\psi} + \underline{\beta}) && (3.12)\\
{}^{e}\dot{Z}_v &= \underline{v}\cos(\underline{\psi} + \underline{\beta}) && (3.13)\\
\dot{\psi} &= \underline{\dot{\psi}} && (3.14)\\
\dot{v} &= \underline{\dot{v}} && (3.15)\\
\ddot{\psi} &= 0 && (3.16)\\
\ddot{v} &= 0 && (3.17)\\
{}^{o}\dot{X}_v &= 0 && (3.18)\\
{}^{o}\dot{Z}_v &= 0 && (3.19)\\
\dot{\beta} &= 0 && (3.20)\\
{}^{o}\dot{X}_1 &= 0 && (3.21)\\
{}^{o}\dot{Y}_1 &= 0 && (3.22)\\
{}^{o}\dot{Z}_1 &= 0 && (3.23)\\
&\vdots \\
{}^{o}\dot{X}_M &= 0 && (3.24)\\
{}^{o}\dot{Y}_M &= 0 && (3.25)\\
{}^{o}\dot{Z}_M &= 0 && (3.26)
\end{aligned}
$$

State variables are underlined above to distinguish the Newton notation for time derivatives of the differential equation system, $\dot{x}(t) \equiv \frac{d}{dt}x(t)$, from parameters of the Kalman filter state, which are considered as constants in these equations. This corresponds to the (constantly) accelerated velocity, constant yaw rate (AVCY) model as proposed in Sec. 2.5.3.

Note that equations (3.12) and (3.13) do not contain the object origin in ego-coordinates, as defined in the state vector, but the vehicle origin in ego-coordinates. Thus, only for the special case that $\boldsymbol{O}_o = \boldsymbol{O}_v$, these equations can be applied to predict the corresponding state vector entries. In general, the object origin, with $\boldsymbol{O}_o \neq \boldsymbol{O}_v$, has a different motion than the rotation point.

Thus, instead of formulating the vehicle motion in the ego-system, the following transformation sequence is applied. First, the translation and rotation of an arbitrary point within the vehicle coordinate system is computed based on the motion parameters v, $\dot{v}$, and $\dot{\psi}$. This gives a predicted position of this point in the vehicle coordinate system S_v. Then, the predicted point position is transformed into object coordinates, using ${}^{o}X_v$, ${}^{o}Z_v$, and β, and finally into ego-coordinates based on ${}^{e}X_o$, ${}^{e}Z_o$, and ψ.

The translative motion ${}^{v}\boldsymbol{T}(\Delta t)$ of a given point in vehicles coordinates, within a discrete time interval Δt, is given by

$$ {}^{v}\boldsymbol{T}(\Delta t) = \begin{bmatrix} \int_0^{\Delta t}(v+\dot{v}\tau)\sin(\dot{\psi}\tau)d\tau \\ 0 \\ \int_0^{\Delta t}(v+\dot{v}\tau)\cos(\dot{\psi}\tau)d\tau \end{bmatrix}. \tag{3.27} $$

If v, $\dot{v}$, and $\dot{\psi}$ are constant (which is a proper assumption for a small time interval Δt), the integrals can be solved to

$$ {}^{v}\boldsymbol{T}(\Delta t) = \begin{bmatrix} \frac{1}{\dot{\psi}^2}\Big(\dot{v}\sin(\dot{\psi}\Delta t) - (\dot{v}\dot{\psi}\Delta t + v\dot{\psi})\cos(\dot{\psi}\Delta t)\Big) + \frac{v}{\dot{\psi}} \\ 0 \\ \frac{1}{\dot{\psi}^2}\Big((\dot{v}\dot{\psi}\Delta t + v\dot{\psi})\sin(\dot{\psi}\Delta t) + \dot{v}\cos(\dot{\psi}\Delta t) - \dot{v}\Big) \end{bmatrix}. \tag{3.28} $$

Following L'Hospital's Rule, for $\lim \dot{\psi} \to 0$ the translation vector reduces to

$$ {}^{v}\boldsymbol{T}(\Delta t) = \begin{bmatrix} 0 \\ 0 \\ v\Delta t + \frac{1}{2}\dot{v}\Delta t^2 \end{bmatrix} \tag{3.29} $$

which corresponds to a translation in the ${}^{v}Z$-direction only. By definition, the vehicle ${}^{v}Z$-axis corresponds to the longitudinal axis of the car, i.e., the car is driving straight. The reduced translation vector depends on the velocity and acceleration only.

In practice, for small $\dot{\psi}$, the nonlinear functions $f_x(\tau) = (v+\dot{v}\tau)\sin(\dot{\psi}\tau)$ and $f_z(\tau) = (v+\dot{v}\tau)\cos(\dot{\psi}\tau)$ are linearized via Taylor expansion (see Ap. A.2), yielding:

$$ {}^{v}\boldsymbol{T}(\Delta t) \approx \begin{bmatrix} \frac{1}{2}v\dot{\psi}\Delta t^2 + \frac{1}{3}\dot{v}\dot{\psi}\Delta t^3 - \frac{1}{24}v\dot{\psi}^3\Delta t^4 \\ 0 \\ v\Delta t + \frac{1}{2}\dot{v}\Delta t^2 - \frac{1}{6}v\dot{\psi}^2\Delta t^3 - \frac{1}{8}\dot{v}\dot{\psi}^2\Delta t^4 \end{bmatrix}. \tag{3.30} $$

The rotational motion around the vehicle origin in time interval Δt integrates to $\dot{\psi}\Delta t$, thus

$$ {}^{v}\mathsf{R}(\dot{\psi}\Delta t) = \begin{bmatrix} \cos(\dot{\psi}\Delta t) & 0 & \sin(\dot{\psi}\Delta t) \\ 0 & 1 & 0 \\ -\sin(\dot{\psi}\Delta t) & 0 & \cos(\dot{\psi}\Delta t) \end{bmatrix}. \tag{3.31} $$

The total movement in vehicle coordinates can be expressed in homogeneous coordinates as

$$ {}^{v}\mathbf{P}'' = \begin{bmatrix} {}^{v}\mathsf{R}'(\dot{\psi}\Delta t) & {}^{v}\boldsymbol{T}'(\Delta t) \\ \mathbf{0}_3^{\mathsf{T}} & 1 \end{bmatrix} {}^{v}\mathbf{P}' \tag{3.32} $$

$$ = {}^{v}\mathsf{M}\,{}^{v}\mathbf{P}'. \tag{3.33} $$

Accordingly, the translation vector of a point in ego-coordinates is computed by first transforming the point into vehicle coordinates, applying the motion matrix, and then re-transforming the point into the ego-coordinate system as follows:

$$ {}^{e}\mathbf{P}'' = \underbrace{{}^{e}\mathsf{W}_o\, {}^{o}\mathsf{W}_v\, {}^{v}\mathsf{M}\, {}^{v}\mathsf{W}_o\, {}^{o}\mathsf{W}_e}_{{}^{e}\mathsf{M}}\, {}^{e}\mathbf{P}' \tag{3.34}$$

Applying ${}^{e}\mathsf{M}$ to ${}^{e}\boldsymbol{O}_o$, represented by ${}^{e}\boldsymbol{T}_o = [{}^{e}X_o, 0, {}^{e}Z_o]^{\mathsf{T}}$, yields the predicted object origin in ego-coordinates. Since ${}^{o}\mathsf{W}_e\, {}^{e}\mathbf{T}_o = {}^{o}\boldsymbol{T}_o = [0\ 0\ 0]^{\mathsf{T}}$, (3.34) can be further simplified, i.e.

$$ {}^{e}\mathbf{T}''_o = {}^{e}\mathsf{W}_o\, {}^{o}\mathsf{W}_v\, {}^{v}\mathsf{M} \left[(-R'_y(\beta)\, {}^{o}\boldsymbol{T}'_v)^{\mathsf{T}}\ 1\right]^{\mathsf{T}}. \tag{3.35}$$

3.4.2. Motion of Ego-vehicle

In the general case where the ego-vehicle is not stationary, the ego-motion matrix ${}^{e''}\mathsf{W}_{e'}\,(\boldsymbol{u}(k))$, transforming a given point in the ego-coordinate system at one discrete time step into the next one, has to be considered. The control vector $\boldsymbol{u}$, containing the rotational and translational parameters of the ego-motion, and the time index k will be skipped in the following for ease of readability. The ego-motion compensated predicted position of the reference point is then given by

$$ {}^{e''}\boldsymbol{T}''_o = {}^{e''}\mathsf{W}_{e'}\, {}^{e'}\boldsymbol{T}''_o. \tag{3.36}$$

Then,

$$ \Delta\, {}^{e}\boldsymbol{T}_o = {}^{e''}\boldsymbol{T}''_o - {}^{e'}\boldsymbol{T}'_o = \begin{bmatrix} {}^{e}\dot{X}_o \\ {}^{e}\dot{Y}_o \\ {}^{e}\dot{Z}_o \end{bmatrix} \Delta t \tag{3.37}$$

is the time discrete update of ${}^{e}X_o$ and ${}^{e}Z_o$ and thus a generic alternative formulation of (3.12) and (3.13) that does not longer require the reference point to be identical to the rotation point.

The method proposed by Badino [H. Badino 2004] is used for computation of the ego-motion matrix ${}^{e''}\mathsf{W}_{e'}$. This method estimates all six degrees of freedom of the camera motion from a sequence of stereo images based on correspondeces of static points in the scene. Ego-motion information from inertial sensors is also incorporated in this method. Any other approach for estimating the camera motion between two consecutive frames could be used alternatively.

3.4.3. Combined Relative Motion

The state updates for a non-stationary ego-coordinate system are, in time discrete form,

$$\Delta \boldsymbol{x} = \begin{bmatrix} \Delta\, {}^{e}X_o \\ \Delta\, {}^{e}Z_o \\ \Delta\psi \\ \Delta v \\ \Delta\underline{\dot{\psi}} \\ \Delta\underline{\dot{v}} \\ \Delta\, {}^{o}X_v \\ \Delta\, {}^{o}Z_v \\ \Delta\beta \\ \Delta\, {}^{o}X_1 \\ \Delta\, {}^{o}Y_1 \\ \Delta\, {}^{o}Z_1 \\ \vdots \\ \Delta\, {}^{o}X_M \\ \Delta\, {}^{o}Y_M \\ \Delta\, {}^{o}Z_M \end{bmatrix} = \begin{bmatrix} {}^{e''}X_o'' - {}^{e'}X_o' \\ {}^{e''}Z_o'' - {}^{e'}Z_o' \\ \underline{\dot{\psi}}\Delta t + \underline{\dot{\psi}}_{\text{ego}}\Delta t \\ \underline{\dot{v}}\Delta t \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.38}$$

where $\dot{\psi}_{\text{ego}}$ indicates the yaw rate of the ego-vehicle. Again, underlined variables correspond to parameters and should not be confused with the operator of the continuous differential equations. The system equations above are given for the full object state. The updates for the reduced model are straight forward. One simply has to skip the object point update equations. $\Delta\boldsymbol{x}$ can be seen as a function of $\boldsymbol{x}$, $\boldsymbol{u}$, and Δt.

Finally, the nonlinear functional model $f(\boldsymbol{x}', \boldsymbol{u}'', \boldsymbol{w}'')$ is defined as

$$\boldsymbol{x}'' = f(\boldsymbol{x}', \boldsymbol{u}'', \boldsymbol{w}'') = \boldsymbol{x}' + \Delta\boldsymbol{x} + \boldsymbol{w}'' \tag{3.39}$$

with $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, C_{ww})$ an additive zero-mean, white, Gaussian noise term.

Following the extended Kalman filter approach (cf. Sec. 2.6.4), f has to be linearized at $\widehat{\boldsymbol{x}}'$ for propagation of the state covariance matrix. Then, the matrix A, with

$$A = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\widehat{\boldsymbol{x}}'} \tag{3.40}$$

defines the linearized system matrix. Fig. 3.6 shows the distribution of the non-zero elements of A for the reduced parameter vector. All elements on the main diagonal are one. In practice, only the non-zero entries of the upper left submatrix have to be recomputed each time step. The analytical computation of this Jacobian is quite lengthy and, thus, approximated numerically in practice.

If the full model is used, the matrix A is easily extended, since for each point the additional entries are 1 at the diagonal and zero otherwise.

The system model presented here is a good approximation of the *normal* vehicle dynamics in many traffic scenes, including many urban scenarios. However, there are also

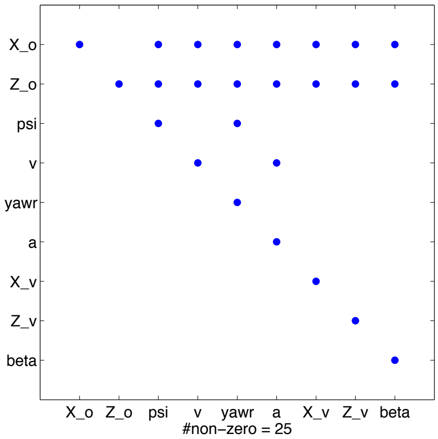


Figure 3.6.: There are 25 non-zero elements in the linearized system matrix for the reduced state vector (without point coordinates). Since the diagonal entries are 1, only 16 values have to be computed each time step.

limits and drawbacks. For example, this model does not incorporate the correlation between velocity and yaw rate, i.e., it is possible that the object orientation can change (non-zero yaw rate), although the vehicle is not moving at all (velocity zero). This unrealistic condition must be suppressed in practice. The motion model has further been derived for cars, vans, or motorbikes, driving on a path of low curvature at relative slow velocity. Any physical effects occurring at higher dynamics, e.g., a dynamic side slip angle, are ignored in that model. The model is also not suited for representing turn maneuvers of long trucks or (articulated) buses, which are excluded from the scope of this thesis.

In Sec. 3.11 an extension of this motion model, which also incorporates the yaw acceleration to yield a better approximation of the vehicle dynamics at intersection turn maneuvers, is given.

With the presented system model it is possible to predict the object state. In the following section it will be described, how the measurements can be predicted based on this predicted state.

3.5. Measurement Model

The measurement model relates the unknown state parameters to the observations. First, a brief overview on the total measurement model is given, which is composed of different components. Then, these components will be introduced in the consecutive sections in detail.

3.5.1. Total Measurement Model

The total measurement model combines the following components,

- point cloud measurements, with $\boldsymbol{z}^{(p)} = h^{(p)}(\boldsymbol{x})$; cf. Sec. 3.5.2 and 3.5.3

- rotation point measurements, with $\boldsymbol{z}^{(rot)} = h^{(rot)}(\boldsymbol{x})$; cf. Sec. 3.5.4
- radar velocity measurements, with $\boldsymbol{z}^{(v)} = h^{(v)}(\boldsymbol{x})$; cf. Sec. 3.5.5 (optional)

where each component consists of a measurement vector $\boldsymbol{z}^{(\cdot)}$ and a functional model $h^{(\cdot)}$ relating the state parameters to the measurements, as introduced in Sec. 2.6.4.

The *point cloud* component is the main part of the total measurement model. The measured 3D point coordinates contain information on the object pose and motion state. In addition, geometrical *rotation point* measurements are derived from dense stereo data, while *radar measurements* improve the velocity estimate. The details will be presented in the corresponding sections below.

The different measurement components are summarized by concatenation to the total measurement model h, yielding

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{z}^{(p)} \\ \boldsymbol{z}^{(rot)} \\ \boldsymbol{z}^{(v)} \end{bmatrix} = \begin{bmatrix} h^{(p)}(\boldsymbol{x}) \\ h^{(rot)}(\boldsymbol{x}) \\ h^{(v)}(\boldsymbol{x}) \end{bmatrix} = h(\boldsymbol{x}). \tag{3.41}$$

This measurement model is then applied, following the standard extended Kalman filter approach.

The velocity measurement is optional, i.e., the system also performs very well if no radar information is available. It is straight forward to add additional measurement modules accordingly. In the following sections, $\boldsymbol{z}$ will always refer to the total measurement vector, and h to the corresponding nonlinear measurement model, if not stated otherwise.

The measurements of the different modules are assumed to be uncorrelated, which will be concretized in Sec. 3.6.2.

3.5.2. Point Cloud Measurements

The 3D point cloud representing the object's shape is directly observable by the stereo vision sensor. Since each point has a fixed position within the object coordinate system, it also has an influence on the object pose parameters.

Let $M(k)$ denote the number of points at discrete time step k. A measurement of point $\mathcal{P}_m$, $1 \leq m \leq M(k)$, in the image consists of a sub-pixel accurate image position (u_m, v_m) and stereo disparity d_m at this position. This information is summarized in the vector $\boldsymbol{z}_m(k) = [u_m(k), v_m(k), d_m(k)]^\mathsf{T}$. In the following, the time indices are skipped for better readability, when all variables refer to the same time instance. If the association between an object point and an image position is known at one point in time, it is possible to reassign it by tracking the image position over time using a feature tracker such as the KLT tracker [Tomasi and Kanade 1991]. All M measurements build the *point measurement vector* $\boldsymbol{z}^{(p)}$ with

$$\boldsymbol{z}^{(p)} = \left[\boldsymbol{z}_1^\mathsf{T}, \boldsymbol{z}_2^\mathsf{T}, ..., \boldsymbol{z}_M^\mathsf{T}\right]^\mathsf{T}. \tag{3.42}$$

The nonlinear measurement model $\boldsymbol{z}^{(p)} = h^{(p)}(\boldsymbol{x})$ results from the perspective camera model as proposed in (2.3) and (2.5). With this measurement model it is possible

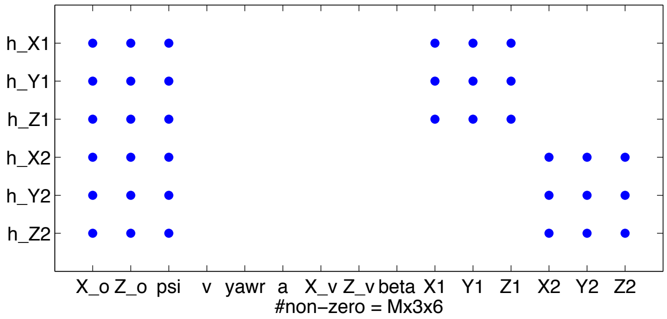


Figure 3.7.: Example measurement matrix for $M = 2$ measurements and the full state vector. For each measurement, there are 3×6 non-zero elements of the linearized measurement matrix to be computed. In the reduced state model, the point-wise 3×3 blocks at the right-hand side are omitted.

to predict the measurements $\hat{\boldsymbol{z}}_m = [\hat{u}_m, \hat{v}_m, \hat{d}_m]^\mathsf{T}$ based on the current a priori state estimate $\hat{\boldsymbol{x}}^-$, i.e., $\hat{\boldsymbol{z}}_m = h_m^{(p)}(\hat{\boldsymbol{x}}^-)$. The function $h^{(p)}$ is defined element-wise as

$$h_m^{(p)}(\boldsymbol{x}) = \left[f_x \frac{{}^cX_m}{{}^cZ_m} + x_0, \quad f_y \frac{{}^cY_m}{{}^cZ_m} + y_0, \quad f_x \frac{b}{{}^cZ_m} \right]^\mathsf{T} \tag{3.43}$$

where f_x and f_y represent the scaled principal distance of the camera for horizontal and vertical direction, respectively, (x_0, y_0) the principal point, and b the base line of the stereo system. ${}^c\boldsymbol{P}_m = [{}^cX_m, {}^cY_m, {}^cZ_m]^\mathsf{T}$ is the point ${}^o\boldsymbol{P}_m$ in the camera system S_c. The total transformation is composed of an object to ego transformation (parameterized by the state variables ψ, ${}^eX_o$, and ${}^eZ_o$), and the constant transformation ${}^c\mathsf{W}_e$ between ego and camera system:

$${}^c\mathbf{P}_m = {}^c\mathsf{W}_e \, {}^e\mathsf{W}_o \, {}^o\mathbf{P}_m \tag{3.44}$$

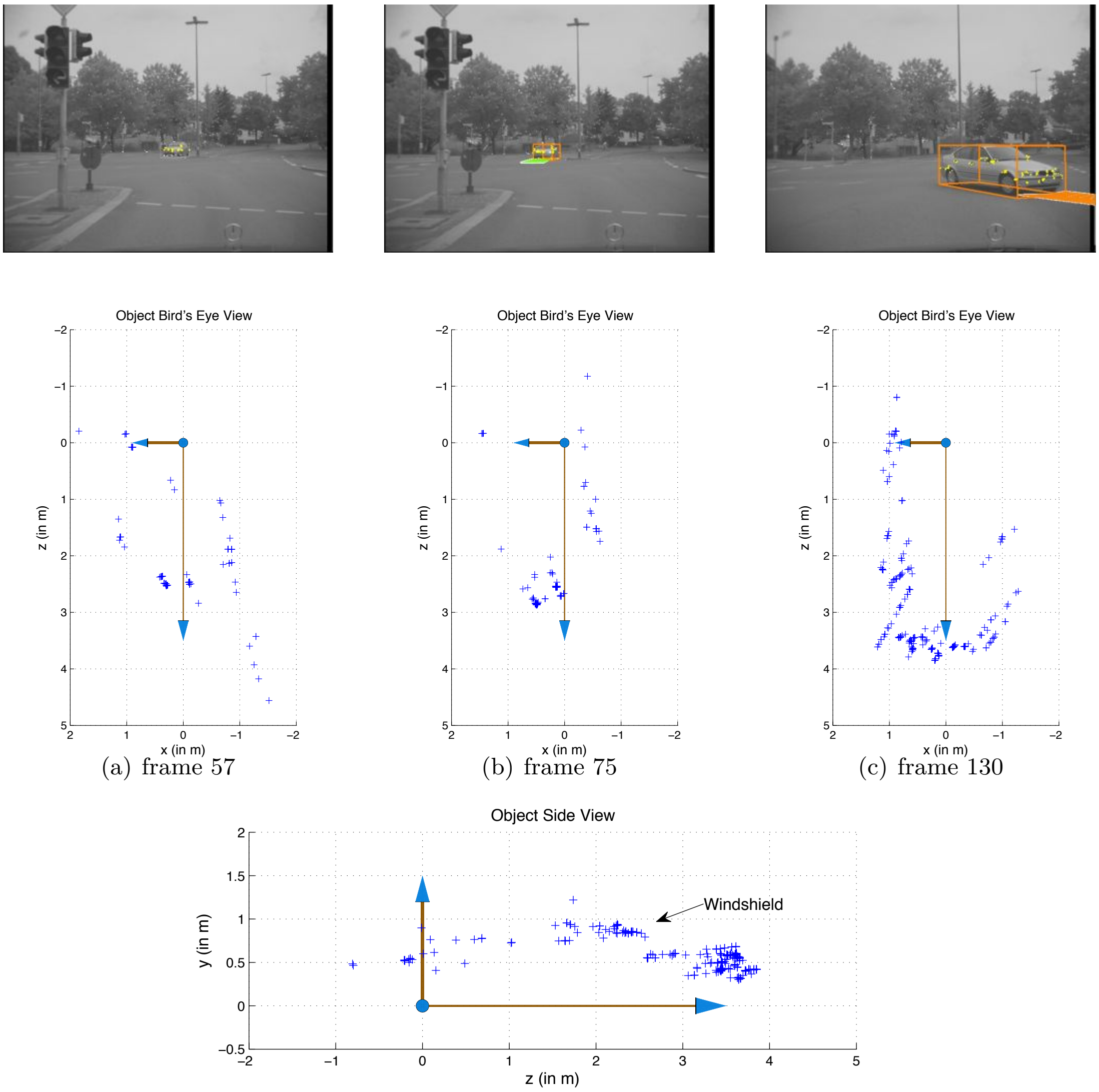
with ${}^c\mathsf{W}_e$ including the rotation and translation of the camera with respect to the ego-system. The equations above assume rectified image pairs, i.e., image distortions have been compensated before.

Following the extended Kalman filter approach, $h^{(p)}$ has to be linearized at the current estimated state $\hat{\boldsymbol{x}}^-$ for computation of the Kalman gain and propagation of the state covariance matrix, yielding:

$$\mathsf{H}^{(p)} = \left. \begin{bmatrix} \frac{\partial h_1^{(p)}}{\partial \boldsymbol{x}} \\ \frac{\partial h_2^{(p)}}{\partial \boldsymbol{x}} \\ \vdots \\ \frac{\partial h_M^{(p)}}{\partial \boldsymbol{x}} \end{bmatrix} \right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}^-} . \tag{3.45}$$

The non-zero elements of the measurement matrix are shown in Fig. 3.7.

As will be seen later, the tracking performance increases with the number of contributing points. A varying number of M points due to loss or addition of newly tracked points can be easily handled, since each point simply adds another three measurements to the filter.



(a) frame 57

(b) frame 75

(c) frame 130

(d) frame 130

Figure 3.8.: Bird's eye view of the 3D point model of an oncoming vehicle at different time steps is shown from (a) to (c). The initial noisy point cloud is refined over time and the shape of a vehicle becomes visible in (d) from side view. All points are given in object coordinates.

3.5.3. External Shape Model Update

Since the exact position of a given object point is typically not known at initialization, it has to be estimated from the noisy measurements. An example for an initial noisy object point cloud and the same point cloud refined over several time steps is given in Fig. 3.8. As can be seen, the initial object model can be quite noisy, incomplete and will usually contain outliers.

In the full model, the object point positions are estimated and updated by the Kalman filter. However, if the problem of motion estimation is separated from the problem of shape reconstruction for real-time applicability, the point positions have to be refined outside the Kalman filter. Without updating the existing object points, the noisy positions lead to systematic errors in the prediction step.

The external point update is performed in two steps. First, each point is updated independently of all other points based on its measured position, yielding an estimate ${}^{o}\widehat{\boldsymbol{P}}^{*}_{m}(k)$. Then, all updated point positions are corrected by a common rigid transformation to the posterior estimated position ${}^{o}\widehat{\boldsymbol{P}}_{m}(k)$, ensuring that the centroid and the principal axis of the point cloud are not changed by the individual updates as will be motivated in more detail below.

Step 1: For each object point $\mathcal{P}_m$, observed at the current time step k and K previous discrete time steps $k-1, \ldots, k-K$, $K \geq 1$, a maximum likelihood estimation, assuming uncorrelated measurements and zero-mean Gaussian measurement noise, is given by

$$ {}^{o}\widehat{\boldsymbol{P}}^{*}_{m}(k|k, k-1, \ldots, k-K) = \left[\sum_{\kappa=k-K}^{k} \left({}^{o}C^{(m)}_{PP}(\kappa) \right)^{-1} \right]^{-1} \sum_{\kappa=k-K}^{k} \left({}^{o}C^{(m)}_{PP}(\kappa) \right)^{-1} {}^{o}\boldsymbol{P}_{m}(\kappa) \tag{3.46} $$

where ${}^{o}C^{(\mathrm{m})}_{PP}(\kappa)$ denotes the 3×3 covariance matrix of ${}^{o}\boldsymbol{P}_{m}(\kappa)$ with respect to the object coordinate system. This equation can be efficiently computed in a time-recursive formulation:

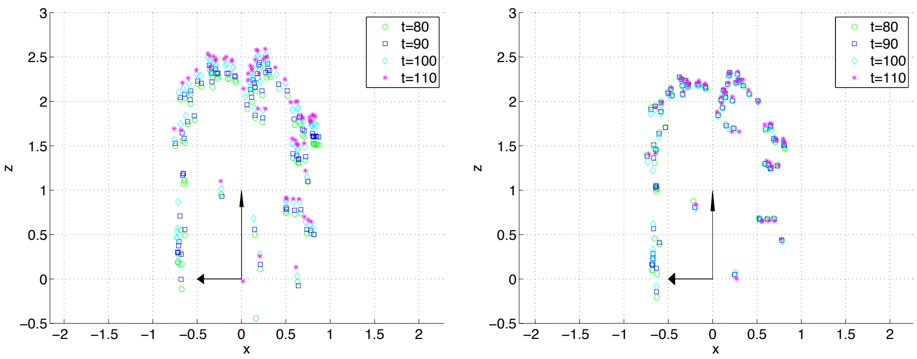
$$ \begin{aligned} {}^{o}\widehat{\boldsymbol{P}}^{*}_{m}(k) &= \left[C^{(m)}_{\Sigma}(k-1) + {}^{o}C^{(m)}_{PP}(k) \right]^{-1} \\ &\quad \left[C^{(m)}_{\Sigma}(k-1)\, {}^{o}\widehat{\boldsymbol{P}}_{m}(k-1) + \left({}^{o}C^{(m)}_{PP}(k) \right)^{-1} {}^{o}\boldsymbol{P}_{m}(k) \right] \end{aligned} \tag{3.47} $$

$$ C^{(m)}_{\Sigma}(k) = C^{(m)}_{\Sigma}(k-1) + {}^{o}C^{(m)}_{PP}(k) \tag{3.48} $$

where $C^{(m)}_{\Sigma}(k)$ corresponds to the sum of all covariance matrices ${}^{o}C^{(m)}_{PP}$ for a given point up to time step k.

Step 2: Let Q denote an index set of points that have been observed at two consecutive time steps and not detected as outlier, with $Q \subseteq \{1 \ldots M\}$. Then, each updated point ${}^{o}\widehat{\boldsymbol{P}}^{*}_{q}(k)$ is corrected as follows:

$$ \begin{aligned} {}^{o}\widehat{\boldsymbol{P}}_{q}(k) =& R_{y}^{-1}(\Delta_{\theta}) \left({}^{o}\widehat{\boldsymbol{P}}^{*}_{q}(k) - {}^{o}\bar{\boldsymbol{P}}(k) \right) \\ &+ {}^{o}\bar{\boldsymbol{P}}(k-1), \quad \forall q \in Q. \end{aligned} \tag{3.49} $$



(a) (b)

Figure 3.9.: Object points from bird's eye view at four time steps. (a) If all object points are updated independently by weighted averaging (step 1 only), they drift in the moving direction as the vehicle accelerates. (b) The gauge consistency correction (step 2) successfully prevents a point drift.

with ${}^{o}\bar{\boldsymbol{P}}(k) = \frac{1}{|Q|}\sum_{q\in Q} {}^{o}\widehat{\boldsymbol{P}}^{*}_{q}(k)$ the mean of the points in Q at time step k and $R_y(\Delta_\theta) \in \mathbb{R}^{3\times 3}$ a rotation about the height axis by angle Δ_θ. This angle is defined as the difference between the orientation of the main principal component of the points in Q at time k and $k-1$. At this step it is assumed that the rotation between both point clouds does not change more than $\pi/2$ rad. This global correction ensures that the sum of all point updates is zero (*centroid consistency*) and that the orientation of the principal component of the adapted point cloud is equal to the orientation of the previous point cloud in object coordinates (*principal component consistency*). It prevents that points can systematically drift within the object coordinate system in situations where the predicted object pose differs considerably from the actual pose, without changing the pose and motion parameters.

The effect is demonstrated in Fig. 3.9. In this example, the tracked vehicle is accelerating. As can be seen in (a), updating the point positions within the local object coordinate system by weighted averaging (step 1 only), leads to a drift in the movement direction. This indicates that the estimated velocity is too low. The gauge correction in (3.49) prevents the drift, i.e., errors between the predicted and measured point positions must be mainly compensated by altering the motion parameters in the filter, instead of changing a point's relative position with respect to the local object coordinate system.

3.5.4. Rotation Point Measurements

The origin of the vehicle system, $\boldsymbol{O}_v$, has to be close to the actual vehicle's center rear axle to be able to correctly predict the object pose at turn maneuvers. Since the vehicle origin defines the rotational center, larger errors in its position lead to a significant prediction error at rotational movements, which directly impacts the resulting motion parameter estimates.

Initially, the translation ${}^{o}\boldsymbol{T}_v$ between the (arbitrary) object system and the center rear axle is typically unknown and has to be estimated over time. The noisy point cloud measurements alone are not sufficient to constrain the position of $\boldsymbol{O}_v$, since it becomes only weakly observable at turn maneuvers. At straight motion or as the tracked vehicle is not moving, the rotational center cannot be observed at all.

The idea is to constrain the position of the rotation point geometrically to the object center in lateral direction by additional measurements. In longitudinal direction, the exact location of the center rear axis is not that easy to define and varies between vehicles. However, as introduced before, the location can be approximated to be at a constant distance ρ from the object's rear.

The cuboid model provides a basic semantical meaning on vehicle sides (front, rear, left, right) or characteristic points, e.g., front left corner or center of the right side. If the dimension of the object is known, it is straight forward to describe the rotation point relative to a given object side or corner. This means, all corners or sides, observable by a sensor, can be used to constrain the position of the rotation point.

In the following, it will be first introduced how the rotation point measurement model is defined. Then, the effect of the rotation point measurements is illustrated based on a short experiment. Finally, two examples are presented that compute the geometric location of the center rear axle of a vehicle based on dense stereo data. These examples may be skipped at first reading.

Rotation Point Measurement Model

The rotation point measurements are integrated as direct observations of the state entries, i.e., the measurement model is defined by

$$\boldsymbol{z}^{(rot)} = h^{(rot)}(\boldsymbol{x}) = \mathsf{H}^{(rot)}\boldsymbol{x}, \tag{3.50}$$

with

$$\mathsf{H}^{(rot)} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\ldots0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\ldots0 \end{bmatrix}, \tag{3.51}$$

and

$$\boldsymbol{z}^{(rot)} = \left[{}^{o}X_v{}^{*}, {}^{o}Z_v{}^{*}\right]^{\mathsf{T}}. \tag{3.52}$$

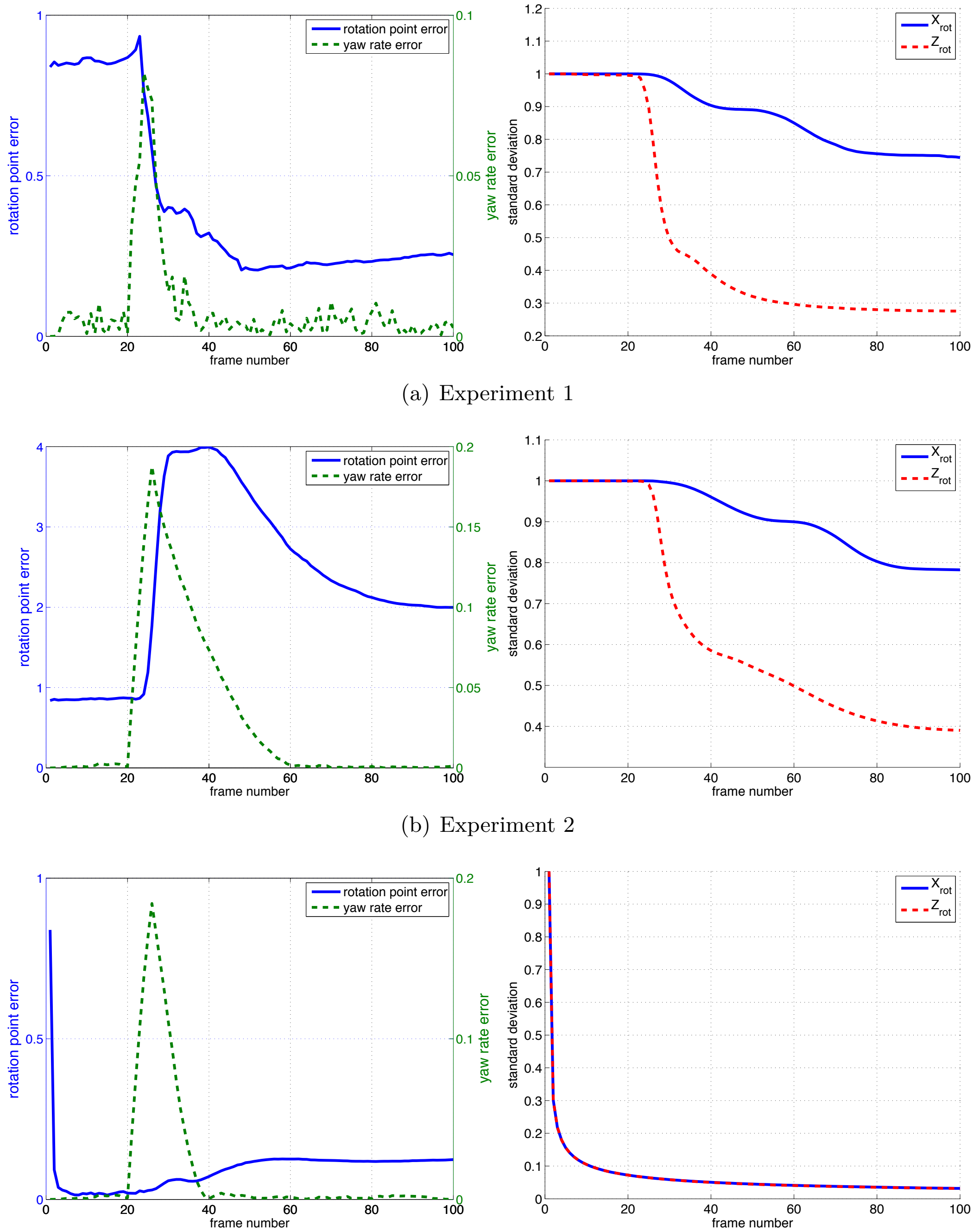
The measurements ${}^{o}X_v{}^{*}$ and ${}^{o}Z_v{}^{*}$ are computed from external modules, corresponding to the measured lateral and longitudinal position of the rotation point in object coordinates.

The corresponding stochastical model incorporating the uncertainties of the rotation point measurements is addressed in Sec. 3.6.2.

Effect of Rotation Point Measurements

The following experiment should demonstrate the importance of the rotation point measurements.

In a simulation, a rigid 3D point cloud, representing a virtual vehicle, is moved along a predefined trajectory (the simulation environment will be presented in detail in Sec. 4.2). The trajectory has been generated by the proposed vehicle motion model. The ground truth vehicle state, including the rotation point position, is known for all



(a) Experiment 1

(b) Experiment 2

(c) Experiment 3

Figure 3.10.: Estimation error of rotation point and yaw rate (left column) and the corresponding uncertainties (right column) over time. The rotation point position can only be observed during rotational movements (top row). Here, a turn maneuver starts at frame 20. If the filter cannot adapt the yaw rate quickly enough, the error is compensated by wrongly altering the rotation point position (middle row). Additional measurements for the rotation point position significantly reduce the estimation error (bottom row).

time steps. The maneuver consists of a straight-line motion phase ($\dot{\psi} = 0$), followed by a turning maneuver ($\dot{\psi} > 0$).

The question is now, how well does the Kalman filter (as proposed so far) reconstruct the true position of the rotation point if the estimate is initialized about 0.8 m away from the ground truth?

Three approaches will be compared:

- **Experiment 1:** filter with point cloud measurements, larger changes of the yaw rate are allowed via the system noise configuration.
- **Experiment 2:** filter with point cloud measurements, only small changes of the yaw rate are allowed via the system noise configuration.
- **Experiment 3:** equal to Exp. 2, with additional rotation point measurements (true position + Gaussian noise).

The expectation is that all filters reduce the absolute error between the true position and the estimated position, as the maneuvers starts. It is not observable before. Beside the rotation point error, the absolute error of the yaw rate estimate is considered. The results of this experiment are illustrated in Fig. 3.10.

At **Exp. 1**, the filter behaves as expected. During the first 20 frames (straight-line motion), the rotation point is not significantly changed, nor is the uncertainty decreased. During the maneuvering phase, the rotation point error is significantly reduced, as is the uncertainty of the ${}^{o}Z_{v}$ coordinate. Only for a few time steps the filter wrongly alters the rotation point to compensate for the sudden increase of the yaw rate, indicating the correlation between these two parameters. At **Exp. 2**, the filter is not able to adapt the yaw rate fast enough, since only small changes are allowed in this configuration. Instead, the Kalman filter moves the rotation point away from the true position to minimize the measurement/prediction error. The estimated uncertainty of the rotation point position decreases slower compared to the first row.

When additional measurements for the rotation point center are provided as in **Exp. 3**, the initial error is quickly decreased even before the maneuver starts. The same holds for the uncertainties. The good estimate on the rotational center helps to decrease the error of the yaw rate during the maneuvering phase much faster compared to the second row, although the filter parameterization has not changed. Due to the small uncertainties of the rotation point, the filter has to compensate for the measurement/ prediction error by correctly adapting the yaw rate.

This experiment emphasizes the importance of rotation point measurements, as the accuracy of the rotation point estimate has a direct influence on the rotational velocity, which is of particular interest in this contribution. Here, the measurements have been derived from the true position, corrupted by some additional Gaussian noise. In practice, one has to compute such measurements from the available sensor data. Two examples will be given in the following.

Example 1: Stereo Profile Histograms

The distribution of the object point cloud contains valuable information on the object dimension and boundaries, as long as there are sufficient points in the model and the point cloud covers all parts of the object. However, not only the tracked feature points on the object surface provide information on the object's size. With dense stereo disparity maps, there is much more information available. The idea is to analyze the distribution of **all** 3D points within a local neighborhood of the tracked object.

The approach can be sketched as follows:

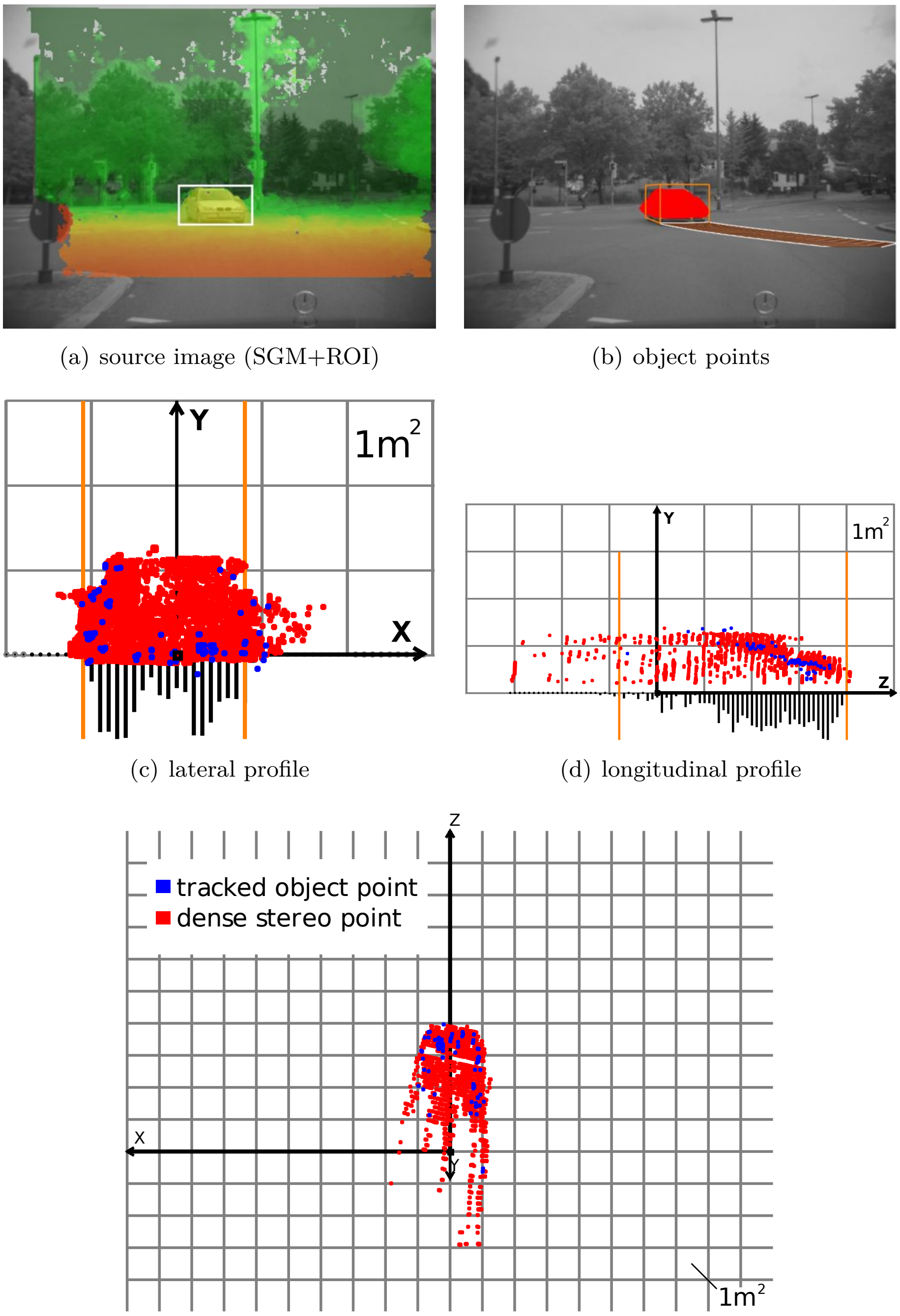
1. transform **all** 3D points within a region of interest (ROI) around the predicted object position into vehicle coordinates
2. quantize the resulting ${}^{v}X$-coordinates into a finite number of histogram bins (*lateral profile*)
3. quantize the resulting ${}^{v}Z$-coordinates into a finite number of histogram bins (*longitudinal profile*)
4. compute the lateral and longitudinal object boundaries based on these two 1D histograms
5. compute the rotation point position $[{}^{o}X_v{}^{*}, {}^{o}Z_v{}^{*}]^{\mathsf{T}}$ from the object boundaries based on the cuboid model (cf. Fig. 3.5 in Sec. 3.3.3).

The reliable computation of the object boundaries from the 1D histograms is the most challenging task as will be concretized below. First, an example should illustrate the procedure in Fig. 3.11.

Here, an oncoming vehicle at about 30 m distance is considered. The input image is shown in (a) with the disparity map and region of interest superimposed. In this region, all points within an expected distance and height range, are visualized in red in (b). The corresponding bird's eye view on the selected points in vehicle coordinates (projected onto the ground plane) is shown in (e). Tracked points that are currently part of the object model are visualized in blue. The resulting quantized lateral and longitudinal stereo profiles are shown in (c) and (d) respectively. The length of the black bars is proportional to the number of points in a given bin. The detected object boundaries are superimposed by the orange lines. The maximum precision of the object boundaries is restricted to the histogram bin size.

Object Boundary Computation: As long as the object is isolated, very simple and fast algorithms can be applied to correctly estimate the object boundaries from the stereo profile histograms. For example by detecting the two most outer *significant* histogram entries, where significance is related to the number of points in the given bin. To be robust to outliers in the data, a pruning of the histogram is performed, i.e., a small constant percentage of extreme values are removed first.

In practice, this simple method fails in situations where the predicted object ROI, in which all points are assumed to belong to the given object, also includes other objects. For example, other moving cars in dense traffic scenarios or parking cars at the roadside. Thus, all potential object boundaries, i.e., positions in the histogram showing a significant increase or decrease of histogram entries within a small local neighborhood yield *object boundary candidates*. A prior on the expected object dimension (width and



(a) source image (SGM+ROI)

(b) object points

(c) lateral profile

(d) longitudinal profile

(e) bird's eye view

Figure 3.11.: Dense stereo data within the image ROI around a detected object is analyzed based on lateral and longitudinal stereo profile histograms. The detected object boundaries (orange lines) are used to derive geometric measurements for the rotation point and to update the object dimensions.

length) is used to sort out all object boundary configurations based on the candidates which deviate from the expectation. If more than one possible configuration survives this criteria, the solution which is closest to the predicted object state is selected.

As an optional extension, individual likelihoods are computed for each point in the ROI, indicating how likely does a given point belong to the tracked object or the static background. This includes motion information, which is very discriminative, for example, in separating the tracked object from close-by stationary obstacles. The likelihood computation is analog to the method which will be proposed in Sec. 3.9 with respect to assigning new points to an existing object. So far it is assumed, that there exists such an object likelihood. Accordingly, the points are not added with equal weight to the histogram, e.g., by incrementing a counter by one, but with a weight proportional to their object likelihood.

Another problem at estimating the object boundaries is visibility. Depending on the object orientation, some boundaries are hidden to the cameras, e.g., the rear side of an oncoming vehicle. Thus, in some situations only the visible side is reconstructed from the histograms. The opposite side is then assumed to be located at the expected distance, i.e., based on the dimension prior.

More sophisticated methods for detecting the object boundaries could be used alternatively, however, the sketched approach yields very promising and fast results in practice.

One drawback of the stereo profile histograms is that many point transformations from the image domain to the vehicle system have to be computed before the histograms can be further analyzed quite efficiently. Furthermore, the presented stereo profile approach is not capable of estimating the object orientation from the stereo data, i.e., a very good prior on the orientation of the vehicle's axis is required.

Example 2: Stixel Silhouettes

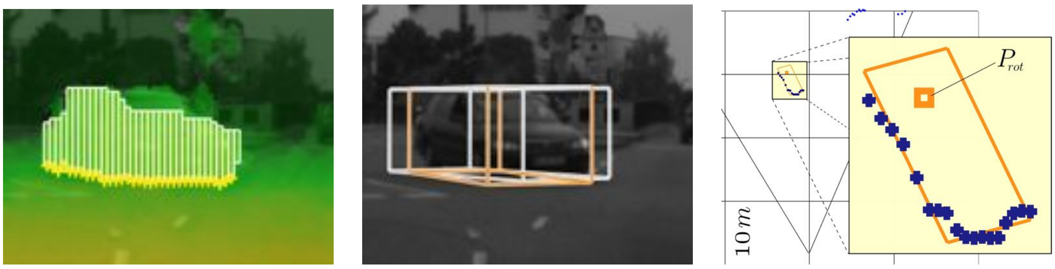
The second example for estimating the rotation point geometrically is based on the Stixel World representation as introduced in Sec. 2.3.4. Each stixel contains information on the distance and height of the obstacle that limits the freespace at this position. Clusters of neighboring stixels that can be assigned to a tracked object, provide a good estimate on the object's silhouette in the image. An example of such silhouette is given in Fig. 3.12(a).

The idea is to estimate the pose and dimension parameters of a cuboid in a maximum likelihood sense based on measurements derived from a cluster of stixels. The projection of the resulting cuboid should be consistent with the corresponding stixel silhouette in the image (see Fig. 3.12(b) and (c)). Since the stixel representation reduces the amount of data to process significantly, the solution can be computed quite efficiently.

In the following, first a brief overview on the approach is given. Then, the parameters to be estimated as well as the input measurements are concretized.

For a given object prior (cuboid pose and dimension) and a list of candidate stixel clusters, the method can be sketched as follows:

1. project the cuboid onto the image plane
2. find a cluster of stixels that overlaps significantly with the cuboid projection



(a) (b) (c)

Figure 3.12.: The stixel silhouette in (a) is used to refine the initial pose (white bounding box) in (b). The orange box indicates the consistent pose. The refinement result is shown from bird's eye view in (c), together with the predicted rotation point position.

3. derive several measurements from that cluster that constrain the object pose, e.g., the horizontal image coordinate and disparity of the most outer stixels in the cluster.

4. refine the prior cuboid pose and dimension based on these measurements using iterative least-squares estimation

5. compute the rotation point position $[{}^{o}X_v{}^{*},\, {}^{o}Z_v{}^{*}]^{\mathsf{T}}$ from the refined cuboid based on the cuboid model (cf. Fig. 3.5 in Sec. 3.3.3).

The unknown parameters to be estimated are

$$\boldsymbol{y} = [{}^{o}X_v{}^{*},\, {}^{o}Z_v{}^{*}, \beta^{*}, \xi^{*}(\widehat{\boldsymbol{x}})]^{\mathsf{T}}\,, \tag{3.53}$$

corresponding to the pose parameters of the vehicle system with respect to the object system as well as the size $\xi^{*}(\widehat{\boldsymbol{x}})$ of the side that is presumingly covered by more stixels based on the pose prior $\widehat{\boldsymbol{x}}$. Thus, only one dimension, width or length, is estimated at one time. With an increasing stereo uncertainty at larger distances (> 40 m), the parameter vector is reduced to contain only the rotation point position, since reliable orientation and size measurements cannot be obtained.

Six measurements are derived from the given stixel cluster and summarized to the stixel measurement vector $\boldsymbol{z}_s$ as

$$\boldsymbol{z}_s = [u_l, u_r, d_l, d_r, d_{s1}, d_{s2}]^{\mathsf{T}}\,. \tag{3.54}$$

The image columns of the left and right stixel of the corresponding stixel cluster, denoted as u_l and u_r respectively, define the viewing range and constrain the expected vehicle position in lateral direction (see Fig. 3.13). At the same time, the disparity values d_l and d_r of the boundary stixels introduce constraints on distance.

Depending on the given object hypothesis $\widehat{\boldsymbol{O}}$, the left and right most stixel are directly linked to corresponding object corners. The inner stixels cannot be assigned to a concrete point at the visible object sides that easy, although they also provide valuable

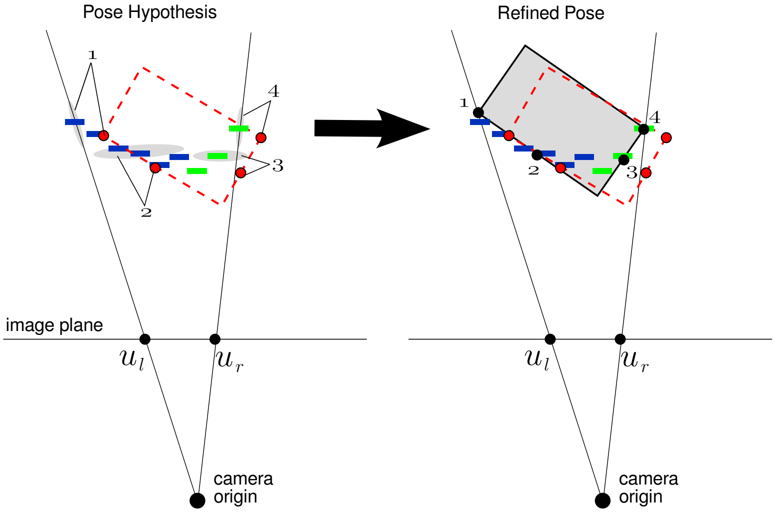


Figure 3.13.: Six measurements on the depth and lateral position are derived from a stixel cluster and assigned to four characteristic object points, that depend on visibility properties of the pose prior. The pose hypothesis is iteratively refined by a maximum likelihood estimation.

information on depth. Due to perspective, one or two vehicle sides are visible in the image at one time. Thus, the stixels are divided based on an expectation on the number of image columns to be covered by a visible side. The median disparity d_{si} over all stixels assigned to a given object side s_i, $i \in \{1, 2\}$, is taken as additional depth constraint on the center of that side. Since the median is more robust to outliers compared to the mean, inaccuracies in assigning the inner stixels to object sides are acceptable.

Let $h^{(s)}$ denote the functional model between the measurements $\boldsymbol{z}_s$ and the parameters $\boldsymbol{y}$, i.e., $\boldsymbol{z}_s = h^{(s)}(\boldsymbol{y})$. It transforms the two most outer visible corners and the visible side centers from vehicle coordinates into image coordinates and stereo disparities. All required parameters that are not included in $\boldsymbol{y}$, e.g., the pose parameters of object system w.r.t. the ego-system, are taken from the current object hypothesis $\widehat{\boldsymbol{O}}$ and assumed to be constant.

Now, the unknown parameters are estimated via the least-squares method (cf. Sec. 2.6.1). Since $h^{(s)}$ is nonlinear, it has to be linearized at $\boldsymbol{y}_0$ derived from the pose prior. The updates $\Delta\boldsymbol{y}$, with $h^{(s)}(\boldsymbol{y}) \approx h^{(s)}(\boldsymbol{y}_0 + \Delta\boldsymbol{y})$, are computed as

$$\Delta\boldsymbol{y} = \left(\mathsf{G}^\mathsf{T} \mathsf{C}_{ss}^{-1} \mathsf{G}\right)^{-1} \mathsf{G}^\mathsf{T} \mathsf{C}_{ss}^{-1} \left(\boldsymbol{c}_s - h^{(s)}\left(\boldsymbol{y}_0\right)\right), \tag{3.55}$$

where the matrix G indicates the Jacobian $\left.\frac{\partial h^{(s)}}{\partial \boldsymbol{y}}\right|_{\boldsymbol{y}=\boldsymbol{y}_0}$, and C_{ss} the covariance matrix of the stixel measurements. This estimation procedure is iterated a few times (typically three iterations are sufficient).

One advantage of the stixel-based approach, compared to the stereo profile histogram method, is, that the pose and dimension parameters are estimated jointly and not independently for the lateral and longitudinal direction. Furthermore, the prior orientation is also refined at this procedure.

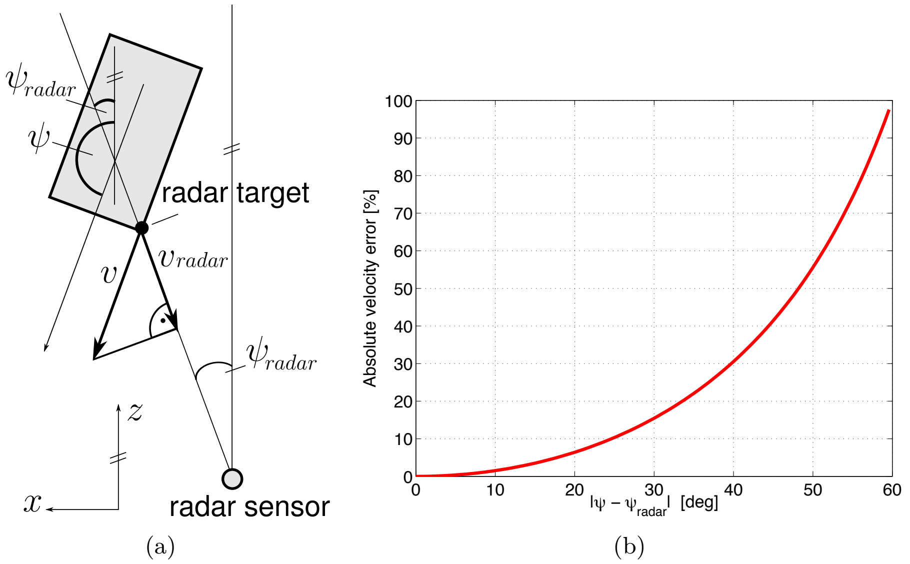


Figure 3.14.: (a) The measured radar velocity has to be transformed into the object velocity, defined along the longitudinal vehicle axis, using a nonlinear transformation function, based on the estimated object orientation. (b) Theoretical absolute velocity error, when orientation difference is ignored.

The stixel silhouette method presented above requires objects to be isolated. Any violations of the requirement that the most outer stixels of the selected stixel cluster correspond to the actual boundaries may lead to significant errors in the estimation result.

3.5.5. Radar Velocity Measurements

Radar sensors provide very accurate estimates on relative velocities. As the speed of the ego-vehicle is known from inertial sensors, the absolute velocity of objects gets observable. In the following, v_{radar} will thus refer to the ego-motion compensated radar velocity measurement of an object. This velocity measurement corresponds to the velocity in direction ψ_{radar} of the radar beam (see Fig. 3.14(a)). It is related to the longitudinal velocity of the vehicle, v, by the following nonlinear measurement model, that depends both on the state vector parameter ψ and the measurements v_{radar} and ψ_{radar}:

$$v = v_{\text{radar}}/\cos(\psi - \psi_{\text{radar}}) \tag{3.56}$$

It is possible to reformulate this relationship by an implicit function $h^{(v)}(x, \boldsymbol{z}_{\text{radar}})$, with $\boldsymbol{z}_{\text{radar}} = [v_{\text{radar}}, \psi_{\text{radar}}]^{\mathsf{T}}$ and

$$\begin{aligned} 0 &= h^{(v)}(\boldsymbol{x}, \boldsymbol{z}_{\text{radar}}) && (3.57) \\ &= v - v_{\text{radar}}/\cos(\psi - \psi_{\text{radar}}) && (3.58) \end{aligned}$$

In an implicit Kalman filter formulation, $h^{(v)}(\boldsymbol{x}, \boldsymbol{z}_{\text{radar}})$ is used as pseudo-innovation [Soatto et al. 1994] in the state update equation

$$\widehat{\boldsymbol{x}}^{+} = \widehat{\boldsymbol{x}}^{-} + K\ h^{(v)}(\widehat{\boldsymbol{x}}^{-}, \boldsymbol{z}_{\text{radar}}) \tag{3.59}$$

If the expected angle between the object orientation and the radar beam is small, it is possible to replace the nonlinear, implicit measurement function by a direct measurement, $v_{\text{radar}} = h^{(v)}(\boldsymbol{x})$, with

$$h^{(v)}(\boldsymbol{x}) = H^{(v)}\boldsymbol{x} = v \tag{3.60}$$

and $H^{(v)} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\ldots 0 \end{bmatrix}$. This simplification is acceptable, for example, if only leading vehicles are tracked that have approximately the same orientation as the ego-vehicle, i.e., $\psi \approx \psi_{\text{radar}}$ in (3.56). The theoretical error in the radar velocity for a given error in orientation is shown in Fig. 3.14(b). For an angular difference of 5 degree, the velocity error is below 0.4%, and for 10 degree angular difference it is still below 1.6%. In practice, the linear version of the radar measurements is used for angular distances up to 10 degree, and it is skipped for larger angular deviations.

3.6. Stochastic Model

In the proposed vehicle tracking approach, both the system and measurement model are nonlinear and only approximations of the unknown real models. Any errors in the system model must be incorporated into the stochastic model. The same holds for errors in the measurement model, as well as the uncertainties of the sensors that provide the measurements. In the following, the basic stochastic assumptions for the present vehicle tracking approach are summarized.

3.6.1. System Noise Model

Errors in state prediction using the system model, as proposed in Sec. 3.4, originated from two main components. First, the dynamic model is nonlinear and only a highly simplified approximation of the complex real vehicle dynamics. Linearization of this model reduces accuracy in many situations. Secondly, prediction errors are induced by errors in reconstruction of the movement of the ego-vehicle. Both errors are modeled as zero-mean, white Gaussian noise processes, since in the absence of any higher order statistics, there is no better form to assume than Gaussian density [Maybeck 1979].

In this approach, errors in the system model are modeled as additive system noise, i.e.,

$$\begin{aligned} \boldsymbol{x}'' &= f(\boldsymbol{x}', \boldsymbol{u}, \boldsymbol{w}) \\ &= f(\boldsymbol{x}', \boldsymbol{u}, \boldsymbol{0}) + \boldsymbol{w} \end{aligned} \tag{3.61}$$

with $\boldsymbol{w} \sim (\boldsymbol{0}, C_{ww})$. In the common formulation of the extended Kalman filter, as introduced in Sec. 2.6.4, the control input vector $\boldsymbol{u}$ is assumed to be error-free or the uncertainties are incorporated into the system noise matrix C_{ww}.

Given the covariance matrix of the control vector C_{uu}, the state covariance matrix update equation in (2.39) extends to

$$C''^{-}_{xx} = AC'^{+}_{xx}A^{\mathsf{T}} + UC_{uu}U^{\mathsf{T}} + C_{ww} \tag{3.62}$$

where A and U are the Jacobian matrices of f w.r.t. $\boldsymbol{x}$ and $\boldsymbol{u}$ respectively.

The system noise matrix C_{ww} then only contains the uncertainties induced by the dynamic model used for state prediction. In practice, C_{ww} is unknown and has to be chosen appropriately as design parameter. Its parameterization controls the influence of the system model compared to the measurements. Small variances in C_{ww} enforce the system model, while larger variances leave more influence to the measurements.

Furthermore, the covariance matrix $C_{xx}(t_0)$ for the initial state at time t_0 estimate must be provided. This is an additional design parameter. The exact choice of the parameter values depends on the application.

3.6.2. Measurement Noise Model

The stochastic measurement model assumes that all measurements are corrupted by at least some amount of additive zero-mean, white Gaussian noise (cf. (2.21) and (2.22) in Sec. 2.6):

$$\boldsymbol{z} \quad = \quad h(\tilde{\boldsymbol{x}}, \boldsymbol{v}) = h(\tilde{\boldsymbol{x}}, \mathbf{0}) + \boldsymbol{v} \tag{3.63}$$

with the true state $\tilde{\boldsymbol{x}}$ to which the true measurements $\tilde{\boldsymbol{z}}$ are related via the measurement model h, i.e., $\tilde{\boldsymbol{z}} = h(\tilde{\boldsymbol{x}}, \mathbf{0})$, and $\boldsymbol{v} \sim (\mathbf{0}, C_{vv})$. The nonsingular measurement noise matrix C_{vv} contains all variances and covariances of the measurements. It is assumed that the individual point measurements, geometric measurements, e.g., for the rotation point, or radar velocity measurements, as proposed in Sec. 3.5, are uncorrelated, and thus

$$C_{vv} = \begin{bmatrix} C^{(1)}_{vv} & \dots & 0 & 0 & 0 \\ \vdots & \ddots & 0 & 0 & 0 \\ 0 & 0 & C^{(M)}_{vv} & 0 & 0 \\ 0 & 0 & 0 & C^{(rot)}_{vv} & 0 \\ 0 & 0 & 0 & 0 & C^{(v)}_{vv} \end{bmatrix} . \tag{3.64}$$

The individual covariance matrices are concretized in the following.

Point Measurements

For each (u_m, v_m, d_m)-measurement of a point $\mathcal{P}_m$, the 3×3 covariance matrix $C^{(m)}_{vv}$ is assumed to be a constant measurement covariance matrix $C^{(P)}_{vv}$, with

$$C^{(P)}_{vv} = \mathrm{Diag}(\sigma_u^2, \sigma_v^2, \sigma_d^2) \tag{3.65}$$

where σ_u^2 and σ_v^2 are the variances of a given feature position, indicating the precision of the used feature tracker, and σ_d^2 the variance of the corresponding stereo disparity measurement.

Theoretically, the uncertainties of the measurements are correlated. The amount of correlation depends on the actual feature tracking approach and stereo algorithm. However, in the absence of a reliable quantity of the covariances, these correlations are also neglected in practice. Thus, any correlations between image position and stereo disparity are not considered in the used noise model, and σ_u^2, σ_v^2, and σ_d^2 are manually selected as system specific design parameters. Instead of assuming uniform variances over the whole image, it is possible to derive individual variances for each pixel position as shown in [Wedel et al. 2009b] for scene flow data. Integration of these values into the noise model is straight forward.

Rotation Point Measurements

The covariance matrix for the rotation point measurements $C_{vv}^{(rot)}$ depends on the approach how the measurements are generated.

Stereo Profile Histograms: The histogram analysis does not directly result in an uncertainty of the estimated rotation point. However, it is inverse proportional to the number of points in the histogram, and proportional to the individual uncertainties of these points in vehicle coordinates, as well as the histogram bin size. An exact error propagation is not straight forward due to strong nonlinearities at the histogram analysis (quantization, different weighing of point entries, etc.). In practice, the covariance matrix of a fictive point at the object center in the vehicle system, rotated to the ego-system, gives a fast and sufficient approximation of the covariance matrix of the estimation results.

Stixel Silhouettes: The measurement uncertainty of the rotation point measurement based on stixels follows from the pose estimation procedure and the resulting covariance matrix of the parameter updates (see (3.55))

$$C_{vv}^{(rot)} = C_{\Delta y \Delta y} = \left(G^{\top} C_{ss}^{-1} G \right)^{-1} . \tag{3.66}$$

The choice of C_{ss} is based on the expected variances of the stixel data in lateral direction and disparity. The latter is lower compared to the expected variance of a single disparity measurement, due to the averaging of disparities within a stixel column, while the uncertainty in lateral direction depends on the stixel width.

Velocity Measurements

The velocity measurements depend on the accuracy of the radar sensor. If the outer variance $\sigma_{v_{radar}}^2$ of the radar velocity is known, it can be directly used as measurement noise, i.e., $C_{vv}^{(vel)} = \sigma_{v_{radar}}^2$. Alternatively it is empirically determined in terms of the inner precision $\widehat{\sigma}_{v_{radar}}^2$, i.e., the variance over a number of velocity measurements of an object moving with constant velocity. For example, the radar sensor used in this thesis has an inner precision of 0.16 m/s (= 0.4 m/s standard deviation).

In the next section it is shown how the measurement noise matrices are further adapted, to increase the robustness of the estimator with respect to outliers in the input data.

3.7. Robust Outlier Detection

In a real system, the measurements used for updating the Kalman filter state contain errors that do not follow a Gaussian random noise process, as assumed in the stochastic model. Such errors can affect the estimation result significantly, if used for updating the filter state. Typical error sources are error-prone feature tracks or disparity measurements, e.g., due to wrong correspondences, or points that are wrongly assigned to a given object, although belonging to the background or another object.

To be able to deal with such *outliers* in the measurements, a two step method is applied. First, gross errors that deviate significantly from the expectation are sorted out. The remaining measurements are added based on the robust Kalman filter extension, as proposed in Sec. 2.6.7, i.e., the measurement covariance matrix is reweighed based on the residual between a measurement and its predicted value.

In the following, the outlier handling is proposed for the point measurements $\boldsymbol{z}^{(p)}$ only. However, the same concepts are applied accordingly to the other measurements such as radar or rotation point measurements.

3.7.1. Gross errors

The following test statistic is used to detect gross errors in the data:

$$\delta_m = \boldsymbol{r}_m^\mathsf{T} {\boldsymbol{C}_{rr}^{(m)}}^{-1} \boldsymbol{r}_m \sim \chi^2_{|\boldsymbol{r}_m|} \tag{3.67}$$

where $\boldsymbol{r}_m = \boldsymbol{z}_m - \widehat{\boldsymbol{z}}_m$ denotes the measurement/prediction residual of the m-th point measurement (cf. Sec. 3.5.2), and $\boldsymbol{C}_{rr}^{(m)}$ represents the corresponding covariance matrix, including both the propagated state covariance matrix $\boldsymbol{C}_{xx}$ and the measurement noise matrix ${\boldsymbol{C}_{vv}}^{(m)}$, with

$$\boldsymbol{C}_{rr}^{(m)} = \boldsymbol{H}_m \boldsymbol{C}_{xx} \boldsymbol{H}_m^\mathsf{T} + \boldsymbol{C}_{vv}^{(m)}. \tag{3.68}$$

This matrix is typically fully occupied. The test statistic is computed for each point individually, as the stochastical model assumes statistical independence. If $\delta_m > T_{\max}$, the point is assumed as *gross error*. Any point detected as gross error is ignored at the filter update step and removed from the object model. There are two different ways to define the threshold $T_{\max}$:

Constant Gating In this variant, the threshold $T_{\max}$ follows from the cumulative distribution function (cdf) of the $\chi^2_{|\boldsymbol{r}_m|}$-distribution with $|\boldsymbol{r}_m|$ degrees of freedom, where $|\boldsymbol{r}_m|$ is the cardinality of $\boldsymbol{r}_m$, i.e., 3 for the point measurements. Let α denote the probability of a valid measurement being erroneously detected as outlier by the outlier test, then $T_{\max} = \mathrm{icdf}_{\chi^2_3}(1-\alpha)$ follows from the inverse cdf. For example, for $\alpha = 0.01$, one yields a threshold of $T_{\max} = 11.3$.

This variant of the threshold selection is also referred to as *constant confidence interval gating* in the remainder of this thesis.

Adaptive Gating At sudden maneuvers, the state prediction by the system model, assuming stationary movement patterns, leads to large deviations between the predicted and the actual measurements. If the filter has matched the previous (stationary) object motion well, the values of the state covariance matrix become quite small and are dominated by the additive system noise matrix. This also influences the covariance matrix of the residual C_{rr}. Accordingly, the resulting normalized residuals become much larger at maneuvering phases.

To prevent that all measurements are considered as outliers as the maneuver starts, an adaptive threshold is introduced. It is designed in a way that at least 50% of all measurements are guaranteed to survive the outlier test, assuming there are no more than 50% outliers among the measurements. The maximum residual T_{max} is then replaced by T'_{max}, defined as

$$T'_{\text{max}} = \max\left(T_{\text{max}}, \text{median}(\delta_1, \ldots, \delta_m, \delta_M)\right) \tag{3.69}$$

This requires that all normalized residuals in (3.68) are computed in advance.

3.7.2. Reweighing of Measurement Noise

All measurements that pass the gross error test are used to update the filter state. However, there still might be outliers in the data that cannot be separated from the valid measurements by the first step.

Thus, the influence of a particular measurement is additionally steered based on the normalized residual by adapting the measurement noise matrix ${\mathsf{C}_{vv}}^{(m)}$, which has been defined as constant for all measurements in Sec. 3.6.2.

For all points with a normalized residual $\sqrt{\delta_m} > C_{\text{Huber}}$, the system noise matrix is adapted as follows

$$\mathsf{C}_{vv}^{(m)*} = \frac{\sqrt{\delta_m}}{C_{\text{Huber}}} \mathsf{C}_{vv}^{(m)} \tag{3.70}$$

where $C_{\text{Huber}} = 1.345$ corresponds to the Huber constant (cf. Sec. 2.6.7). Measurements with a residual below this constant are added with the original measurement noise. As a result, points with a larger deviation from the expectation yield less influence on the estimation results.

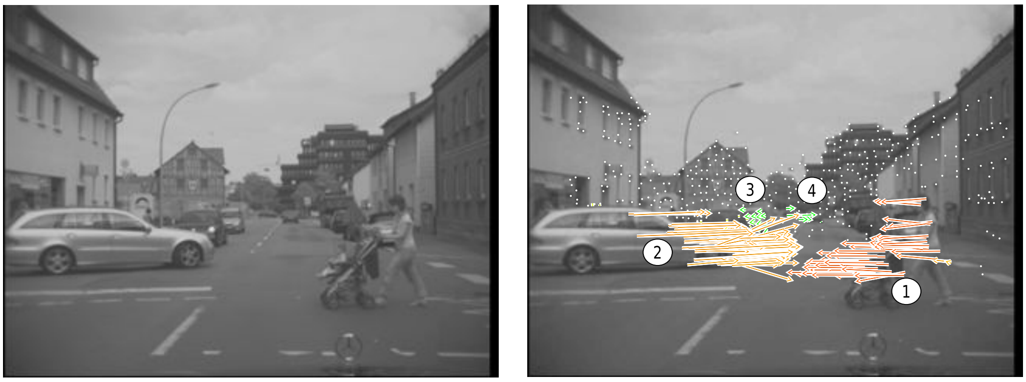
As mentioned above, the outlier detection mechanism as well as the adaptive reweighing of the measurement noise matrix can be applied accordingly to all other uncorrelated groups of observations. The effect of the different outlier detection methods will be addressed in the experimental results in Sec. 4.2.5.

3.8. Initialization

Before the tracking can be started, the filter has to be initialized with pose and motion parameters of an object hypothesis. Two different initialization methods have been realized: One is based on computer vision, while the other uses radar information.

For each generated object hypothesis, both methods yield a filter state $\widehat{\boldsymbol{x}}(k_0)$, with

$$\widehat{\boldsymbol{x}}(k_0) = \left[{}^{e}X_o(k_0), {}^{e}Z_o(k_0), \psi(k_0), v(k_0), 0, 0, 0, 0\right]^{\mathsf{T}}, \tag{3.71}$$



(a)

(b)

Figure 3.15.: 6D-Vision results of a scene with four moving objects: (1) A pedestrian with a stroller walking to the left, (2) A vehicle turning into the street from the left, (3) A slow oncoming vehicle, and (4) a vehicle moving from left to right at a far distance. The arrows indicate the predicted linear point motion for the next half a second. Clustering of neighboring 6D vectors with common motion yields object candidates.

defining the initial transformation between the object system and ego-system and the initial object velocity at discrete initialization time step k_0. The remaining parameters are initialized with zeros.

3.8.1. Image-based Initialization

In this approach, the 6D-Vision motion field is used as input for generating object hypothesis (cf. Sec. 2.4.2).

Fig. 3.15 shows an example 6D-Vision result, taken from a sequence of images with four moving objects. The arrows point to the position where a given point (projected onto the image plane) will be in half a second based on the current filter state. The white dots superimposed indicate stationary points in the scene. Given this data, humans easily group the arrows belonging to the pedestrian with the stroller (1) and the car turning into the street from the right (2). Having a closer look, even the oncoming car (3), approaching slowly, can be detected as well as a vehicle moving from left to right at an intersection ahead (4).

A group of 6D vectors within a local neighborhood with compatible motion indicates an object candidate. Now the task is to automate the object detection by clustering techniques. In this approach a fast histogram based clustering is used, where *compatibility* is defined in terms of the Mahalanobis distance. This is sketched as follows:

1. project all tracked 3D points (in S_e) onto the ground plane and quantize these points to a finite number of grid cells.

2. search for grid cells with significant accumulation of 3D points and select these cells as *seeds*.

3. for each seed
   a) compute the average velocity vector in that cell (seed velocity)
   b) recursively search for points in neighboring cells, which are *compatible* in motion to the seed velocity, and add these points to a candidate list (candidate points ${}^{e}\boldsymbol{P}_1, \ldots, {}^{e}\boldsymbol{P}_M$)
   c) if the number of candidate points M is above a threshold, create a new object hypothesis and initialize the state vector based on the candidate points

The parameters of the state vector $\boldsymbol{x}(k_0)$, as proposed in (3.71), are derived from the candidate point list by the following equations:

$$
\begin{aligned}
{}^{e}X_o(k_0) &= \frac{1}{M}\sum_{m=1}^{M} {}^{e}X_m(k_0) && (3.72)\\
{}^{e}Z_o(k_0) &= \frac{1}{M}\sum_{m=1}^{M} {}^{e}Z_m(k_0) && (3.73)\\
\psi(k_0) &= \arccos\left({}^{e}\bar{v}_z \| {}^{e}\bar{\boldsymbol{v}}\|^{-1}\right) && (3.74)\\
v(k_0) &= \| {}^{e}\bar{\boldsymbol{v}}\| && (3.75)
\end{aligned}
$$

where ${}^{e}\bar{\boldsymbol{v}} = [{}^{e}\bar{v}_x, {}^{e}\bar{v}_y, {}^{e}\bar{v}_z]^{\mathsf{T}}$ corresponds to the average velocity vector over all candidate points, i.e., it is assumed that the average moving direction of all vectors indicates the object orientation.

The candidate points further provide the initial point cloud for the object shape model. Each measured point is transformed into the object coordinate system defined by the initial state vector. The vehicle coordinate system parameters ${}^{o}X_v(k_o)$, ${}^{o}Z_v(k_o)$, and $\beta(k_o)$ are initialized with 0.

If the full model is used (see Sec. 3.3.4), the state vector in (3.71) is extended by the object point coordinates. The state covariance matrix, C_{xx}, is initialized as block diagonal matrix with constant variances for the pose and motion matrices and 3 × 3 covariance matrices for the point positions in object coordinates:

$$
\mathsf{C}_{xx}(t_0) = \mathrm{blkdiag}\left(\sigma^2_{{}^{e}X_o}, \sigma^2_{{}^{e}Z_o}, \sigma^2_{\psi}, \sigma^2_{v}, \sigma^2_{\dot{v}}, \ldots, \underbrace{{}^{o}\mathsf{C}^{(1)}_{PP}, \ldots, {}^{o}\mathsf{C}^{(M)}_{PP}}_{\text{only in full model}}\right). \qquad (3.76)
$$

The covariance matrix ${}^{o}\mathsf{C}^{(m)}_{PP}$ of the m-th points in object coordinates follows from error propagation of the uncertainties of the corresponding image coordinate and disparity.

It is possible to restrict the initialization method to objects exceeding a certain velocity threshold, motion direction, or dimensional constraint. The initial object dimension is optionally computed from the bounding box of the initial point cloud or set to a constant expectation on a typical vehicle size.

One drawback of the image-based initialization method described above is a delay between initialization of 6D vectors and generating an object hypothesis. It takes a few time steps until the Kalman filtered point tracks are reliable for clustering, especially at large distances due to the stereo uncertainty which increases quadratically with

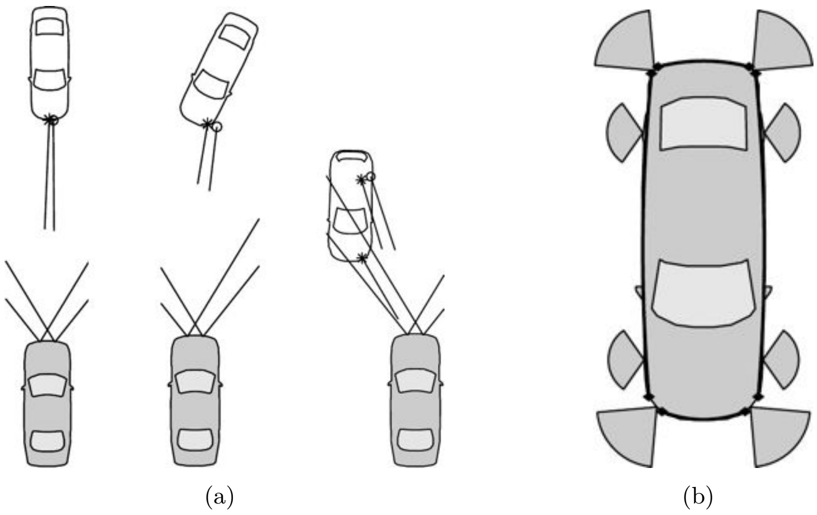


Figure 3.16.: Radar reflectance model of Bühren [Bühren and Yang 2006]. (a) The radar reflection point can originate from different locations of the vehicle (front, corner, wheel house, etc) depending on the object pose. (b) An estimate on the object center is achieved by consideration of 8 point reflection centers, 4 plane reflectors, and a prior on the object pose and dimension.

distance. Depending on the relative velocity and cycle time, an oncoming vehicle can approach up to $10 - 30$ m in this time period. Since the Kalman filter estimating the object's motion state also requires a couple of cycles to converge, the objective should be to initialize object tracking as early as possible.

3.8.2. Radar-based Initialization

The far-range radar sensor, providing a list of detected objects, is used for the alternative initialization method. The actual object detection is done within the sensor and cannot be controlled from outside. The output is a point position and the relative velocity between this point and the radar sensor, which can be transformed into an absolute velocity, given the known ego-motion.

To be able to initialize the pose of a cuboid, a geometric radar reflectance model, proposed by [Bühren and Yang 2006], is utilized. The objective is to predict from which part of the vehicle the radar position originates. The model incorporates eight potential point reflection centers and four plane reflectors (see Fig. 3.16). It is parameterized by the expected object width and length and requires a prior guess on the object pose. The outcome is an estimate of the object center position, ${}^{e}\boldsymbol{P}_{\text{radar}} = [{}^{e}X_{\text{radar}}, 0, {}^{e}Z_{\text{radar}}]^{\mathsf{T}}$, at which the object origin is initialized.

The radar is used to initialize the filter state parameters as follows:

$$
\begin{aligned}
{}^{e}X_o(k_0) &= {}^{e}X_{\text{radar}} & (3.77)\\
{}^{e}Z_o(k_0) &= {}^{e}Z_{\text{radar}} & (3.78)\\
\psi(k_0) &= \psi_{\text{radar}} & (3.79)\\
v(k_0) &= v_{\text{radar}} & (3.80)
\end{aligned}
$$

where v_{radar} represents the absolute radar velocity. The object orientation ψ_{radar} can be derived from a number of previous radar positions using linear regression [Bühren and Yang 2007a]. Initializing the tracker with $\psi_{\text{radar}} = \pi$, i.e., the reversed heading direction of the ego-vehicle, is a fast alternative in practice if only oncoming vehicles are considered.

The radar-based initialization method does not add 3D points to the object shape model, i.e., the object is initialized with an empty point cloud. At runtime, new 3D points that are compatible in position and motion are detected and added to the shape model based on a *data association strategy*. This mechanism is an essential part of the tracking framework ensuring that the system is able to deal with lost point tracks and changing visibility of object parts. It will be introduced in Sec. 3.9.

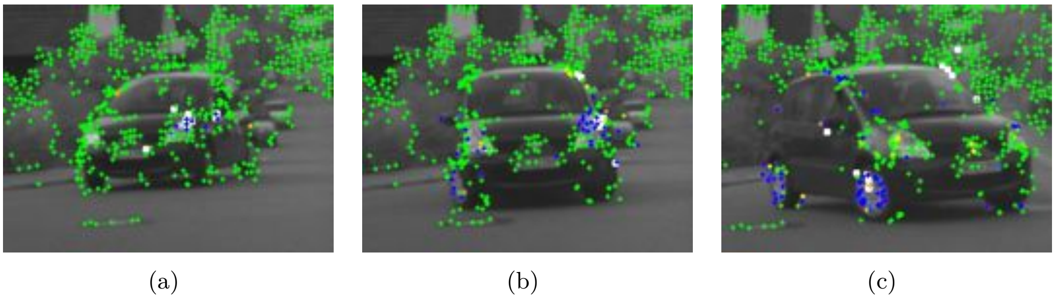
The state vector covariance matrix $\boldsymbol{C}_{xx}(t_0)$ is initialized as proposed in (3.76) (excluding the point positions), however, the actual variances can differ. For example, the initial velocity estimate obtained by the radar is much more precise, i.e., the corresponding variance for the velocity component can be chosen significantly smaller compared to the vision based approach.

3.9. Data Association

The more object points are contributing to the measurement model, the more robust is the state estimation. However, feature tracks get lost at runtime due to varying conditions in the scene, for example illumination changes, or (partly) occlusions by other objects. There are also self occlusions, e.g., at turn maneuvers with changing visibility of object sides as can be exemplarily seen in Fig. 3.17. If a given feature cannot be reassigned in the next frame, the corresponding 3D point in the shape model does not have a valid measurement and, thus, cannot be considered to update the Kalman filter state. Since lost feature tracks are compensated by new feature tracks that are permanently initialized all over the image plane until a maximum number of tracks is reached, the objective is to assign these new tracks to existing objects.

The decision whether an feature belongs to an existing object or to the (static) background is formulated in a probabilistic hypotheses testing approach instead of applying a threshold based *if-then-else* reasoning.

The problem can be formalized as follows. Let $\mathcal{L}_{\boldsymbol{O}} = \{\boldsymbol{O}_0, \boldsymbol{O}_1, \ldots, \boldsymbol{O}_J, \boldsymbol{O}_{J+1}\}$ denote the finite set of candidate objects, where the objects with index 1 to J correspond to tracked objects and $\boldsymbol{O}_0$ represents the static background hypothesis. To be able to also deal with non-static background points, or points corresponding to independently moving objects that are currently not tracked, the additional hypothesis $\boldsymbol{O}_{J+1}$ is introduced.



(a) (b) (c)

Figure 3.17.: As the vehicle turns, some parts get occluded while others become visible. Accordingly, feature tracks get lost or are initialized at runtime. The dot color encodes the track length (green=VALID for ≥ 15 frames, blue=VALID for ≤ 5 frames, white=LOST). The tracking system must be able to deal with a varying number of features on the object.

For each candidate feature, currently not assigned to any object, the conditional probability $p(\boldsymbol{O} = \boldsymbol{O}_j|\boldsymbol{y}_i)$ is computed, i.e., the probability that the i-th feature corresponds to the j-th object in $\mathcal{L}_{\boldsymbol{O}}$ given the data vector $\boldsymbol{y}_i$. The i-th feature is then assigned to the object with index ι, which is computed as

$$\iota = \arg\max_j p(\boldsymbol{O} = \boldsymbol{O}_j|\boldsymbol{y}_i) \quad , 0 \leq j \leq J+1. \tag{3.81}$$

This means the feature is assigned to the object to which it belongs most probably. If $\iota \in [1, J]$, the feature is actually added to the shape model of $\boldsymbol{O}_\iota$. In practice it is possible to restrict the number of points contributing to the shape model, i.e., new points are only added if the maximum number of points is not exceeded yet.

Since $p(\boldsymbol{O} = \boldsymbol{O}_j|\boldsymbol{y}_i)$ is actually unknown, the Bayesian rule is applied, yielding

$$p(\boldsymbol{O} = \boldsymbol{O}_j|\boldsymbol{y}_i) = \frac{p(\boldsymbol{y}_i|\boldsymbol{O} = \boldsymbol{O}_j)p(\boldsymbol{O} = \boldsymbol{O}_j)}{p(\boldsymbol{y}_i)}. \tag{3.82}$$

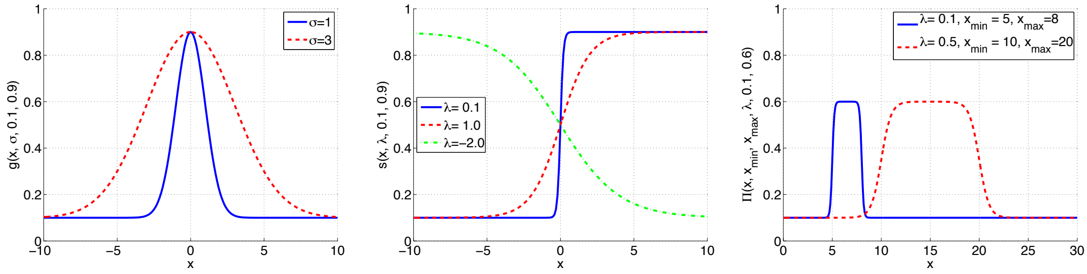
Here, $p(\boldsymbol{y}_i) = \sum_{j=0}^{J+1} p(\boldsymbol{y}_i|\boldsymbol{O} = \boldsymbol{O}_j)p(\boldsymbol{O} = \boldsymbol{O}_j)$ is a constant for all objects that follows from the total probability theorem [Bishop 2006] and, thus, does not influence the result in (3.81). Assuming the data vector contains N mutually independent subvectors $\boldsymbol{y}_{i,n}$, one can write the likelihood $p(\boldsymbol{y}_i|\boldsymbol{O} = \boldsymbol{O}_j)$ as

$$p(\boldsymbol{y}_i|\boldsymbol{O} = \boldsymbol{O}_j) = p(\boldsymbol{y}_{i,1}, \ldots, \boldsymbol{y}_{i,N}|\boldsymbol{O} = \boldsymbol{O}_j) = \prod_{n=1}^{N} p(\boldsymbol{y}_{i,n}|\boldsymbol{O} = \boldsymbol{O}_j). \tag{3.83}$$

Let $l_{i,j}^{(\mathrm{n})}$ abbreviate the individual likelihood terms, i.e., $l_{i,j}^{(\mathrm{n})} = p(\boldsymbol{y}_{i,n}|\boldsymbol{O} = \boldsymbol{O}_j)$. Then, the conditional probability $p(\boldsymbol{O} = \boldsymbol{O}_j|\boldsymbol{y}_i)$ is proportional to $p(\boldsymbol{O} = \boldsymbol{O}_j)\prod_{n=1}^{N} l_{i,j}^{(\mathrm{n})}$.

The result in (3.81) is equal to

$$\iota = \arg\max_j \left(\ln p(\boldsymbol{O} = \boldsymbol{O}_j) + \sum_{n=1}^{N} \ln(l_{i,j}^{(\mathrm{n})}) \right) \tag{3.84}$$



(a) gaussian

(b) sigmoidal

(c) Π-shaped

Figure 3.18.: Base functions used for defining the potentials.

where the product has further been replaced by the sum of log-likelihoods. The class prior $p(\boldsymbol{O} = \boldsymbol{O}_j)$ can also be ignored, if it is constant for all classes.

The actual likelihood functions $l_{i,j}^{(\mathrm{n})}$ are still unknown, however, this formulation allows for learning these functions from training data, or an intuitive manual modeling, which will be pursued in this approach.

In the following sections, concrete manually designed realizations of the likelihood functions are given for $N = 4$ data inputs $\boldsymbol{y}_{i,n}$, including the point position with respect to a given object pose prior (*region of interest likelihood*, *distance likelihood*), its height over ground (*height likelihood*), as well as its compatibility in motion with respect to the expected motion of a given object hypothesis (*motion likelihood*).

These likelihood functions are defined based on three base functions which will be briefly introduced first.

3.9.1. Likelihood Base Functions

The single likelihood functions $l_{i,j}^{(\mathrm{n})}$ are defined based on different parametrization of three basic functions scaled to the range $\boldsymbol{\kappa} = [\kappa_{\min}, \kappa_{\max}]$ (see Fig. 3.18).

Gaussian: A bell-shaped, zero-mean, multi-dimensional Gaussian function g with covariance matrix C_x, defined as

$$g(\boldsymbol{x}, \mathsf{C}_x, \kappa_{\min}, \kappa_{\max}) = (\kappa_{\max} - \kappa_{\min}) \exp\left(-1/2\ \boldsymbol{x}^\mathsf{T} \mathsf{C}_x^{-1} \boldsymbol{x}\right) + \kappa_{\min} \tag{3.85}$$

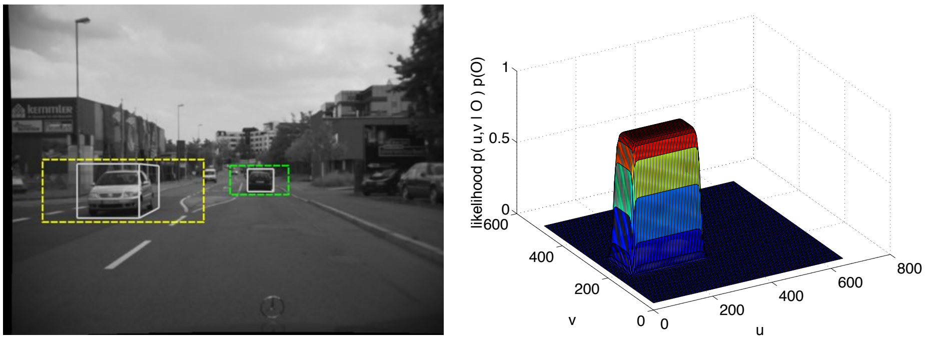
The function is scaled in a way that its maximum is $\kappa_{\max}$ and it converges towards a minimum value of $\kappa_{\min}$. For $\kappa_{\max} = |2\pi \mathsf{C}_x|^{-1/2}$ and $\kappa_{\min} = 0$ it corresponds to a normal distribution.

Sigmoidal: A one-dimensional sigmoidal function s with width λ and turning point at $x = 0$, scaled to the range $\boldsymbol{\kappa}$ with

$$s(x, \lambda, \kappa_{\min}, \kappa_{\max}) = (\kappa_{\max} - \kappa_{\min})/(1 + \exp(-x/\lambda)) + \kappa_{\min}. \tag{3.86}$$

Π-shaped: A gating function Π that is composed of two opposite sigmoidal functions with slope λ

$$\begin{aligned}\Pi(x, x_{\min}, x_{\max}, \lambda, \kappa_{\min}, \kappa_{\max}) =& (\kappa_{\max} - \kappa_{\min})\,(s(x - x_{\min}, \lambda, 0, 1)\\ &-s(x - x_{\max}, \lambda, 0, 1)) + \kappa_{\min}\end{aligned} \tag{3.87}$$



(a)

(b)

Figure 3.19.: ROI likelihood. The bounding boxes and scaled ROIs of the two detected cars are superimposed in (a), the corresponding ROI likelihood function is shown in (b). Features that lie outside the ROI are very unlikely to belong to the given object.

It has its maximum value $\kappa_{\max}$ within $x_{\min}$ and $x_{\max}$, respectively, and converges towards $\kappa_{\min}$ outside this range.

To limit the number of parameters, $\kappa_{\min}$ and $\kappa_{\max}$ will be assigned to one of three basic likelihood levels $\kappa_{\mathtt{VL}}$, $\kappa_{\mathtt{UL}}$, and $\kappa_{\mathtt{DK}}$ for *very likely*, *unlikely*, and *don't know*. Each level can be increased by the constant offset $\kappa_{\mathtt{SP}}$ to be able to *slightly prefer* a given class (notation: $\kappa^{+}_{\mathtt{XX}} = \kappa_{\mathtt{XX}} + \kappa_{\mathtt{SP}}$).

The actual choice of the likelihood levels is a design parameter. Throughout this thesis these parameters are chosen as follows: $\kappa_{\mathtt{VL}} = 0.9$, $\kappa_{\mathtt{UL}} = 0.1$, $\kappa_{\mathtt{DK}} = 0.5$, and $\kappa_{\mathtt{SP}} = 0.1$.

3.9.2. Region of Interest Likelihood

A feature is likely to belong to an object, if its current image position $(u_i(k), v_i(k))$ is near the expected position of the object and it is very unlikely if it is outside the expectation.

For each object $\boldsymbol{O}_j$ in $\mathcal{L}_{\boldsymbol{O}}$, the function $\mathrm{ROI}(\boldsymbol{O}_j)$ computes an image region of interest likely to include the given object. It is parametrized by the minimum and maximum image corner $u_{\min,j}$, $u_{\max,j}$, $v_{\min,j}$, and $v_{\max,j}$ respectively.

For all tracked objects, i.e., $j \in [1, J]$, the ROI is computed by projecting the cuboid model onto the image plane and determining the surrounding rectangle (aligned with the image coordinate axis). The cuboid pose is determined by the predicted object state, i.e., after applying the filter system model, but before incorporating the measurement update. This allows for adding new points before the update. In practice the resulting ROI is additionally scaled by a certain amount to ensure the object is really included in the ROI to compensate for uncertainties in the object dimension and pose. The ROI of the background models $\boldsymbol{O}_0$ and $\boldsymbol{O}_{J+1}$ is defined over the full image.

The likelihood function for $l^{(ROI)}_{i,j} = p(\boldsymbol{y}^{(ROI)}_{i,1} | \boldsymbol{O} = \boldsymbol{O}_j)$ for a given measurement

$\boldsymbol{y}_{i,1}^{(ROI)} = [u_i, v_i]^\mathsf{T}$ is modeled as follows

$$\begin{aligned} l_{i,j}^{(ROI)} =&(\kappa_{\mathtt{VL}} - \kappa_{\mathtt{UL}}) \\ &\Pi(u_i, u_{\min,j}, u_{\max,j}, \lambda^{(ROI)}, 0, 1) \\ &\Pi(v_i, v_{\min,j}, v_{\max,j}, \lambda^{(ROI)}, 0, 1) + \kappa_{\mathtt{UL}} \end{aligned} \tag{3.88}$$

where $\lambda_1^{(ROI)}$ steers the steepness of the transition at the ROI boundaries. A fast approximation of this function in practice is given below:

$$l_{i,j}^{(ROI)} \approx \begin{cases} \kappa_{\mathtt{VL}} & , (u_i, v_i) \in \mathrm{ROI}(\boldsymbol{O}_j) \\ \kappa_{\mathtt{UL}} & , \text{otherwise} \end{cases} \tag{3.89}$$

Fig. 3.19(a) shows an example scene with an oncoming and a leading car. The object hypotheses are superimposed by the gray bounding boxes, the scaled ROIs are visualized by the dashed rectangles. The corresponding ROI likelihood function for the oncoming vehicle is shown in Figure 3.19(b).

The design of this function is not very precise in terms of indicating that a given pixel belongs to an object. However, it gives a strong evidence that a feature does not correspond to an object and, thus, can be used to exclude features quite early to safe computation time.

3.9.3. Disparity Likelihood

The predicted object pose also provides an expectation on the minimum and maximum distance an object point might have. The expected distance range for $\boldsymbol{O}_j$ is transformed into a corresponding disparity range $[d_{\min,j}, d_{\max,j}]$. Stereo disparities within this range are much more likely to belong to the given object than points outside this range. For all object hypothesis $\boldsymbol{O}_j$, $j \in [1, J]$, the disparity likelihood $l_{i,j}^{(disp)} = p(\boldsymbol{y}_{i,2}^{(disp)} | \boldsymbol{O} = \boldsymbol{O}_j)$ for a given measurement $\boldsymbol{y}_{i,2}^{(disp)} = d_i$ is defined as

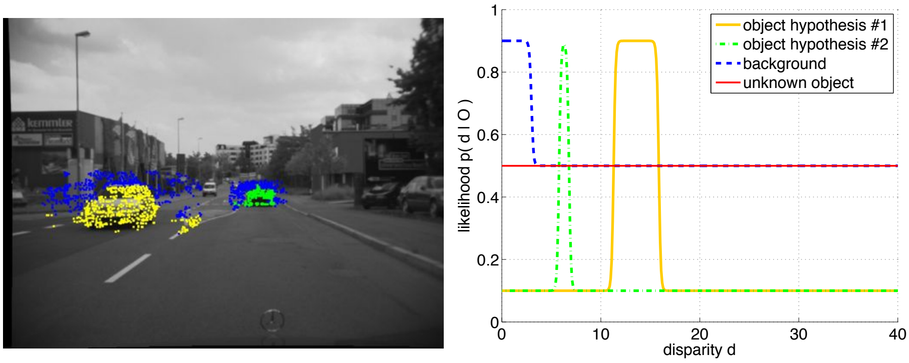
$$l_{i,j}^{(disp)} = \Pi(d_i, d_{\min,j}, d_{\max,j}, \lambda^{(disp)}, \kappa_{\mathtt{UL}}, \kappa_{\mathtt{VL}}), \quad j \in [1, J] \tag{3.90}$$

with $\lambda^{(disp)} = 0.1$ in the default parametrization.

For the background hypothesis $\boldsymbol{O}_0$, a different likelihood function is used. Theoretically, all disparities are equally likely to correspond to background. However, small disparities of only a few pixels, corresponding to really far points in the scene, are very uncertain and, thus, should not be assigned to an object with the same likelihood than closer points in practice. Therefore, it is assumed that all points below a minimum disparity value $d_{\min,0}$ are more likely to be background. This is expressed by the following equation:

$$l_{i,0}^{(disp)} = s(d_i - d_{\min,0}, \lambda_0^{(disp)}, \kappa_{\mathtt{DK}}, \kappa_{\mathtt{VL}}) \tag{3.91}$$

with $\lambda_0^{(disp)} = -1$, and $d_{\min,0} = 3$ in the default parameterization. This configuration means that disparities below 3 pixels (> 80 m) are very likely to belong to the background while larger disparities do not contain information on whether they belong to the background or foreground, i.e., they are equally likely.



(a)

(b)

Figure 3.20.: Disparity likelihood. In (a) the features within both ROIs are superimposed. The color encodes the maximum likelihood hypothesis based on disparity only, i.e., yellow = hypothesis 1 (oncoming car), green = hypothesis 2 (leading car), blue = background. The corresponding disparity likelihood functions are shown in (b).

There is no expectation on the disparity range for the unknown object hypothesis $\boldsymbol{O}_{J+1}$. It differs from the static background hypothesis only based on the motion. Thus, $l_{i,J+1}^{(disp)} = \kappa_{\mathtt{DK}})$ is assumed, i.e., the disparity does not contribute to the unknown object hypothesis.

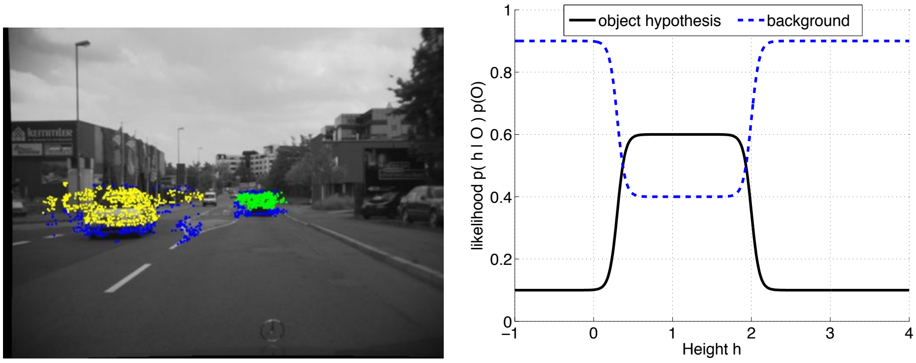
The effect of the disparity likelihood for the two objects in the example scene is shown in Fig. 3.20. The color encodes the maximum disparity likelihood of $l_{i,j}^{(disp)}$ for all hypotheses $j \in [0, J+1]$. Blue crosses mark background features, yellow and green features the oncoming (hypothesis 1) and leading (hypothesis 2) vehicle respectively. Most points on the objects are classified correctly by the disparity ranges. However, without any information on height and motion there are also points on the street that are classified as object pixels, since they have the same disparity. The leading object is approximately 40 m away from the ego-vehicle, the oncoming vehicle about 20 m in this example. Thus, the disparity range of object hypothesis 1 is significantly larger compared to hypothesis 2 due to the nonlinear mapping between distances and disparities.

3.9.4. Height Likelihood

Road vehicles have a particular height over ground, i.e., very low and very high points are more likely to belong to the background. In the range of about 0.3 to 2 m height background objects are not distinguishable from vehicles. This is expressed by the following likelihood function $l_{i,j}^{(height)} = p(\boldsymbol{y}_{i,3}^{(height)} | \boldsymbol{O} = \boldsymbol{O}_j)$ for the observed data $\boldsymbol{y}_{i,3}^{(height)} = h_i$, i.e., the height of the i-th feature:

$$l_{i,j}^{(height)} = \begin{cases} \Pi(h_i, h_{\min,j}, h_{\max,j}, -\lambda^{(height)}, \kappa_{\mathtt{DK}} - \kappa_{\mathtt{SP}}, \kappa_{\mathtt{VL}}) & , j = 0 \\ \Pi(h_i, h_{\min,j}, h_{\max,j}, \lambda^{(height)}, \kappa_{\mathtt{UL}}, \kappa_{\mathtt{DK}} + \kappa_{\mathtt{SP}}) & , j > 0 \end{cases} \tag{3.92}$$

with $\lambda^{(height)} = 1/20$ in the default parameterization.



(a)

(b)

Figure 3.21.: Height likelihood. (a) Blue features are very unlikely to belong to an object, since they are either too low or too high. Yellow and green features are within the expected height range of 0.3 − 2 m. The corresponding likelihood functions are shown in (b).

Fig. 3.21(b) visualizes the height likelihood functions with the default parameters. Blue crosses in Figure 3.21(a) mark features that are outside the expected height range, yellow and green crosses indicate features that have the expected height and, thus, are potential object features. For visualization, the object features have been colored in yellow or green based on the ROI information in this example. The height does not contain any information on which object the feature belongs to in this context (the unknown object hypothesis would be equally likely). As can be seen, there are also many features at far distance that have the expected object height.

3.9.5. Motion Likelihood

A point is *compatible* in motion with a particular hypothesis if its predicted image position and disparity, computed from a corresponding motion model, is close to the measured position and disparity. Let $\mathit{\Pi}_j$[1] denote the transformation function that, given the feature position $\boldsymbol{p}'_i = [u'_i, v'_i, d'_i]^\mathsf{T}$ at discrete time step $k-1$, predicts the current feature position $\boldsymbol{p}''_i = [u''_i, v''_i, d''_i]^\mathsf{T}$ based on the object hypothesis $\boldsymbol{O}_j$ (including the object pose and motion model) and the ego-motion parameters in vector $\boldsymbol{u}$, i.e.,

$$\widehat{\boldsymbol{p}}''_{i,j} = [\widehat{u}''_{i,j}, \widehat{v}''_{i,j}, \widehat{d}''_{i,j}]^\mathsf{T} = \mathit{\Pi}_j(\boldsymbol{p}'_i, \boldsymbol{O}_j, \boldsymbol{u}). \tag{3.93}$$

For the static background hypothesis, the prediction function $\mathit{\Pi}_0$ compensates for the ego-motion only. Each feature at time step $k-1$ is first transformed into ego-coordinates. The resulting 3D position is then transformed using the current ego-motion matrix analog to (3.36) and back-projected onto the current image plane.

For any object hypothesis $\boldsymbol{O}_j$, $j \in [1, J]$, $\mathit{\Pi}_j$ first transforms $\boldsymbol{p}'_i$ to ${}^{o}\boldsymbol{P}_i$ in the object coordinate system, which is defined by the a posteriori object pose parameters $\boldsymbol{\Omega}^+_j(k-1)$ derived from the Kalman filter state $\boldsymbol{x}^+(k-1)$. Then, the a priori object pose $\boldsymbol{\Omega}^-_j(k)$

[1] Not to be confused with the gate likelihood function Π.

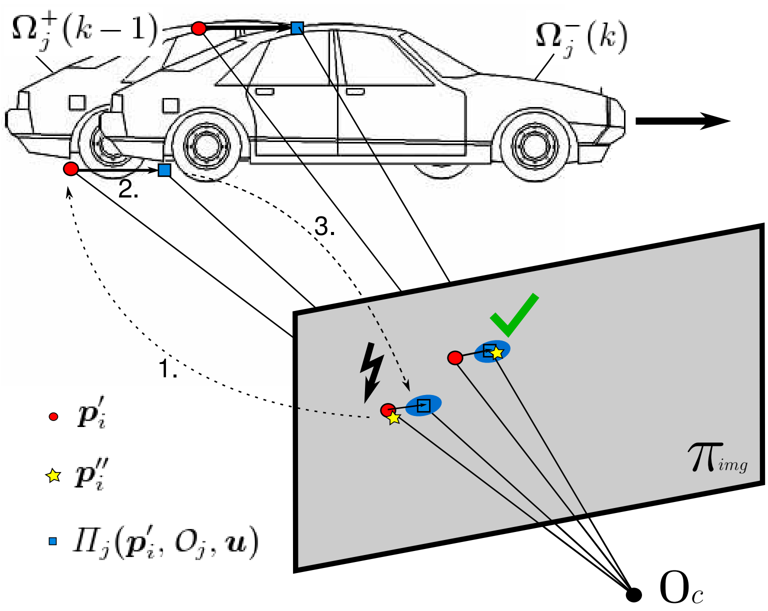

$\boldsymbol{\Omega}_j^+(k-1)$
$\boldsymbol{\Omega}_j^-(k)$
2.
3.
1.
$\boldsymbol{p}'_i$
$\boldsymbol{p}''_i$
$\Pi_j(\boldsymbol{p}'_i, O_j, \boldsymbol{u})$
π_{img}
O_c

Figure 3.22.: Each feature $\boldsymbol{p}'_i$, measured at time step $k-1$, is registered to the object system, defined by $\boldsymbol{\Omega}_j^+(k-1)$. The closer the measured position $\boldsymbol{p}''_i$ at time k is to the reprojected object point position, $\Pi_j(\boldsymbol{p}'_i, \boldsymbol{O}_j, \boldsymbol{u})$, the more likely does the point belong to the object.

is predicted using the object motion model f_j, the a posteriori motion parameters $\boldsymbol{\Phi}_j(k-1)^+$ from $\boldsymbol{x}^+(k-1)$, and the ego-motion information $\boldsymbol{u}$ (see Sec. 3.4 for details).

Finally, the object point ${}^o\boldsymbol{P}_i$ is transformed into ego-coordinates, using the parameters of $\boldsymbol{\Omega}_j^-(k)$, and projected onto the image plane. Thus, the function computes the current image position of a given feature, assuming it has a fixed position on the (moving) object between the previous and the current time step.

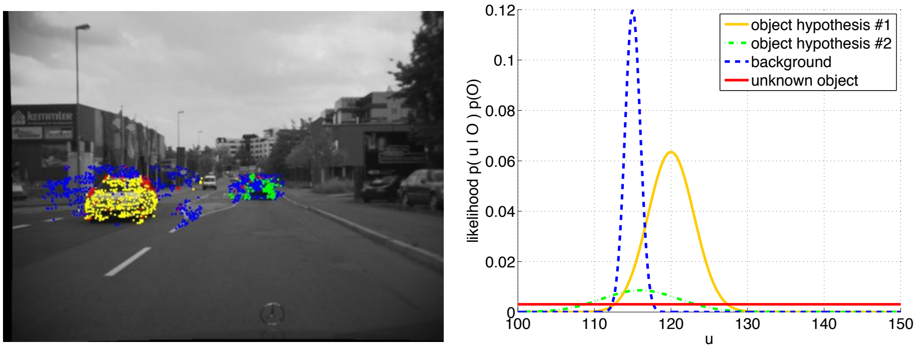
Given the predicted image position, the following data likelihood function for the data vector $\boldsymbol{y}_{i,4}^{(motion)} = [(\boldsymbol{p}'_i)^\mathsf{T}\ (\boldsymbol{p}''_i)^\mathsf{T}]^\mathsf{T} = [u'_i, v'_i, d'_i, u''_i, v''_i, d''_i]^\mathsf{T}$, containing the previous and current measured feature position, is defined based on a standard normal distribution:

$$p(\boldsymbol{y}_{i,4}^{(motion)}|\boldsymbol{O}=\boldsymbol{O}_j) = g\left(\boldsymbol{\Delta p}_{i,j}, C_{\Delta\Delta}^{(i,j)}, 0, |2\pi C_x|^{-1/2}\right) \tag{3.94}$$

with $\boldsymbol{\Delta p}_{i,j} = \Pi_j(\boldsymbol{p}'_i, \boldsymbol{O}_j, \boldsymbol{u}) - \boldsymbol{p}''_i$, i.e., the residual vector between the predicted and the measured feature positions. The 3×3 matrix $C_{\Delta\Delta}^{(i,j)}$ corresponds to the combined covariance matrix of the residual:

$$C_{\Delta\Delta}^{(i,j)} = \Upsilon_j \begin{bmatrix} C_{\boldsymbol{p}'_i\boldsymbol{p}'_i} & 0 & 0 & 0 \\ 0 & C_{\boldsymbol{\Omega}'^+_j\boldsymbol{\Omega}'^+_j} & 0 & 0 \\ 0 & 0 & C_{\boldsymbol{\Omega}''^-_j\boldsymbol{\Omega}''^-_j} & 0 \\ 0 & 0 & 0 & C_{uu} \end{bmatrix} \Upsilon_j^\mathsf{T} + C_{\boldsymbol{p}''_i\boldsymbol{p}''_i} \tag{3.95}$$

where $\Upsilon_j = \left[\frac{\partial \boldsymbol{\Pi}_j}{\partial \boldsymbol{p}'_i}, \frac{\partial \boldsymbol{\Pi}_j}{\partial \boldsymbol{\Omega}'^+_j}, \frac{\partial \boldsymbol{\Pi}_j}{\partial \boldsymbol{\Omega}''^-_j}, \frac{\partial \boldsymbol{\Pi}_j}{\partial \boldsymbol{u}}\right]$ corresponds to the Jacobian of Π_j with respect to the input parameters. The 3×3 covariance matrices $C_{\boldsymbol{p}'_i\boldsymbol{p}'_i}$ and $C_{\boldsymbol{p}''_i\boldsymbol{p}''_i}$ incorporate



(a)

(b)

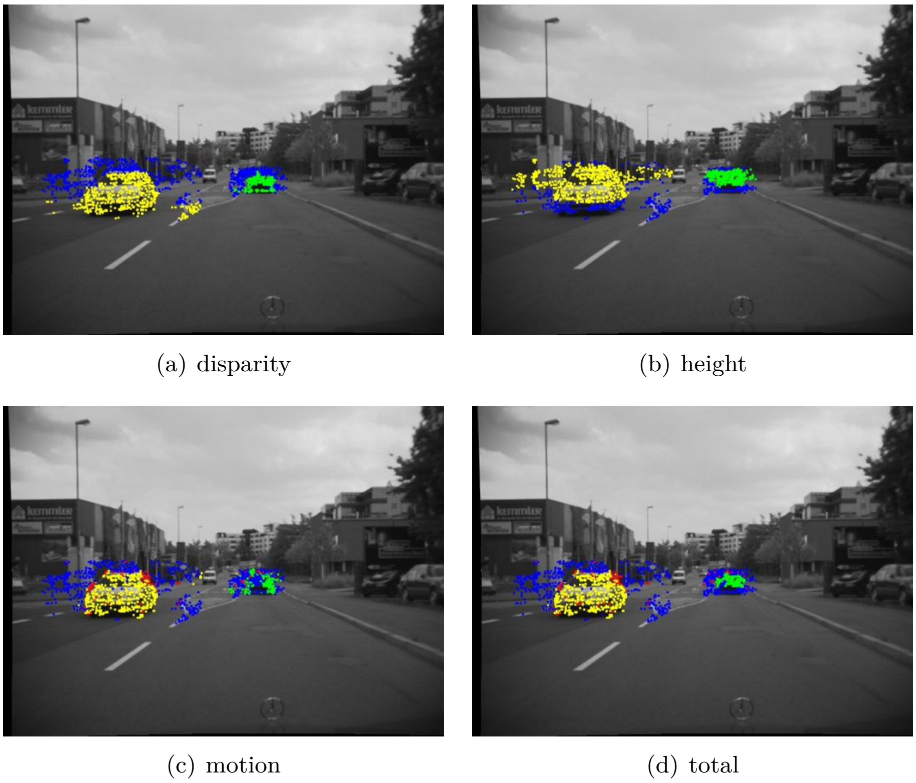
Figure 3.23.: (a) Motion likelihood for the two object hypotheses (yellow and green), background (blue), and the unknown object class (red). (b) Fictive example of the likelihood functions for the horizontal image coordinate u.

the uncertainty of the actual feature measurements. A constant measurement noise is assumed if not other stated, i.e., $\boldsymbol{C}_{\boldsymbol{p}'_i \boldsymbol{p}'_i} = \boldsymbol{C}_{\boldsymbol{p}''_i \boldsymbol{p}''_i} = \text{Diag}(\sigma_u^2, \sigma_v^2, \sigma_d^2)$ analog to (3.65). The 3×3 covariance matrix $\boldsymbol{C}_{\boldsymbol{\Omega}'^+_j \boldsymbol{\Omega}'^+_j}$ follows from the a posteriori covariance matrix $\boldsymbol{C}^+_{\widetilde{x}\widetilde{x}}(k-1)$ of the previous time step, including the uncertainties of the entries specifying the object pose, i.e., ${}^eX_o$, ${}^eZ_o$, and ψ. The a priori covariance matrix $\boldsymbol{C}_{\boldsymbol{\Omega}''^-_j \boldsymbol{\Omega}''^-_j}$ covers the uncertainties of the predicted pose and is extracted from $\boldsymbol{C}^-_{\widetilde{x}\widetilde{x}}(k)$. The computation of the latter also incorporates the uncertainties of the motion parameters, as proposed in (3.62). If the uncertainty of the ego-motion is known, it is represented by $\boldsymbol{C}_{uu}$.

If the residual falls outside a given confidence interval, the feature is very likely to not belong to the given hypothesis. Thus, the likelihood function of the unknown class, $p(\boldsymbol{y}_{i,4}^{(motion)}|\boldsymbol{O} = \boldsymbol{O}_{J+1})$, is defined as a (small) constant that corresponds to the value of the standard normal distribution at the boundaries of the confidence interval. Here, a 98% confidence interval is considered, i.e., $p(\boldsymbol{y}_{i,4}^{(motion)}|\boldsymbol{O} = \boldsymbol{O}_{J+1}) = 0.0267$.

In Fig. 3.23(a) the motion likelihood results are shown. The color encodes the maximum likelihood hypothesis for a given feature. Blue crosses mark features that yield the smallest (normalized) residual between predicted and measured position for the background hypothesis, i.e., the points follow the ego-motion. Yellow crosses indicate features that are compatible in motion with the oncoming vehicle. Note that there are also a few yellow crosses at the second oncoming car and on the shadow of this car on the road at far distance. Features that move according to the leading vehicle are marked as green, while red crosses correspond to features that fall into the unknown object class. The effect of the different likelihood functions for a fictive pixel is demonstrated in Fig. 3.23(b). For simplicity, only the u component is considered here.

The likelihood results are summarized in Fig. 3.24. The combined likelihood according to (3.84), i.e., the sum of the log likelihoods of the ROI, disparity, height, and motion likelihood is shown in (d). As can be seen, the different likelihood terms yield partly orthogonal information. For example, features on the road that have the same disparity



(a) disparity

(b) height

(c) motion

(d) total

Figure 3.24.: Summary of the different independent likelihood components (a) - (c). The combined likelihood, according to (3.84), is shown in (d). All yellow points are added to the shape model of the oncoming object, all green points to the leading vehicle, respectively.

than the object, are contradictory to the height likelihood. On the other hand, features that potentially belong to an object based on the height or motion likelihood, are very unlikely based on the disparity likelihood. The combination ensures that only features that are consistent in all likelihood terms are finally added to the object.

In this example, the likelihood terms have been computed for all tracked image features within the two object ROIs. In practice, only those points that are currently not assigned to any object have to be considered. This reduces the computational load significantly. Points that are wrongly added to the object model by the likelihood-based data association mechanism, have to be detected and rejected by the outlier detection mechanism of the measurement model.

3.10. Object Verification

Once an object track has been initialized, there must be also a mechanism to delete it, e.g., if the object leaves the visual field of the camera, the measurements do not support the object hypothesis any longer, or the estimated parameters are simply implausible. Furthermore, large objects, such as trucks or buses, are likely to generate multiple object hypotheses in practice, if assuming a typical car dimension of about 4.5 m length and 1.7 m width at initialization. The same problem occurs with single cars if detected at far distance, where the stereo point cloud spreads over a large range.

This section briefly proposes the concepts of how object tracks are invalidated and how close-by object hypotheses are merged.

3.10.1. Object Removal

One common confidence measure for Kalman filtering approaches is the *normalized innovation squared* (NIS) measure as proposed in (2.47), indicating how good the current measurements fit the prediction based on the current state and covariance estimate. This scalar measure could be directly used for object verification by means of a threshold test. A small NIS proves a good fit of the current state estimate with respect to the current observations and confirms the object track. As soon as the NIS exceeds a certain threshold, the object is invalidated.

The threshold is derived from the cumulative distribution function (cdf) of the χ^2-distribution with M degrees of freedom (one-sided test), since assuming that the residuals are normal distributed, the sum of squares of the residuals follows a χ^2_M-distribution, with M the number of valid measurements.

In practice, there are two main drawbacks. First, a few (non-detected) outliers in the measurements as well as errors in the estimated parameters can have a strong influence on the resulting NIS measure, e.g., if the variance for a given parameter is significantly underestimated or the tracked object enters a maneuvering phase. Thus, a large NIS does not automatically mean the tracking result is bad. Secondly, if many points are contributing, the computation of the NIS measure becomes computationally expensive.

In this approach, an alternative confidence measure is used. The number of points in the shape model used to update the Kalman filter, i.e., points that have passed the outlier test independently, is a strong indicator for the quality of the tracking results. The more points are contributing, the more likely is the current object existing.

The minimum number of points that are sufficient to *confirm* an object track depends on the distance. At large distances, the threshold must be lower compared to close objects, since there are less features. The actual choice of the threshold function is a design parameter of the overall system. If not other stated, a linear ramp is used that requires 5 valid points at 60 m and 30 valid points at 10 m distance.

The lifetime of an object is modeled as a finite state machine as shown in Fig. 3.25.

After initialization, the object gets the status `HYPOTHESIS`. If there are sufficient measurements supporting the object hypothesis, the object state changes to `CONFIRMED`. As long as there are enough valid measurements, the object remains in the state `CONFIRMED`, otherwise it changes to the state `EXTRAPOLATED`. At this state, the object state is based on the prediction only, i.e., the object movement is extrapolated using the vehicle motion model. Short periods without measurements occur in practice, for example, if all feature tracks are lost at once and have to be reinitialized due to strong illumination changes or partial occlusions. As soon as there are again sufficient measurements supporting the object, the object changes to state `CONFIRMED` again. Otherwise, a counter is increased indicating how many frames in a row the object has been extrapolated. As soon as a maximum number of frames in the extrapolated state is exceeded, the object

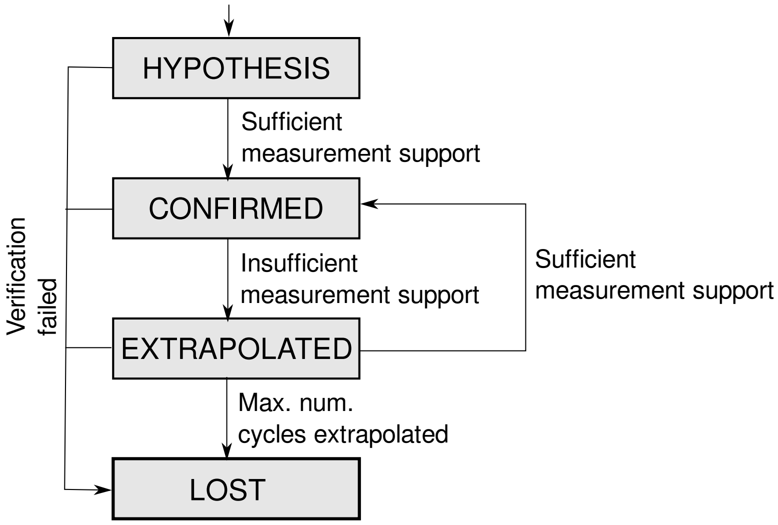

HYPOTHESIS
Sufficient measurement support
CONFIRMED
Insufficient measurement support
EXTRAPOLATED
Sufficient measurement support
Max. num. cycles extrapolated
LOST
Verification failed

Figure 3.25.: Finite state machine of object states

is invalidated by changing to state `LOST`. Lost objects are removed from the internal object list before the next tracking cycle starts.

Beside the valid point count there are several fast checks testing the plausibility of the current state estimate to be able to sort out false positive objects quickly. Such test include, for example, a size check (object dimension within expected range and ratio?), a motion check (velocity or yaw rate physically possible?), distance check (too far objects can be rejected), and a height check (extremely flat objects where all points lie on the ground plane must be incorrect). Each of these criteria is a necessary but not sufficient condition to confirm the object. If one test fails, the object directly changes to the state `LOST`. Fig. 3.26 shows three examples of implausible object states.

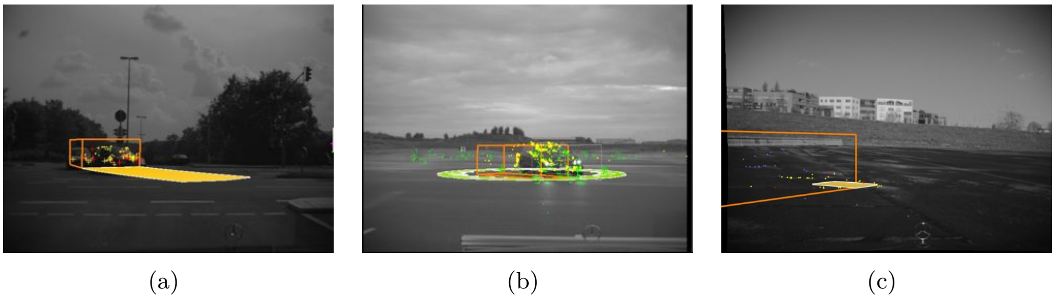


Figure 3.26.: Examples for which the verification failed. (a) The two nearby objects cannot be separated, leading to one huge object that is wider than long. (b) Unrealistic in-place turning due to a much too large yaw rate. (c) Phantom object on empty road, where all points lie on the ground surface.

3.10.2. Merging

Two objects are merged, if there is a spatial overlap of their bounding boxes and if the moving direction and velocity is compatible. A three step algorithm is used to test whether neighboring objects should be merged. This algorithm is sketched as follows:

Spherical Distance First, a fast spherical overlap test is performed. For each object $\boldsymbol{O}_i$, a sphere is placed at the cuboid center with the radius equal to the cuboid length. Let ${}^e\boldsymbol{P_{c,i}}$, ${}^e\boldsymbol{P_{c,j}}$ be the center position of object i and j, then object i and j do very likely not overlap, if

$$[{}^e\boldsymbol{P_{c,i}} - {}^e\boldsymbol{P_{c,j}}]^\mathsf{T} \; [{}^e\boldsymbol{P_{c,i}} - {}^e\boldsymbol{P_{c,j}}] > (l_i + l_j)^2 \tag{3.96}$$

where l_i and l_j indicate the length of object i and j respectively. Only really close objects pass this test and have to be further processed. This test is reduced to the X-Z-plane in practice, ignoring the height distance.

Cuboid Overlap Testing the overlap of two cuboids is a common task in the computer graphics domain. It is typically referred to as collision detection of oriented bounding boxes in the literature. A fast test based on the separating axis theorem for convex shapes has been proposed in [Eberly 2001]. In the 2D case (cuboid projected onto ground plane), four necessary conditions that preclude an overlap are checked successively. If all tests are passed, the bounding boxes overlap. Only objects whose bounding boxes overlap to some amount are considered in the final motion compatibility test.

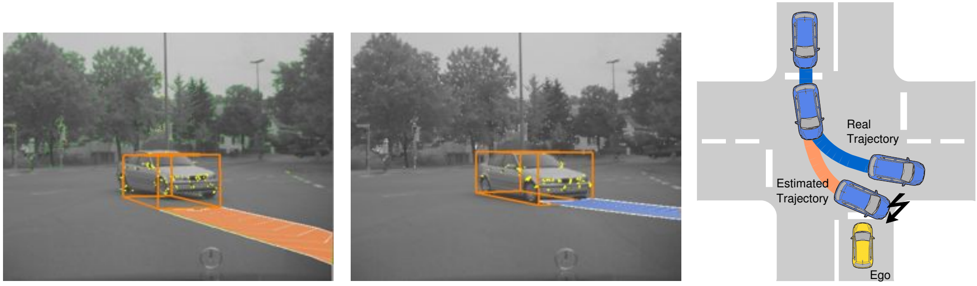
Motion Compatibility Objects with overlapping bounding boxes are finally merged, if they are also compatible in motion. Here, the object motion is approximated by the linear velocity vector that points in the moving direction and is scaled by the current velocity. The Mahalanobis distance of the velocity vectors of two objects is used as indicator of motion compatibility.

Let $\boldsymbol{V}_i = [v_x, v_z]^\mathsf{T} = [\sin(\psi_i + \beta_i)v_i, \cos(\psi_i + \beta_i)v_i]^\mathsf{T}$ denote the 2D velocity vector, that is derived from the current a posteriori state estimate of a given object $\boldsymbol{O}_i$. The corresponding variances and covariances are extracted from the state covariance matrix and summarized in the matrix

$$\mathsf{C}^{(i)} = \begin{bmatrix} \sigma^2_{\psi_i} & \sigma_{\psi_i\beta_i} & \sigma_{\psi_i v_i} \\ \sigma_{\psi_i\beta_i} & \sigma^2_{\beta_i} & \sigma_{\beta_i v_i} \\ \sigma_{\psi_i v_i} & \sigma_{\beta_i v_i} & \sigma^2_{v_i} \end{bmatrix}. \tag{3.97}$$

The covariance matrix of $\boldsymbol{V}_i$ follows from error propagation, i.e., $\mathsf{C}^{(i)}_{VV} = \mathsf{F}^{(i)}\mathsf{C}^{(i)}\mathsf{F}^{(i)\mathsf{T}}$, where $\mathsf{F}^{(i)}$ in this context is the Jacobian of the nonlinear transformation from the state entries to the velocity vector with respect to the parameters ψ, β, and v. It is defined as

$$\mathsf{F}^{(i)} = \begin{bmatrix} \cos(\psi_i + \beta_i)v_i & \cos(\psi_i + \beta_i)v_i & \sin(\psi_i + \beta_i) \\ -\sin(\psi_i + \beta_i)v_i & -\sin(\psi_i + \beta_i)v_i & \cos(\psi_i + \beta_i) \end{bmatrix}. \tag{3.98}$$



(a) (b) (c)

Figure 3.27.: (a) The filter cannot follow a turning vehicle if parametrized for mainly longitudinal movements. (b) The same filter parametrized for turn maneuvers allows for accurate tracking. (c) Bird's eye view on the problem at hand.

The motion compatibility measure is now defined as

$$\delta_{ij} = [\boldsymbol{V}_i - \boldsymbol{V}_j]^\mathsf{T} \left(C_{VV}^{(i)} + C_{VV}^{(j)} \right)^{-1} [\boldsymbol{V}_i - \boldsymbol{V}_j] \,. \tag{3.99}$$

If $\delta_{ij} < T_{\max}$, for a given threshold $T_{\max}$ derived from the χ^2-distribution with 2 degrees of freedom, the object are assumed to be compatible in motion. In this case, the two objects $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$ are merged.

The proposed three step method is a very basic merging strategy, that yields promising results in practice for many situations. More sophisticated methods exist in the field of object fusion, for example, by matching not only the current pose and motion parameters, but also trajectories as has been proposed, for example, in [Hermes et al. 2009a].

3.11. Highly Dynamic Turn Maneuvers

In the approach proposed above, higher order derivatives, such as yaw acceleration, are modeled as zero-mean white Gaussian noise. At highly dynamic turn maneuvers, however, the yaw acceleration becomes a significant issue. Vehicles quickly develop a yaw rate if turning left or right at an intersection. A single (Kalman) filter, parametrized to yield smooth tracking results for mainly longitudinal movements, is often too slow to follow in such situations (see Fig. 3.27). On the other hand, making the filter more reactive in general increases the sensitivity to noise and outliers in the measurements.

In Sec. 2.6.6, different strategies to track maneuvering targets have been presented. Based on this ideas, three solutions to automatically adapt the dynamics of the filter to the dynamics of the object are proposed in the following sections. These solutions include an extension of the motion model considering higher order terms, an IMM-based multi-filter setup, as well as the adaptive parametrization of the filter variances using an independent maximum likelihood estimator. A comparison of the performance of the approaches will be given in the experimental results (see Sec. 4.2.7).

3.11.1. Higher Order Term Motion Model

Mehrotra and Mahapatra [Mehrotra and Mahapatra 1997] have emphasized the importance of higher order terms at tracking highly maneuvering targets. They have proposed a jerk model that describes a maneuvering target up to third order terms.

In terms of the vehicle tracking approach proposed in the previous chapter, the vehicle dynamics are modeled up to second order terms for the position and first order terms in orientation, i.e., longitudinal acceleration and yaw rate, respectively. At turn maneuvers, however, the unmodeled yaw acceleration, i.e., the second derivative of the orientation, becomes crucial.

Thus, an alternative higher order extended Kalman filter approach with constant yaw acceleration motion model is defined. It is denoted as *EKF-YA* in the following. The state vector $\boldsymbol{x}$, as proposed in (3.8), is augmented by an additional motion parameter for the yaw acceleration, $\ddot{\psi}$, yielding

$$\boldsymbol{x}_{\mathrm{ya}} = \left[\underbrace{{}^{e}X_{o},\, {}^{e}Z_{o},\, \psi}_{pose},\, \underbrace{v,\, \dot{v},\, \dot{\psi},\, \ddot{\psi},\, {}^{o}X_{v},\, {}^{o}Z_{v},\, \beta}_{motion} \right]^{\mathsf{T}} \tag{3.100}$$

In case of the full model, the object points $\boldsymbol{\Theta}_{points}$ are appended to this state vector. In this model, the differential equations of the motion parameters, as introduced in (3.14) - (3.17), change to

$$\dot{\psi} = \underline{\dot{\psi}} \tag{3.101}$$

$$\dot{v} = \underline{\dot{v}} \tag{3.102}$$

$$\ddot{\psi} = \underline{\ddot{\psi}} \tag{3.103}$$

$$\ddot{v} = 0 \tag{3.104}$$

$$\dddot{\psi} = 0 \tag{3.105}$$

where underlined variables again correspond to the state vector entries and can be seen as constants in this context. Accordingly, the linearized system matrix A and system noise matrix C_{ww} have to be adapted and extended, respectively, while the measurement model remains unchanged.

3.11.2. Interacting Multiple Models

The IMM-framework enables tracking of targets with time-varying dynamics, by running multiple Kalman filters in parallel. Each filter is designed and parametrized for a different dynamic *mode*. The decision which filter matches the current dynamics best is made automatically at runtime, and the resulting state estimates are combined in a probabilistic sense. For details on the IMM-framework see Sec. 2.6.6.

Two variants of IMM filter configurations are presented in the following. Each consists of two modes: A stationary (or non-maneuvering) mode and a dynamic (maneuvering) mode. The first variant has been extensively tested and evaluated both on simulated and real-world data. The second variant has resulted from the experiences with the first model as well as the EKF-YA approach proposed above. It yields improved results at tracking turning vehicles.

Variant 1 (*IMM-1*): A stationary filter, designed to yield smooth trajectories at mainly longitudinal movements, is combined with a more reactive filter, that is able to follow even at highly dynamic turn maneuvers. Both filters use the same object model, measurement model, and motion model.

The different behavior of the non-maneuvering filter (mode 1) and maneuvering filter (mode 2) is configured via the system noise matrices only, denoted as $C_{ww}^{(stat)}$ and $C_{ww}^{(mnv)}$, respectively. It is possible to parametrize the system matrices in a way that the non-maneuvering filter corresponds to a (constantly) accelerated velocity / constant yaw rate motion model, while the maneuvering filter allows for larger changes of the yaw rate and acceleration. A concrete filter parametrization is given in Sec. 4.2.7.

Variant 2 (*IMM-2*): In this approach, two filters with different motion models are combined. The object and measurement model remains identically. The first filter utilizes a (constantly) accelerated velocity / constant yaw rate (AVCY) motion model, the second filter an accelerated velocity / accelerated yaw rate (AVAY) model.

The AVCY model is parametrized in a way that it covers mainly stationary processes, while the AVAY model is designed for more dynamic maneuvers, especially turn maneuvers. The latter incorporates also the yaw acceleration as proposed before for the EKF-YA filter.

To prevent the problem that both filters do not have the same state vector dimension, the filter state of mode 1 is also augmented by the yaw acceleration. However, it is ignored by the system model and manually set to 0 after the filter update each time step. This has the positive side effect that the yaw acceleration, estimated by the second filter, is always competing with the zero yaw acceleration of the first filter at the IMM mixing step.

In general, combining filters with different motion models is superior to approaches where each filter uses the same motion model and the filter behavior is controlled via the system noise matrices as in variant 1 [Bar-Shalom et al. 2001]. The advantage is that different models are able to generate more distinctive, competing hypotheses at the prediction step.

Adaptation of IMM Framework: The linear mixing of states and covariance matrices, as proposed in (2.48) and (2.49), cannot be applied to our model, since the state vector contains Euler angles. Thus, an alternative nonlinear formulation is used that handles the angles separately. Instead of directly averaging the Euler angles, the corresponding unit vectors, pointing in the direction of the Euler angles, are averaged. The resulting average vector then defines the averaged angle.

In general, the ι th element of $\widehat{\boldsymbol{x}}_j^*$ is computed as

$$\widehat{\boldsymbol{x}}_{j,\iota}^* = \begin{cases} \arctan(\Lambda_1/\Lambda_2) & \iota \in \{\mathrm{idx}(\psi), \mathrm{idx}(\beta)\} \\ \sum_{i=1}^{I} \mu_{i|j} \widehat{\boldsymbol{x}}_{i,\iota}^+ & \text{otherwise} \end{cases} \tag{3.106}$$

with $\Lambda_1 = \sum_{i=1}^{I} \mu_{i|j} \sin(\widehat{\boldsymbol{x}}_{i,\iota}^+)$, $\Lambda_2 = \sum_{i=1}^{I} \mu_{i|j} \cos(\widehat{\boldsymbol{x}}_{i,\iota}^+)$, and the time indices according to (2.48). The idx$(\cdot)$ function returns the index position at which a given variable is stored in the state vector. The same idea is also used for computation of the element-wise difference between state $\widehat{\boldsymbol{x}}_i^+$ and $\widehat{\boldsymbol{x}}_j^*$ in (2.49) for mixing of the covariance matrices.

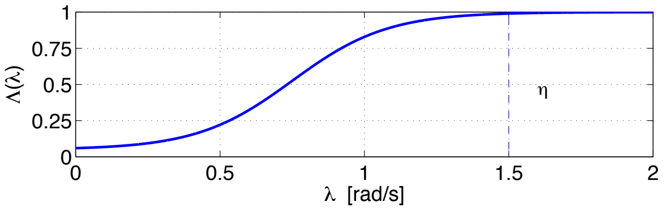


Figure 3.28.: Adaptive noise function as used in the experiments. The function converges to σ_{max} for $\lambda > \eta$ and equals $\sigma_{\text{min}} + \epsilon$ for $\lambda = 0$ (Here: $\sigma_{\text{min}} = 0.05$, $\sigma_{\text{max}} = 1$, $\eta = 1.5$, and $\epsilon = 0.01$).

3.11.3. Adaptive System Noise

The idea of adaptive noise is motivated by manual parameter tuning. In scenes, with highly dynamic turn maneuvers, a human operator parameterizes the filters with larger system variances, e.g., for the yaw rate. This reduces the influence of the system model and enforces the influence of the measurements. On the other hand, at highway scenes with mainly longitudinal movements, the system model is typically much more strengthened by lowering the system variances.

The following approach automates this procedure by steering the reactiveness of a single filter at runtime, depending on the expected dynamics of the tracked object.

Given a set of K measurement vectors $\tilde{\boldsymbol{z}} = \{\boldsymbol{z}(k), \boldsymbol{z}(k-1), \ldots, \boldsymbol{z}(k-(K-1))\}$, where k denotes the current discrete time step, we can estimate

$$\boldsymbol{y} = [\Theta, \Omega(k), \Omega(k-1), ..., \Omega(k-(K-1)]^{\mathsf{T}}, \tag{3.107}$$

i.e., the shape parameters $\boldsymbol{\Theta}$ and a set of object poses, via a maximum likelihood estimation

$$\widehat{\boldsymbol{y}} = \arg\max_{y} \left(p\left(\tilde{\boldsymbol{z}} \middle| \boldsymbol{y}\right)\right), \tag{3.108}$$

with $\widehat{\boldsymbol{y}}$ denoting the estimated parameters. The idea of this approach is, using the known camera geometry, we directly obtain ${}^{e}P_m(k), \ldots, {}^{e}P_m(k-(K-1)$, i.e., the coordinates of the observed object points in the ego-coordinate system at the given time steps. From these point coordinates and the corresponding coordinates in the object system, one can derive the six parameter similarity transformation (no scale) between the ego and object system, which gives the object's pose (see [Horn et al. 1988]). Assuming the object point cloud to be rigid over time, the object point cloud (shape) and the object poses are simultaneously estimated in a least squares sense. Since Gaussian noise is assumed, this is equivalent to a maximum likelihood estimation.

In addition, the motion parameters can be easily derived from the estimated poses by numerical differentiation. The estimation process outside the Kalman filter will be referred to as *oracle* in the following. The oracle is purely data driven and not constrained by a motion model. Thus, the oracle approach is able to follow all movements describable in terms of a 3D rotation and translation.

The yaw rate estimate of the oracle $\widehat{\dot{\psi}}_{\text{OR}}$ gives a strong evidence for maneuvers. At non-maneuvering periods the yaw rate is assumed to be low in typical driving scenarios and may be constant for a longer time interval. On the other hand, a large yaw rate is very likely to change within a short time interval, since vehicles usually do not drive in a circle for very long. This leads to the idea of steering the system noise level based on the magnitude of the yaw rate estimate of the oracle. Using the oracle's yaw acceleration estimate instead of the yaw rate turned out to be too noisy as reliable maneuver detector.

The yaw rate entry in the system noise matrix at a given time step k is set to $\Lambda\left(\widehat{\dot{\psi}}_{\mathrm{OR}}(k)\right)$, where Λ is a sigmoidal function that smoothly interpolates between a minimum and maximum noise level $\sigma_{\min}$ and $\sigma_{\max}$, respectively, as can be seen in Fig. 3.28. It is defined as

$$\Lambda(\lambda) = \frac{\sigma_{\max} - \sigma_{\min}}{1 + \exp\left(-2\ln\left(\frac{\sigma_{\max}-\sigma_{\min}}{\epsilon} - 1\right)\eta^{-1}\left(|\lambda| - \frac{\eta}{2}\right)\right)} + \sigma_{\min} \tag{3.109}$$

where the parameter η corresponds to the value at which Λ has the function value $\sigma_{\max} - \epsilon$. The values of $\sigma_{\min}$, $\sigma_{\max}$, and η are design parameters and ϵ a small constant. A more detailed description on the original approach, that is referred to as *oracle* in this thesis, is given in [Siegemund 2008].

3.12. Summary of Approach

This chapter comprises a detailed description of a road vehicle detection and tracking approach based on stereo image sequences. The overall approach consists of different components, for example, the object model, measurement model, or object detection strategy, that are embedded in one common framework. At several sections above, alternative solutions or parameterizations have been introduced for a particular component, allowing for different system configurations.

For example, two different representations have been proposed in Sec. 3.3.4, that map the proposed object model into a Kalman filter state. The real-time variant separates the problem of estimating the shape parameters, i.e., the structure of the point cloud, from estimating the pose and motion parameters.

The non-linear system equations allow for predicting the current object state a given short time interval ahead. The basic motion model, proposed in Sec. 3.4, assumes constant acceleration and constant yaw rate. With regard to maneuvering vehicles, variants of this motion model have been introduced in Sec. 3.11 that also incorporate the yaw acceleration; either in a single filter implementation or as multi-filter configuration.

The proposed measurement model relates point tracks in the image plane to a fixed object point coordinate. Combining the information of multiple noisy point measurements, the vehicle pose and motion state can be reconstructed. To overcome ambiguities in reconstructing the rotational center, additional geometrical measurements of the observed vehicle origin have been introduced in Sec. 3.5.4. Two examples on how to generate these measurements have been given. In addition, the measurement model allows for fusion with a radar sensor. It has been exemplarily shown, how the velocity measurement of a radar sensor is integrated in Sec. 3.5.5.

Objects are either detected by clustering groups of tracked 3D points with common motion, which is purely image based, or alternatively by the radar sensor as has been proposed in Sec. 3.8. In both cases, the presented data association mechanism (cf. Sec. 3.9) assigns new points to an existing object at run-time to ensure that there are always enough points to contribute to the state update.

Gross errors in the data are detected based on the normalized residuals between the predicted measurements and the actual measurements. Two different strategies for selecting the outlier threshold have been introduced in Sec. 3.7, including constant confidence interval gating and an adaptive threshold that considering the distribution of the normalized residual of all measurements.

In the following chapter, different system configurations are evaluated both on simulated and real-world data. Due to the number of possible configurations, only a selection of all configurations is considered. The experiments will be designed in a way that only one parameter or method is varied at one time.

4. Experimental Results

The vehicle tracking approach is evaluated in three steps. First, the tracking performance of the proposed approach is investigated on simulated data with respect to different filter variants, robustness, and limits in Sec. 4.2. The simulation environment allows for controlled, ideal conditions that can be systematically disturbed.

In Sec. 4.3, the proposed system is tested under realistic conditions on artificial, i.e., rendered, image sequences, where ground truth information on object states is still available from the rendering process. The focus of the artificial scenes is on vehicle tracking at intersection turn maneuvers.

Finally, real world results are shown and discussed in Sec. 4.4, including experiments with robot cars in a controlled testing environment as well as real traffic scenarios. The evaluation criteria used throughout this chapter are presented in Sec. 4.1.

As mentioned before, not all possible configurations and variants of the proposed vehicle tracking approach can be considered at each experiment. Thus, for each particular experiment, a well-defined configuration is used as baseline and only one parameter is varied. The core algorithm is successively extended based on the results of previous results throughout this chapter.

4.1. Evaluation Criteria

Three main aspects are considered for evaluating the quality of a given estimator:

Accuracy: How close are the estimated parameters to the true values on average?

Precision: How reproducible are the results at multiple stochastic runs?

Consistency: How well do the estimated covariance matrices represent the actual error distribution, i.e., does the estimated uncertainty be realistic?

Two accuracy measures are defined. The first one addresses the accuracy of the estimated pose in terms of four corners, which are derived by projecting the object's cuboid model onto the ground plane. The second is used to directly compare the estimated motion parameters (velocity and yaw rate) to the ground truth.

Pose Accuracy and Precision: Considering the cuboid corners overcomes the problem that different configurations of the object and vehicle system define the same pose, which prevents a direct comparison of the pose parameters. Furthermore, the four corners also contain information on the object orientation and the dimension. This leads to the *mean squared corner error* (MSCE), defined for a sequence of K state estimates $\Gamma = \{\widehat{\boldsymbol{x}}(1), \dots, \widehat{\boldsymbol{x}}(K)\}$ as:

$$\mathrm{MSCE}(\Gamma) = \frac{1}{8K} \sum_{k=1}^{K} \boldsymbol{\epsilon}_{\mathrm{corner}}(k)^{\mathsf{T}} \boldsymbol{\epsilon}_{\mathrm{corner}}(k) \tag{4.1}$$

with

$$\boldsymbol{\epsilon}_{\mathrm{corner}}(k) = \begin{bmatrix} {}^{e}\widehat{X}_1(k) & - & {}^{e}X_1(k) \\ {}^{e}\widehat{Z}_1(k) & - & {}^{e}Z_1(k) \\ {}^{e}\widehat{X}_2(k) & - & {}^{e}X_2(k) \\ {}^{e}\widehat{Z}_2(k) & - & {}^{e}Z_2(k) \\ {}^{e}\widehat{X}_3(k) & - & {}^{e}X_3(k) \\ {}^{e}\widehat{Z}_3(k) & - & {}^{e}Z_3(k) \\ {}^{e}\widehat{X}_4(k) & - & {}^{e}X_4(k) \\ {}^{e}\widehat{Z}_4(k) & - & {}^{e}Z_4(k) \end{bmatrix} \tag{4.2}$$

where the object corner coordinates $\left({}^{e}\widehat{X}_i(k),\, {}^{e}\widehat{Z}_i(k)\right)$, $i \in \{1 \dots 4\}$, are computed from the current state estimate $\widehat{\boldsymbol{x}}(k)$ and the object dimension $\boldsymbol{\Theta}_{\mathrm{cube}}$, and $({}^{e}X_i(k),\, {}^{e}Z_i(k))$ denote the corresponding reference corner coordinates. If the reference corners correspond to the true corners (ground truth), this measure is referred to as *pose accuracy* or *outer precision* $\mathrm{MSCE}_{\mathrm{outer}}$, including all error sources, e.g., modeling errors, calibration errors, or noise.

If an experiment contains stochastic processes, such as measurement noise, and this experiment is repeated I times (Monte Carlo simulation), the estimation results will differ between each individual run. However, for an unbiased estimator, the expectation value of all repetitions at one time step must equal the true value if I gets large. Using the sample mean over all estimated corners at time step k as reference in the equations above yields the variance or *inner precision* $\mathrm{MSCE}_{\mathrm{inner}}(\Gamma^{(i)})$ of the pose estimates, given the sequence of states $\Gamma^{(i)}$ of the i-th run. The difference between the outer precision and the inner precision of one run indicates the squared bias of the estimate, i.e.,

$$\mathrm{MSCE}_{\mathrm{outer}}(\Gamma^{(i)}) = \mathrm{MSCE}_{\mathrm{inner}}(\Gamma^{(i)}) + \mathrm{BIAS}^2(\Gamma^{(i)}) \tag{4.3}$$

and thus $\mathrm{BIAS}(\Gamma^{(i)}) = \sqrt{\mathrm{MSCE}_{\mathrm{outer}}(\Gamma^{(i)}) - \mathrm{MSCE}_{\mathrm{inner}}(\Gamma^{(i)})}$. To yield a single score for one experiment, the $\mathrm{MSCE}_{\mathrm{outer}}{}^{(i)}$, $\mathrm{MSCE}_{\mathrm{inner}}{}^{(i)}$, and $\mathrm{BIAS}^{(i)}$ are averaged over all runs and time steps:

$$\overline{\mathrm{MSCE}}_{\mathrm{outer}}(\Gamma^{*}) = \frac{1}{I} \sum_{i=1}^{I} \mathrm{MSCE}_{\mathrm{outer}}(\Gamma^{(i)}) \tag{4.4}$$

$$\overline{\mathrm{MSCE}}_{\mathrm{inner}}(\Gamma^{*}) = \frac{1}{I} \sum_{i=1}^{I} \mathrm{MSCE}_{\mathrm{inner}}(\Gamma^{(i)}) \tag{4.5}$$

$$\overline{\mathrm{BIAS}}(\Gamma^{*}) = \frac{1}{I} \sum_{i=1}^{I} \mathrm{BIAS}(\Gamma^{(i)}) \tag{4.6}$$

with $\Gamma^{*} = \left\{\Gamma^{(1)}, \dots, \Gamma^{(I)}\right\}$ containing all I sequences of state estimates.

Motion Accuracy and Precision: The motion parameters, such as velocity and yaw rate, are evaluated independently based on the mean squared error (MSE) measure, defined as

$$\mathrm{MSE}_x(\Gamma_{\widehat{x}}) = \frac{1}{K}\sum_{k=1}^{K}(\widehat{x}(k) - x_{\mathrm{ref}}(k))^2 \tag{4.7}$$

for a sequence of K estimates $\Gamma_{\widehat{x}} = \{\widehat{x}(1), \ldots, \widehat{x}(K)\}$. In this context, the scalar x is used as template for the velocity v and the yaw rate $\dot{\psi}$. Analog to the pose accuracy and precision, the reference value x_{ref} can be either the true value $\tilde{x}$ or the sample mean $\bar{x}$ over a number of Monte Carlo runs. Accordingly, the $\overline{\mathrm{MSE}}_{\mathrm{outer}}$ and $\overline{\mathrm{MSE}}_{\mathrm{inner}}$ are defined for the motion parameters. Alternatively, the root mean squared error (RMSE) is used, i.e., the square root of the MSE.

Filter Consistency: An estimator is defined to be *consistent*, if the estimated covariances are a realistic approximation of the real error distribution. The consistency of an estimator is tested based on the normalized mean squared error (NMSE) over I Monte Carlo runs at a given time step k, with

$$\mathrm{NMSE}(\Gamma_{\widehat{x}}, k) \quad = \quad \frac{1}{I}\sum_{i=1}^{I}\left(\widehat{\boldsymbol{x}}^{(i)}(k) - \tilde{\boldsymbol{x}}(k)\right)^{\mathsf{T}} \mathsf{C}_{\widehat{x}\widehat{x}}^{(i)^{-1}} \left(\widehat{\boldsymbol{x}}^{(i)}(k) - \tilde{\boldsymbol{x}}(k)\right), \tag{4.8}$$

i.e., the mean squared Mahalanobis distance to the ground truth. Assuming the estimated parameters $\widehat{\boldsymbol{x}}^{(i)}(k) \in \mathrm{I\!R}^{\mathrm{N}}$ are normal distributed, with covariance matrix $\mathsf{C}_{\widehat{x}\widehat{x}}^{(i)}$, and every of two state estimates are uncorrelated, the NMSE follows a χ^2-distribution with NI degrees of freedom, i.e.,

$$I \cdot \mathrm{NMSE} \sim \chi^2_{NI}. \tag{4.9}$$

The hypothesis that a given estimator is consistent is rejected with significance level α, if the NMSE lies outside the confidence interval:

$$[\mathrm{icdf}_{\chi^2_{NI}}(\alpha/2)/I,\ \mathrm{icdf}_{\chi^2_{NI}}(1-\alpha/2)/I], \tag{4.10}$$

where $\mathrm{icdf}_{\chi^2_{NI}}$ represents the inverse cumulative density function of the χ^2_{NI} distribution. If the NMSE is larger than the upper bound, the filter is too optimistic, i.e., the variances are underestimated. On the other hand, if the NMSE is below the lower bound, the filter is too pessimistic, i.e., the estimated variances are too large.

To overcome the problem of the ambiguous definition of the object coordinate system, a reduced state vector $\boldsymbol{x}^\star \in \mathrm{I\!R}^6$ is considered:

$$\boldsymbol{x}^\star = \left[{}^{e}X_v,\ {}^{e}Z_v, \chi, v, \dot{\psi}, \dot{v}\right]^{\mathsf{T}} \tag{4.11}$$

with $[{}^{e}X_v, 0\,{}^{e}Z_v, 1]^{\mathsf{T}} = {}^{e}\mathsf{W}_o\,{}^{o}\mathsf{W}_v[0,0,0,1]^{\mathsf{T}}$ representing the vehicle origin $\boldsymbol{O}_v$ in ego-coordinates, and $\chi = {}^{e}\chi_v = \psi + \beta$ the rotation between vehicle and ego-system. This means, the vehicle system is defined as object system. Accordingly, the covariance matrix entries for ${}^{e}X_v$, ${}^{e}Z_v$, and χ have to be computed via error propagation. The point positions are excluded from the consistency evaluation.

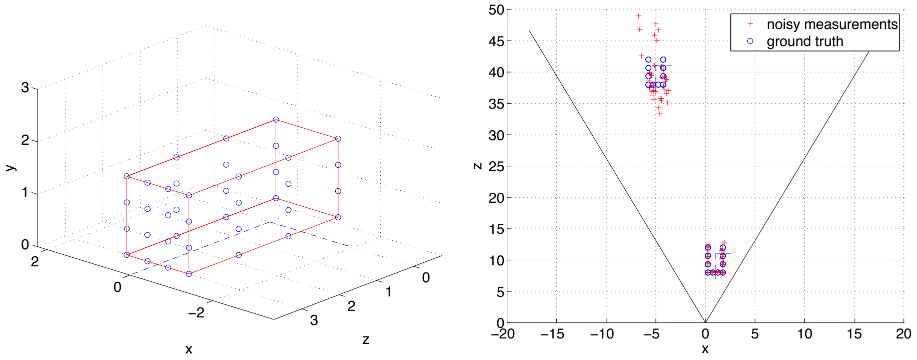
4.2. Simulation Results

The simulation experiments are organized as follows. First, the simulation environment and the considered trajectory test bed is introduced. Then, the results of five different experiments using this simulation environment are presented that should answer the following questions:

1. How accurate are the pose estimates of different filter approaches at increasing random measurement noise?
2. How consistent are these filters?
3. How do the proposed outlier handling strategies perform for different amounts of gross errors in the measurements?
4. What effect have the number of object points and the point update strategy on the estimation accuracy?
5. How accurate are the estimated motion parameters at highly dynamic turn maneuvers?

At the end of this section, a summary of the simulation results is given.

4.2.1. Simulation Setup



(a)

(b)

Figure 4.1.: (a) Simulated object with 40 true point positions in vehicle coordinate system. (b) Bird's eye view on this object in camera coordinates, observed at 10 m and 40 m distance, respectively. Both the ground truth position of the points and the noisy measurements are shown.

In the simulation environment, objects are represented as rigid point clouds that are moved along virtual trajectories. The ground truth point positions in vehicle coordinates are known as well as the pose and motion parameters of the virtual *vehicle* for each time step. The point clouds are observed by a virtual stereo system with known intrinsic and extrinsic camera parameters. The correspondence between a given object point ${}^{o}\boldsymbol{P}_m$ and the corresponding image position (u_m, v_m) in the left and right camera is assumed to be known. The stereo disparity d_m is computed as the difference of the u-coordinate in both images following (2.4).

To model real-world errors in the data, the ground truth image coordinates in horizontal direction are disturbed by zero-mean, white Gaussian noise with standard deviation σ_u in both images. The error of the horizontal image coordinates propagates to the resulting disparity values. This leads to a realistic error distribution of the reconstructed 3D object point clouds, i.e., the error increases quadratically with distance. Beside adding Gaussian noise it is possible to incorporate also gross errors, corresponding to outliers in the data.

Fig. 4.1(a) shows an example of a 2.0 × 4.0 × 1.5 m (W×L×H) object, represented by 40 points in vehicle coordinates. This object has been placed once at 10 m, once at 40 m distance, respectively, in Fig. 4.1(b). The corresponding noisy measurements (crosses) in camera coordinates as well as the ground truth point positions (circles) are visualized. As expected, the object point measurements are spread much more at 40 m compared to points at about 10 m distance. In this example, the standard deviation of the image noise is $\sigma_u = 0.5$ pixels. The parameters of the virtual stereo camera system are summarized in Table 4.1.

Parameter	**Value**
principal distance	840 pix
principal point (u_0, v_0)	$(320, 240)$
camera height	1.26 m
stereo baseline	0.3 m
camera pitch angle	0.0 rad
camera yaw angle	0.0 rad
camera roll angle	0.0 rad

Table 4.1.: Intrinsic and extrinsic parameters of virtual stereo camera system.

Object trajectories are generated based on the dynamic model as proposed in Section 3.4, or extracted from real-world recordings of trajectories based on inertial sensors. Thus, it is possible to move a virtual object along the trajectory of a real vehicle to yield maximum realistic maneuvers. In both cases, ground truth information on the motion parameters to be estimated is available at any time.

A common test bed is defined, covering seven typical scenarios, including straight motion and turn maneuvers at different distances and directions (oncoming, intersecting, leading) as visualized in Fig. 4.2. Each trajectory consists of 100 discrete time steps.

In the following experiments with simulated data, only the raw point measurement model is investigated, i.e., there are no rotation point or radar velocity measurements as proposed in Sec. 3.5.4 and 3.5.5 respectively.

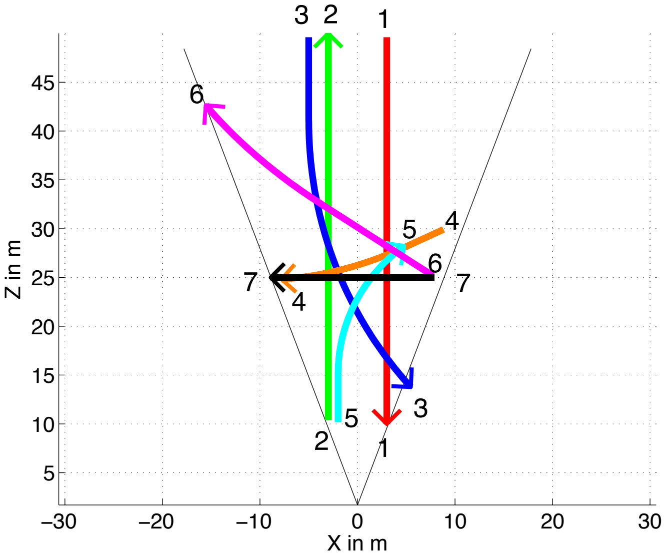


Figure 4.2.: Trajectory test set covering different scenarios such as oncoming, turning, crossing, or leaving vehicles, in the field of view of the camera.

4.2.2. Filter Configuration

In the simulation experiments, the following filter approaches are considered:

- Standard extended Kalman filter (EKF)
- Iterated extended Kalman filter with 3 iterations (IEKF3)
- Unscented Kalman filter (UKF)
- EKF with yaw acceleration (EKF+YA)
- EKF with oracle-based adaptive system noise (EKF+OR)
- Interacting Multiple Models filter (IMM)

The values of the system noise matrices are summarized in Table 4.2. If not stated otherwise below, the object points are part of the state vector and no outlier test is performed.

Throughout the simulation experiments only the IMM variant, introduced as *IMM-1* in Sec. 3.11.2, is considered, which models the different filter behavior by different parametrization of the system noise matrices $C_{ww}^{(stat)}$ and $C_{ww}^{(mnv)}$, respectively. The state vector entries as well as the underlying motion model are equal to the remaining approaches considered in this experiment for fair comparison, i.e., the yaw acceleration is not incorporated here. The mode transition probability matrix $P^{(IMM)}$ for the IMM filter has been chosen as

$$P^{(IMM)} = \begin{bmatrix} 0.98 & 0.02 \\ 0.02 & 0.98 \end{bmatrix} \tag{4.12}$$

	σ_X	σ_Z	$\sigma_\psi, \sigma_\beta$	σ_v	$\sigma_{\dot{\psi}}$	$\sigma_{\dot{v}}$	$\sigma_{\ddot{\psi}}$
	[m]	[m]	[rad]	[m/s]	[rad/s]	[m/s^2]	[rad/s^2]
EKF/IEKF3	0.01	0.01	0.001	0.001	0.05	1.0	-
UKF	0.01	0.01	0.001	0.001	0.05	1.0	-
EKF+YA	0.01	0.01	0.001	0.001	0.0001	1.0	1.0
EKF+OR	0.01	0.01	0.001	0.001	$\Lambda(\lvert\widehat{\dot{\psi}}_{OR}\rvert)$	1.0	-
IMM (stat)	0.01	0.01	0.001	0.001	0.0001	0.0001	-
(mnv)	0.01	0.01	0.001	0.001	1.0	1.0	-

Table 4.2.: System noise matrix parametrization for different filter approaches used throughout the simulation experiments.

where the value at the i-th row and j-th column indicates the *a priori* probability p_{ij} that the filter switches from mode i to mode j, $i, j \in \{1, 2\}$, i.e., it is much more likely that the filter remains in the same mode. Diagonal entries close to 1 prevent frequent mode switching and result in more distinct mode probabilities.

The adaptive noise function $\Lambda(.)$ for the EKF-OR approach has been parametrized with $\sigma_{\text{min}} = 0.05$ rad/s, $\sigma_{\text{max}} = 1.0$ rad/s, and $\eta = 1.5$ (cf. Sec. 3.11.3). This means, for small $\widehat{\dot{\psi}}_{\text{OR}}$ the EKF-OR filter is configured as the standard EKF filter and for $|\widehat{\dot{\psi}}_{\text{OR}}| > 1.5$, the filter corresponds to the maneuvering mode of the IMM filter. A window size of $k = 5$ has been used for the oracle.

A constant measurement noise $C_{vv}^{(p)}$ for each point is assumed throughout all experiments, with $C_{vv}^{(P)} = \text{Diag}(\sigma_u^2, \sigma_v^2, \sigma_d^2) = \text{Diag}(0.5^2, 0.5^2, 0.5^2)$ for all filters (see Sec. 3.6.2).

4.2.3. Filter Precision at Increasing Noise

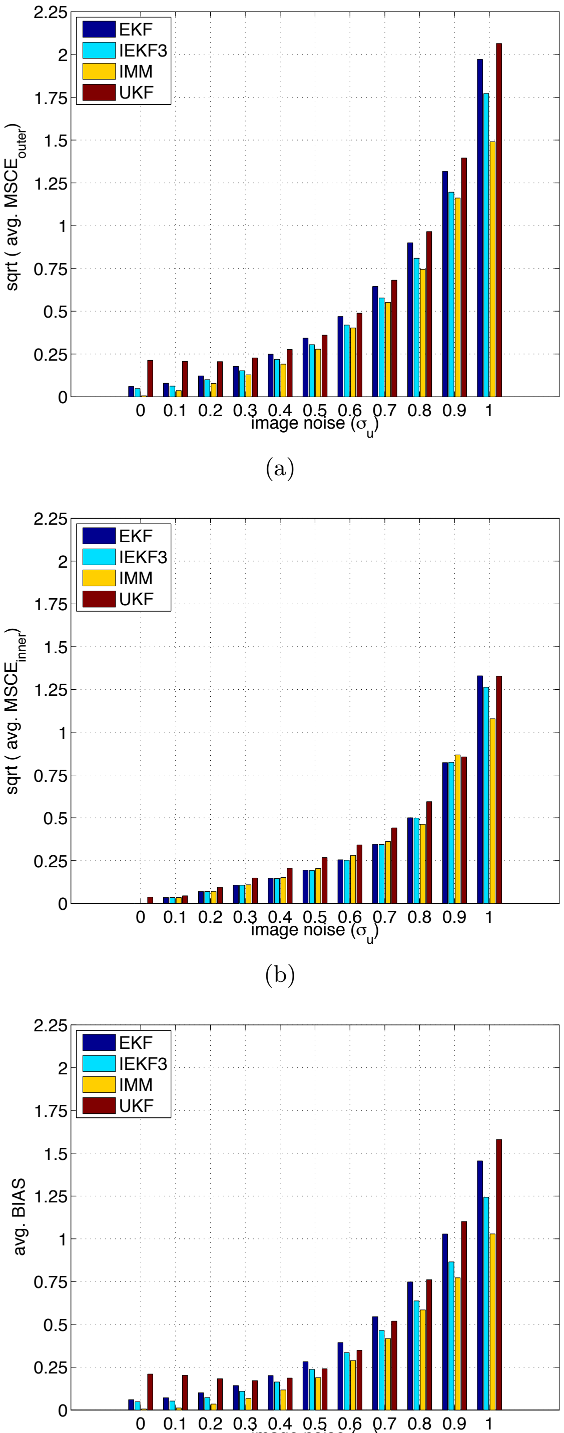
In this experiment, the tracking precision of different Kalman filter approaches on the test bed, with respect to an increasing amount of measurement noise, is compared, including the EKF, IEKF3, UKF, and IMM approach.

Setup: An object, modeled by $M = 40$ points as shown in Fig. 4.1(a), is moved along the test bed trajectories (cf. 4.2.1). The measurements are disturbed by eleven discrete image noise levels in the range of $\sigma_u \in [0, 1]$.

The tracking is initialized with the object coordinate system placed at the centroid of the initial noisy point cloud. All filters are initialized with the same state vector. Orientation and velocity are initialized with the ground truth. The initial shape model is derived from the initial point cloud and has to be refined over time. The remaining state parameters, such as yaw rate or acceleration, are initialized with zero. Each individual experiment has been repeated $I = 20$ times with random measurement noise. The full object state including the object point positions is estimated each run.

Evaluation Criteria: The MSCE, as defined in (4.1), is used as evaluation criteria. Both the average outer and inner precision, $\overline{\text{MSCE}}_{\text{outer}}$ and $\overline{\text{MSCE}}_{\text{inner}}$ respectively, are computed based on this measure, from which the average estimation bias $\overline{\text{BIAS}}$ can be derived. To reduce the influence of initialization errors, for each sequence the first 20 frames out of 100 have been excluded from the evaluation.

Observations: The results are summarized in Fig. 4.3. The root of the mean squared



(a)

(b)

(c)

Figure 4.3.: Filter precision as a function of the measurement noise level. The bars indicate the average object pose error (inner and outer precision) and average estimation bias over all trajectories and iterations with respect to the given image noise level. The IMM filter performs best in this experiment.

corner error increases approximately quadratically with the amount of image noise at all four filter approaches. This indicates there are significant nonlinearities in the system, since for a linear system one would expect a linear increase of the error.

The IMM approach yields the most accurate results in this experiment at all noise levels, while the precision is comparable to the other approaches. The IEKF3 (three iterations) outperforms the standard EKF especially at larger noise levels and is the best single filter approach in this experiment. It turned out that more than three iterations do not further improve the precision. The UKF performs similar to the EKF approach at noise levels close to the expectation ($\sigma_u \approx 0.5$), but results in significantly larger errors at low noise levels.

The $\overline{\text{MSCE}}_{\text{outer}}$ shows that the **accuracy** of all filter approaches is below 0.25 m for noise levels <= 0.4, i.e., the average distance between an estimated object corner and the real position is below 25 cm. Since the test bed includes trajectories at large distance, the error is acceptable small. At $\sigma_u = 0.1$, the accuracy is below 4 cm for the IMM filter and about 6 cm for the IEKF3.

In all experiments there is at least a small **bias** in the estimate, which also increases quadratically with the image noise. There are two main origins for the bias: Restrictions of the system model and systematic errors in the depth estimate.

At maneuvers, e.g., sudden yaw acceleration, the filter can follow the object only with a delay. As the maneuver starts, it takes a few time steps until the filter estimates the real yaw rate accurately. In the meantime, the estimated object pose deviates systematically from the real value.

Furthermore, for noise levels $\sigma_u > 0$, the observed 3D point positions are biased in depth. A normal distribution in 3D space is assumed, although normal distributed image noise spreads out asymmetrically in 3D space around the actual 3D point position due to the nonlinear inverse projection. In general, the estimated point positions are further away then the real value. This effect influences the state estimation at far distances and vanishes in the near range. Since the object position is initialized from the centroid of the initial (noisy) point cloud, the initial object pose is also biased.

In further experiments, not detailed here, it has been verified, that the pose error increases linearly with the system noise for a stationary object at near distance (10 m).

Conclusions: From this experiment it is concluded that the IMM approach is best suited to yield accurate and precise tracking results at a variety of different scenarios. The iterated extended Kalman filter yields the best results among the single filter approaches. Only these two filters will be further investigated in the following.

4.2.4. Filter Consistency

In this experiment, the filter consistency of the IEKF3 and IMM approach is evaluated.

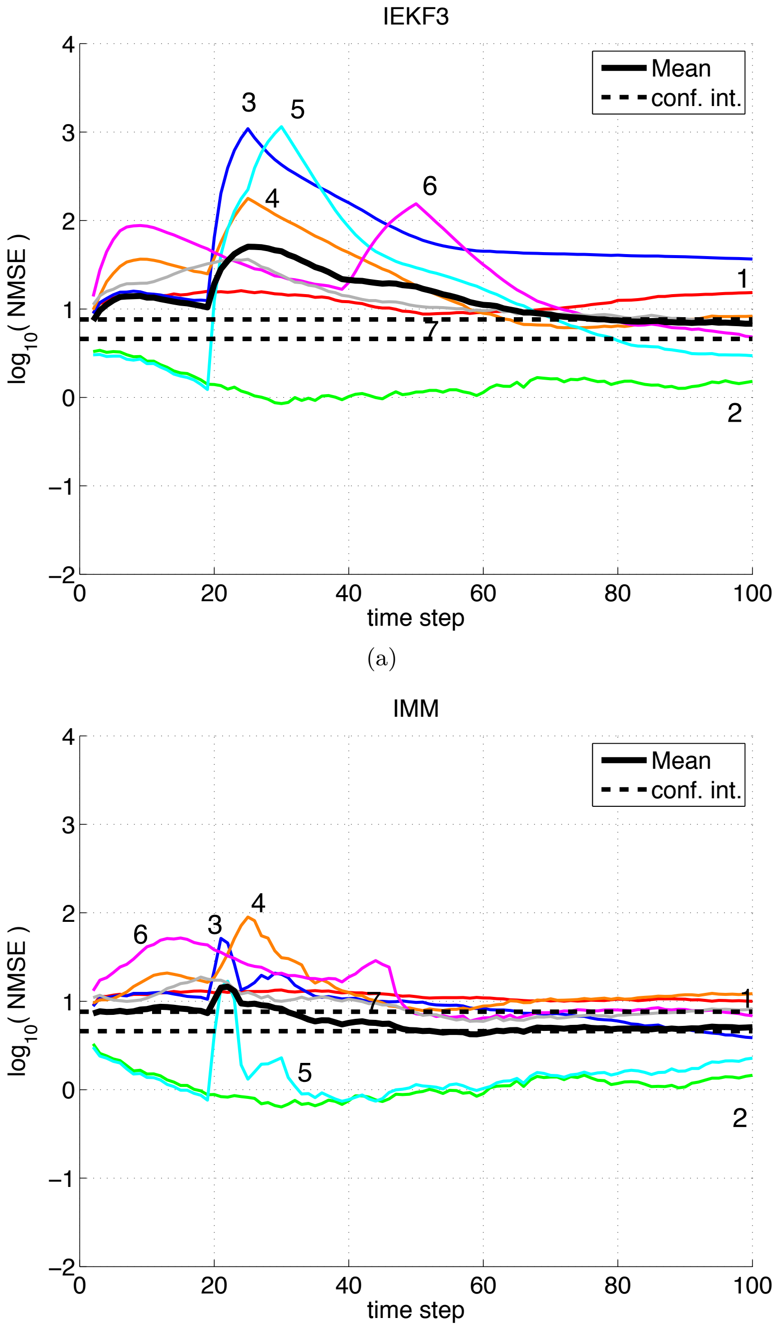
Setup & Evaluation Criteria: The same test bed as in Experiment 1 is considered. For each sequence the NMSE, as defined in (4.8), is computed for each time step k for the transformed state vector $\boldsymbol{x}^*$ (see (4.11)) over $I = 20$ monte carlo runs. This yields a total number of $I = 20$ normalized squared residuals per time step. The sum of these residuals is χ^2_{120} distributed ($120 = 20 \times 6$ degrees of freedom), if the normalized residuals are standard normal distributed. According to (4.10), the hypothesis that the latter is true is rejected with 5% significance level if the actual $NMSE$ is outside the confidence interval $[\text{icdf}_{\chi^2_{120}}(0.025)/I,\ \text{icdf}_{\chi^2_{120}}(0.975)/I] = [4.5786, 7.6106]$. The image noise level has been $\sigma_u = 0.5$ in all runs.

Observations: The development of the NMSE over time for the different trajectories in the data set as well as the average NMSE over all trajectories is shown in Fig. 4.4. For better visualization of the results, the $\log_{10}$ of the NMSE is displayed. The consistency varies between different scenarios and filter approaches.

The IEKF3 filter yields much larger NMSE values compared to the IMM approach, especially at scenarios with turn maneuvers such as trajectory 3, 4, 5, and 6. These scenarios lead to significant inconsistencies during the maneuvering phase. The filters are too optimistic in this case, i.e., the covariances are underestimated, while the actual error is significantly larger. The IMM filter converges much faster to a consistent state than the single filter approach.

In those scenarios in which the object has been initialized at close distance (e.g., scenario 1 or 5), the filters are too pessimistic most of the time, i.e., the actual estimate is more precise than the filter covariance matrices indicate.

Conclusions: At many time steps, the system noise matrix is either too small or too large in these experiments, indicating the compromise that is made at the filter configuration. There is a trade-off between reactiveness of the filter and smoothness, i.e., insensitivity to errors in the measurements by a strong system model. The IMM filter outperforms the IEKF3 approach in consistency in the given experiments, due to its ability to switch between two system models at runtime.



(a)

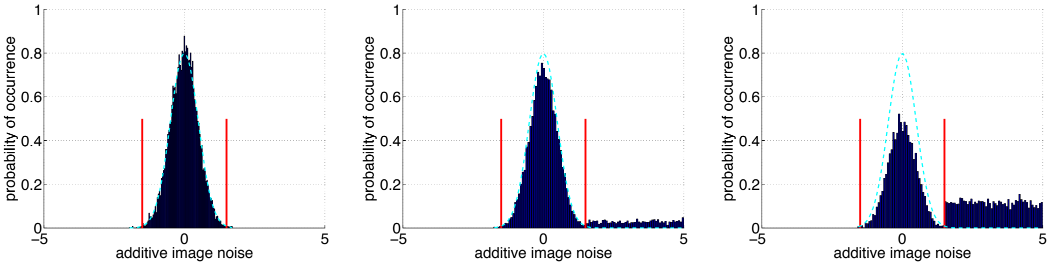
(b)

Figure 4.4.: Consistency over time, computed over 20 monte carlo runs per time step, for (a) IEKF3 and (b) IMM. The filter is *consistent* if the NMSE lies within the confidence interval indicated by the dashed-line (here: logarithmic scale). A NMSE below or above the confidence interval indicates the estimated variances are either too pessimistic or too optimistic, respectively.

4.2.5. Sensitivity to Outliers

The previous two experiments have been based on zero-mean white Gaussian measurement noise. There have not been any gross errors in the data, deviating from the assumed stochastic model. In this experiment it is investigated how the filter precision is influenced by significant outliers in the data. Therefore, the same trajectory test bed is considered, but this time with an increasing percentage of outliers in the data.

Setup & Evaluation Criteria: At each time step, a number of object points is selected randomly based on the given outlier level. Instead of adding a normal distributed noise term to these samples, the noise term is derived from a uniform distribution in the range $[3\sigma_u, 10\sigma_u]$. The resulting noise terms are biased and outside the 3-σ confidence interval of the normal distribution from which the remaining *non-outlier* noise terms are sampled from (see Fig. 4.5).



(a) 0% outliers (b) 10% outliers (c) 40% outliers

Figure 4.5.: Generated noise statistics with different percentages of real outliers. The vertical lines indicate the $3-\sigma$ interval of the expected normal distribution $\mathcal{N}(0, \sigma_u)$, shown as dashed line, with $\sigma_u = 0.5$.

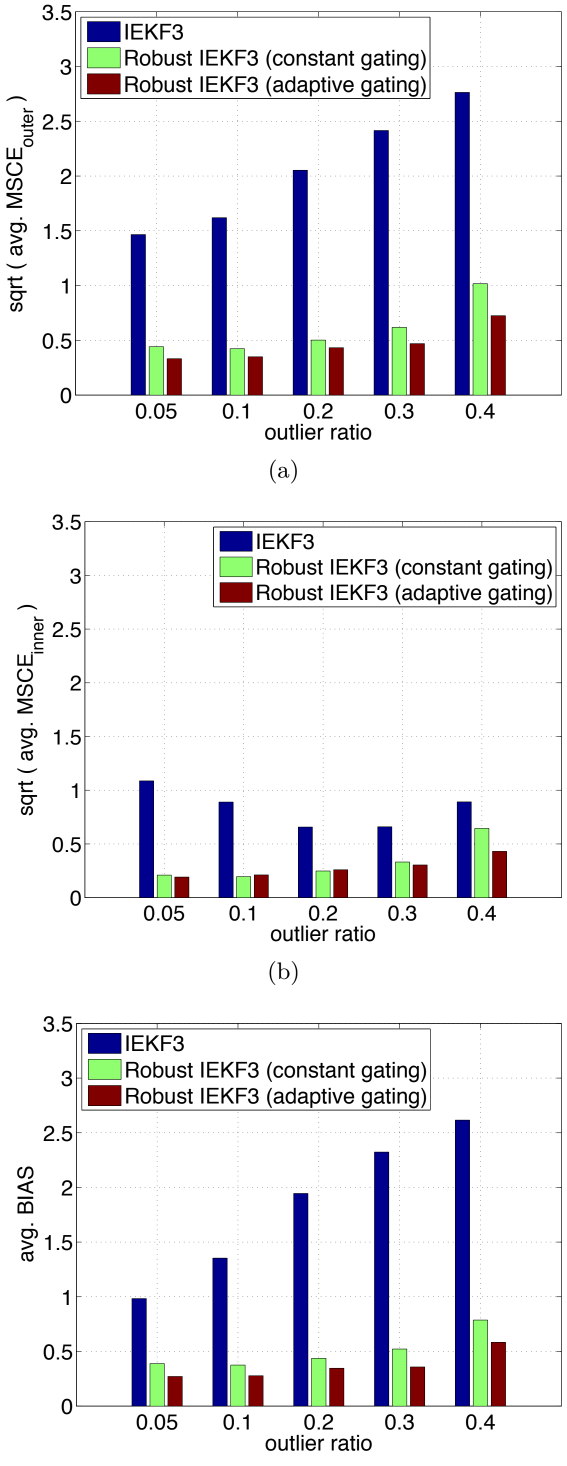
The following percentages of gross errors are investigated: 5%, 10%, 20%, 30%, and 40%. The basic image noise level is $\sigma_u = 0.5$ in this experiment.

Two outlier handling strategies, as proposed in Sec. 3.7, are compared to the standard approach without outlier detection:

- Robust reweighing of residuals with constant gating (3-σ confidence interval)
- Robust reweighing of residuals with adaptive gating

The results are compared based on the average corner error as in previous experiments.

Observations: Fig. 4.6 shows the resulting outer and inner accuracy as well as the bias of the four corner error measure. As can be seen, the outer error and bias increases approximately linearly with the percentage of outliers for the approach without outlier handling (IEKF3). Both robust filter variants yield significantly lower errors. The constant gating approach reduces the outer precision by about 1/4, by excluding all measurements that differ too much from the expectation. The adaptive gating approach guarantees that at least 50% of all measurements contribute to the filter update. Since there are no more than 40% of outliers in the data, the adaptive gating approach reduces the error even up to about 1/6 compared to the standard IEKF3 approach.



(a)

(b)

(c)

Figure 4.6.: Simulation results for an increasing percentage of gross errors in the data for the iterated Kalman filter without outlier test and two robust versions.

Conclusions: Both robust filter variants outperform the original approach without outlier removal. The adaptive gating approach yields the best results in this experiment and, thus, qualifies as default outlier handling strategy for following experiments with artificial and real-world scenes.

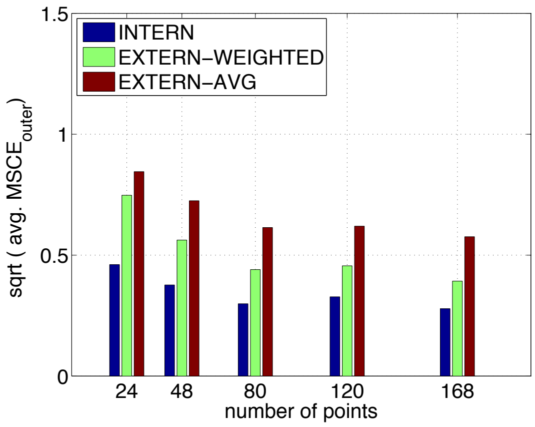
4.2.6. Point Update Strategy

Two different filter representations have been introduced in Sec. 3.3.4: A full model that includes the point positions in the filter state and a reduced model, which updates the point positions outside the filter. In this experiment it is investigated how the different models perform on the simulation test bed for a varying number of object points.
Setup & Evaluation Criteria: The following approaches are compared:

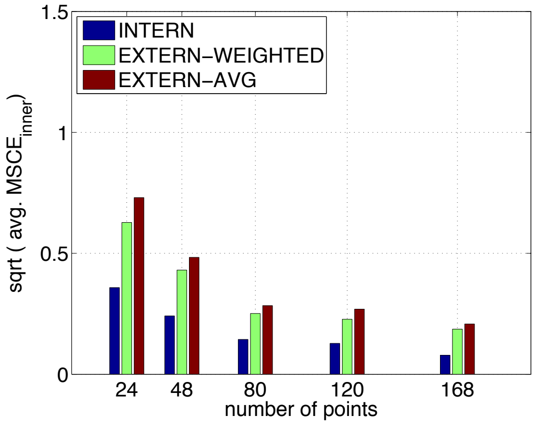
- IEKF3 with full model (INTERN)
- IEKF3 with reduced model + external point update by ML-estimation (EXTERN-WEIGHTED)
- IEKF3 with reduced model + external point update by averaging (EXTERN-AVG)

The first variant of the reduced model follows the approach as proposed in Sec. 3.5.3. Here, the point positions are updated under consideration of the point covariance matrices in a maximum likelihood manner, assuming uncorrelated measurement groups. The second variant ignores the uncertainties of a point, i.e., the point update corresponds to averaging over all previous positions. This is equivalent to setting $\mathsf{C}_m(k) = \mathsf{I}_3$ in (3.46) for each point ${}^{o}\boldsymbol{P}_m$ and time step.

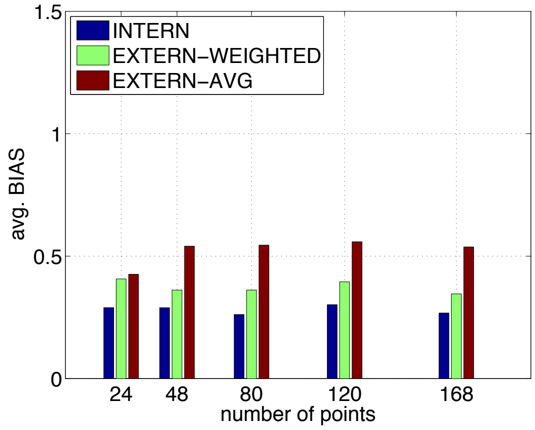
Each scenario in the test bed is repeated 20 times for 24, 48, 80, 120, and 168 object points, respectively. Random zero-mean white Gaussian image noise with $\sigma_u = 0.5$ has been added to the measurements. For evaluation, the measures $\overline{\text{MSCE}}_{\text{outer}}$, $\overline{\text{MSCE}}_{\text{inner}}$, and $\overline{\text{BIAS}}$ are considered.
Observations: The results are summarized in Fig. 4.7. In all approaches, the outer and inner precision decreases with an increasing number of points, while the bias remains approximately constant. As can be seen, the full model outperforms the reduced model approaches significantly in accuracy and precision. Compared to the full model, the root of the $\overline{\text{MSCE}}_{\text{outer}}$ is about 1.6 times larger for the external point update with covariance weighting and about 1.8 time larger for the fast alternative of point averaging at 24 points. This ratio slowly decreases with the number of points to about 1.4 for EXTERN-WEIGHTED and remains approximately constant for the EXTERN-AVG approach.
Conclusions: The more points are contributing, the more accurate are the results. The full model outperforms both approaches that estimate the point positions separately. One significant drawback of the full model is the computation time. Since the filter state increases by three elements for each point in the model, one can easily end up with filter states of more than $300 - 900$ entries in practice. Thus, the reduced model yields a good compromise between estimation accuracy and computation time.



(a)



(b)



(c)

Figure 4.7.: The number of points as well as the update strategy influence the estimation accuracy and precision. Updating the point positions simultaneously with the pose and motion parameters within one large filter state outperforms the much faster approaches that refine the point position independently outside the filter.

4.2.7. Highly Dynamic Turn Maneuvers

In this experiment, the different approaches for tracking highly maneuvering vehicles, as proposed in Sec. 3.11, are compared. While previous experiments have mainly considered the pose estimation error, in the following the focus is on the motion parameters. The evaluation is divided into two parts. First, the different filter behavior to a sudden synthetic yaw acceleration is exemplarily demonstrated. Then, the filters are evaluated based on 57 real vehicle trajectories recorded from in-car sensors at different urban intersections.

Sudden Yaw Acceleration

The first scenario contains a turn maneuver that cannot be followed well by the single EKF approach if parametrized for typical longitudinal or slow turn movements.

Setup & Evaluation Criteria: The simulated vehicle, represented by 60 object points, starts approaching at 50 m distance with constant velocity of 10 m/s and a small yaw acceleration of 0.05 rad/s^2. After 50 time steps, a sudden constant yaw acceleration of $\ddot{\psi} = 2$ rad/s^2 is generated for 10 time steps (0.4 s). The simulation has been repeated 40 times. The measurements are disturbed by adding Gaussian noise with standard deviation $\sigma_u = 0.1$ pixel to the horizontal image coordinate in both images. This lower noise level is chosen to emphasize error sources induced by the motion model.

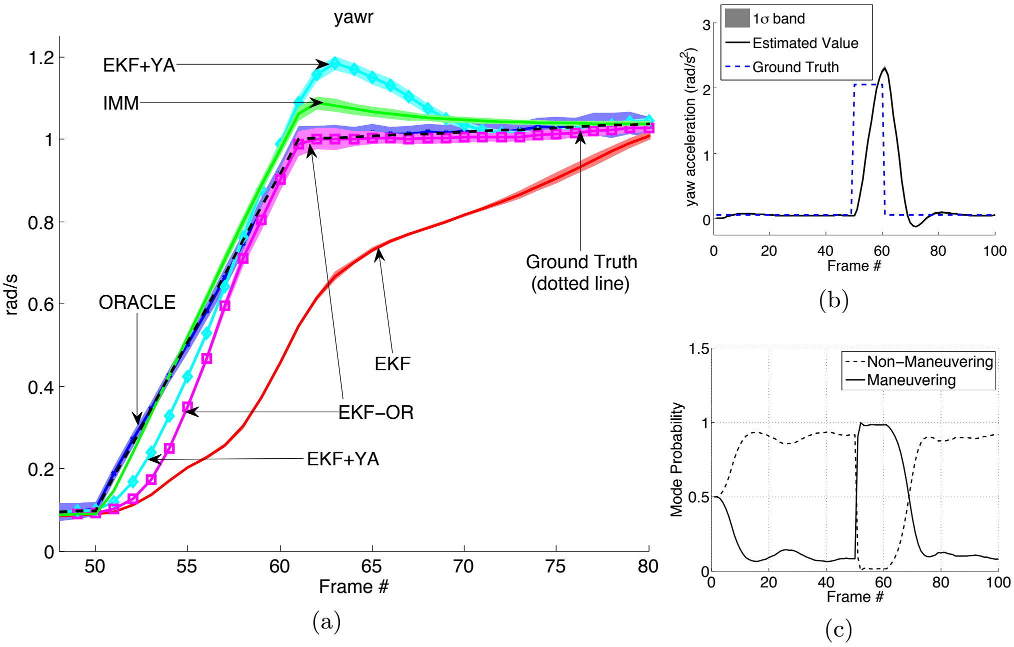
The following filter approaches, configured as proposed in Sec. 4.2.2, are evaluated: EKF, EKF+YA, EKF+OR, and IMM. The considered evaluation criteria is the root mean squared yaw rate error (RMSE$_{\dot{\psi}}$).

Obervations: The mean estimated yaw rate for each filter is shown in Fig. 4.8(a), together with the 1-σ error band. As can be seen, the single EKF approach with constant yaw rate assumption cannot follow the fast yaw rate increase (RMSE$_{\dot{\psi}}$ = 0.1492), while the proposed extended versions approximate the ground truth much better. Differences are in the delay of the yaw rate increase, i.e., how fast does a given filter react to the yaw acceleration, and in *overshooting*.

The EKF-YA filter quickly follows the yaw rate increase by estimating the yaw acceleration (RMSE$_{\dot{\psi}}$ = 0.0443). Fig. 4.8(b) shows the response to the rectangular yaw acceleration input. The resulting overshooting of the estimated yaw rate is about twice as large compared to the IMM approach, which is able to follow the yaw rate increase even faster by switching from non-maneuvering mode to maneuvering mode at frame 51 (see Fig. 4.8(c)). As the yaw acceleration maneuver ends at frame 60, the probability of the maneuvering mode starts to decrease. The root mean squared error in the yaw rate estimate is RMSE$_{\dot{\psi}}$ = 0.0230 for the IMM filter.

The oracle detects the yaw rate increase without delay and shows no overshooting at all. Since the oracle is model-free and unconstrained, the results yield a larger standard deviation compared to the filtered estimates. The resulting trajectories are quite unsteady. However, combining the oracle with a Kalman filter in the EKF-OR approach, yields both a very accurate yaw rate estimate and smooth trajectories with almost no overshooting (RMSE$_{\dot{\psi}}$ = 0.0392). The delay until the yaw rate starts to increase depends on the design of the adaptive noise control function (see Fig. 3.28).

Conclusions: The adaptive system noise approach EKF+OR yields the best results in

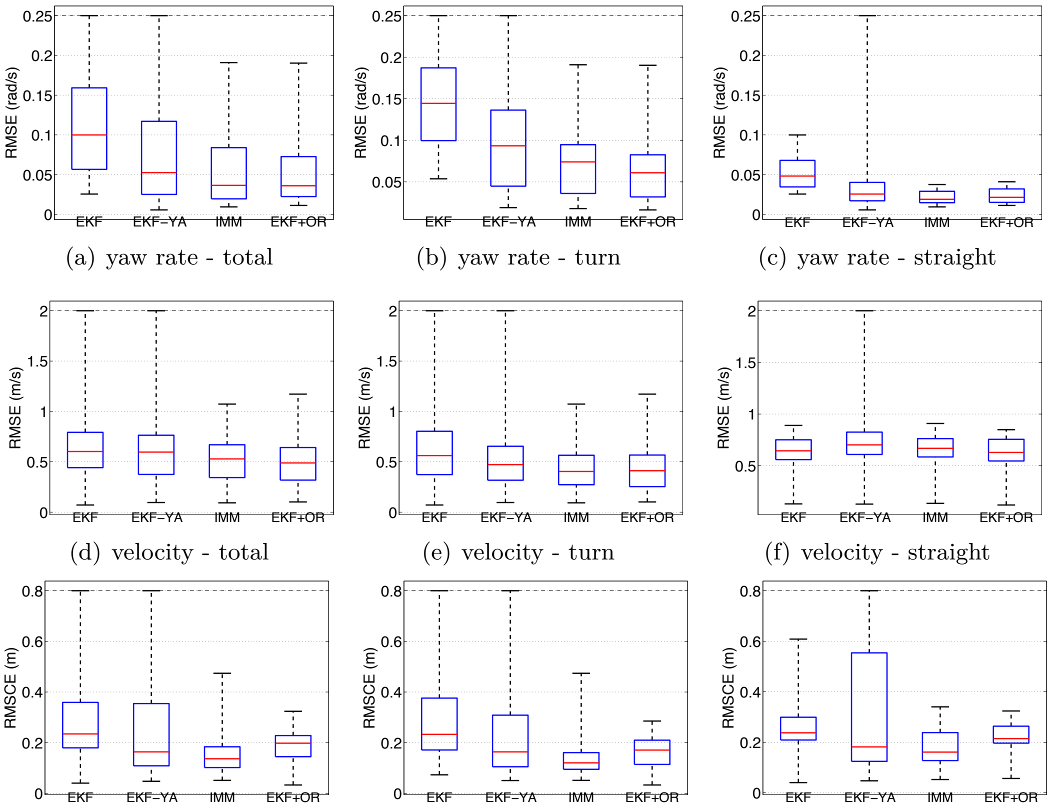


(a)

(b)

(c)

Figure 4.8.: (a) Error bands of yaw rate estimates based on mean and standard deviation over 40 monte carlo runs. All filter extensions approximate the ground truth much better than the original single EKF approach. (b) Estimated yaw acceleration of EKF-YA filter compared to ground truth. (c) IMM mode probabilities.



(a) yaw rate - total

(b) yaw rate - turn

(c) yaw rate - straight

(d) velocity - total

(e) velocity - turn

(f) velocity - straight

(g) pose - total

(h) pose - turn

(i) pose - straight

Figure 4.9.: Distribution of RMSE between estimated state and ground truth for yaw rate (top row), velocity (middle row), and position (bottom row). The boxes indicate the 25th and 75th percentile as well as the median.

this experiment, followed by the IMM and the EKF+YA. The standard EKF is not able to track the vehicle accurately during the maneuvering phase as expected.

Real World Ground Truth

In the following experiment, the virtual point cloud object is moved along real trajectories, recorded from the ego-vehicle at different urban intersections. This means, the simulation environment is augmented by trajectories that have been driven in reality.
Setup & Evaluation Criteria: Analog to the synthetic trajectories, the virtual point cloud object is moved along the real trajectories, including 18 left turn maneuvers, 19 right turn maneuvers, and 20 straight crossing scenarios. These trajectories correspond to the ground truth in this experiment.

For each trajectory, the root mean squared error for the yaw rate, $\mathrm{RMSE}_{\dot{\psi}}$, and velocity RMSE_{v}, as well as the RMSCE, indicating the outer pose error, are computed.
Observations: Fig. 4.9 shows the distribution of the resulting 57 error measures for the different filters in terms of the median and the 25th and 75th percentile (boxes). The whiskers indicate the first and (clipped) 99th percentile. As can be seen, the median

$\mathrm{RMSE}_{\dot{\psi}}$ over all runs is significantly reduced by all proposed extensions compared to the original EKF approach (Fig. 4.9(a)). The IMM and EKF+OR approach yield almost similar results and perform slightly better than the EKF+YA.

The RMSE_v of the velocity estimates is statistically slightly reduced by the IMM and EKF+OR approach, especially at the turn maneuvers. The median RMSE_v of the EKF+YA filter is also reduced for the turn maneuver sequences, while it is slightly increased for the straight-line trajectories.

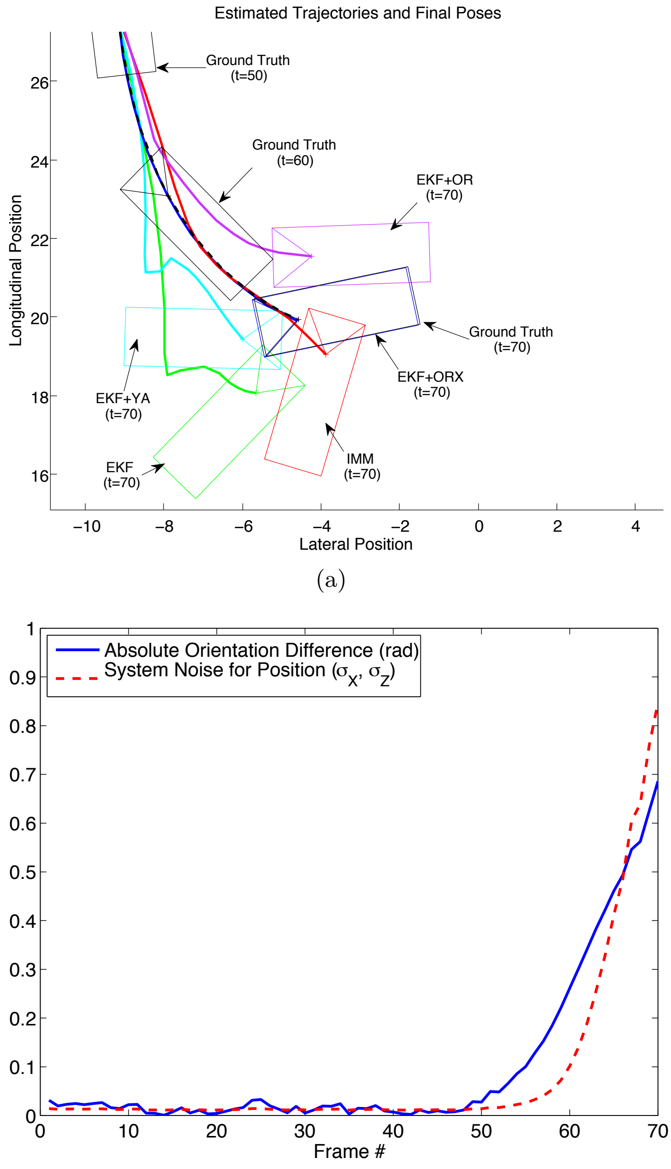
The improvements in the estimated motion parameters directly affect the pose error, which is also decreased for all extensions (Fig. 4.9(g)). Especially if only the subset of trajectories including a turn maneuver is considered for evaluation (second column). On the other hand, even for the straight motion trajectories (third column), improvements are achieved. The parametrization with much lower system variances at the non-maneuvering mode of the IMM filter is beneficial at low dynamic scenes as can be seen at Fig. 4.9(i). The EKF-OR filter equals the configuration of the EKF filter at straight motion, thus, the results do not differ much. Modeling of the yaw acceleration in the EKF-YA leads to larger deviations of the error at non dynamic scenes. The filter has to compensate for small errors in the orientation estimate by adapting the yaw acceleration, which typically leads to oscillation effects that increase the $\mathrm{RMSE}_{\dot{\psi}}$.
Conclusions: The good performance of the EKF+OR filter on the synthetic data is approved on the real-world trajectories and does not show a degradation at straight line motion. The same applies for the IMM filter, which yields almost equal performance in this experiment at significantly less computation time.

4.2.8. Filter Behavior at Extreme Maneuvers

So far only maneuvers that typically occur at urban intersections have been considered. In extreme situations, such as skidding on black ice, the assumption that the vehicle's orientation is aligned with the moving direction is violated. Although these extreme situations are out of scope of this thesis, the following experiment should demonstrate the limits of the proposed approaches and give an outlook on a promising solution for dealing with unexpected maneuvers.

A skidding trajectory is simulated by introducing an external force in terms of a lateral velocity component of 6 m/s, pushing the oncoming vehicle toward the observer while turning left with a constant yaw acceleration of 6 rad/s^2. As can be seen in Fig. 4.10(a), all filters discussed above fail to estimate the unexpected trajectory correctly since it does not agree with a circular path motion model.

However, the model-free oracle is able to follow the side-slipping vehicle very accurately. This leads to the idea of using the oracle not only to control the system noise of the yaw rate, but also for the position (EKF+ORX filter). The deviation between actual moving direction and object orientation, estimated by the oracle, is used as *skidding detector* (see Fig. 4.10(b)). Larger deviations indicate a violation of the circular path motion model. In this case, the system variances for position are increased using the same mechanism as introduced for the yaw rate. With large system variances for position, the vehicle is able to shift in arbitrary direction independent of the current orientation. The resulting final pose estimated by the EKF+ORX approach almost perfectly fits the ground truth, demonstrating the potential of this approach.



(a)

(b)

Figure 4.10.: (a) Estimation results of skidding trajectory. The EKF-ORX approach perfectly reconstructs the ground truth trajectory and final pose, while all other approaches considered so far fail. (b) Absolute difference between object orientation and moving direction used for adaptation of the system noise.

4.2.9. Summary of Simulation Results

The experiments have shown that the considered multi-filter approach outperforms all single filter approaches in **accuracy**, **precision**, and **consistency**. Among the single filters, the unscented Kalman filter does not reach the same tracking performance as the standard extended Kalman filter. A few iterations of the EKF further improve the tracking results at the considered scenarios.

The robust Kalman filter with adaptive gating and linear reweighing of measurements as proposed in Sec. 3.7 yields the best tracking performance if there are **gross outliers** in the measurements. In general, the outer accuracy increases with the number of contributing points. The pose estimation bias is smallest if the point positions are estimated simultaneously with the pose and motion parameters within one state vector. However, refining the points outside the filter by weighted averaging yields also acceptable results and is significantly faster compared to the simultaneous estimation.

All of the proposed extensions for tracking of maneuvering vehicles have improved the results at typical **turn maneuvers** without loosing performance at straight-line motion and without manual parameter tuning. The error in the estimate of yaw rate and position could be significantly reduced compared to the original single EKF approach. The IMM approach yields the best compromise between tracking accuracy and computational complexity. Since the IMM filters can be run in parallel on multi-processor architectures, the additional load reduces to the mode mixing and probability update.

A general problem of the filter design is that in most situations the filter is either too optimistic or too pessimistic, as indicated by the **consistency** tests. The system noise parameterization tries to handle the trade-off between smooth estimates and quick reaction times to maneuvers. Maneuvers can start suddenly, i.e., the system variances must not be too small. On the other hand, one does not want to be too sensitive to noisy measurements by strengthening the system model, i.e., the system variances must not be too large as well.

The best solution to overcome this problem is an adaptive system noise model, which is controlled depending on the current scene, as exemplarily demonstrated by the **oracle** approach. The computation of the oracle is more costly compared to the Kalman filter, even if the sparseness in the involved matrices is exploited, and depends on the considered window size. However, the skidding experiment has shown the potential of the oracle approach not only to improve the tracking at standard turn maneuvers, but also to detect extreme situations that are of special interest with respect to collision avoidance systems.

The EKF-YA filter is an alternative if **computation time** is critical and only a single core processor is available. The additional computation time compared to the original EKF approach is negligible, while the results show a significant improvement at turn maneuvers.

In the next section, the EKF-YA filter will be combined with a constant yaw rate filter via the IMM framework to the *IMM-2* variant as proposed in Sec. 3.11.2. This variant combines the advantages of a fast maneuver detector with the smoothness and stability of the lower order model.

4.3. Artificial Sequences

In the previous section, the simulation environment was based on 3D point clouds that have been projected onto a virtual image plane. The point correspondences have been known for all time steps, i.e., there have not been any (self-)occlusions or wrong correspondences. Furthermore, the measurement noise has been controlled and known for each point.

In this section, a more realistic simulation environment is used that generates artificial stereo image sequences of a virtual 3D road scene using **ray-tracing techniques** (POV-Ray[1]). The cars in the scene are moved based on a realistic physics engine, incorporating information on the object's mass, wheel properties, steering angle, gas pedal, etc. The object pose and motion state (velocity, yaw rate, acceleration,...) is known for each time step. In addition, ground truth depth and motion information for each point in the images is available. However, this information can also be extracted from the image sequences with the same methods as used in the real world system, e.g, using **SGM stereo** and a **KLT feature tracker**. The advantage of the latter is that one yields realistic reconstruction errors.

In the artificial scenes the points contributing to the object point model are assigned to an object at runtime using the approach proposed in Sec. 3.9, i.e., the system has to deal with a varying number of points and outliers in the data. Outliers are detected and rejected based on the adaptive gating (see Sec. 3.7). Object tracks are initialized using the image-based initialization method as introduced in Sec. 3.8.1.

The remainder of this section is organized as follows. First, the filter configuration considered for the artificial scenes is introduced. Then, **two experiments** including an intersection left turn and a right turn maneuver are presented. The results are summarized at the end of this section.

4.3.1. Filter Configuration

Two variants of the IMM filter as proposed in Sec. 3.11.2 are compared in the following experiments. The configuration of the system noise matrices is given in Table 4.3.

The **IMM-1** filter (one motion model, different variances) is configured only slightly different from the setup in Sec. 4.2.7. Here, the variances of the motion parameters for the stationary mode are increased to prevent a frequent switching of the modes at the artificial sequences. Otherwise almost any changes of the motion parameters are realized via the dynamic mode. This effect has not been observed in the simulation environment.

The **IMM-2** filter combines two motion models (AVCY and AVAY, cf. Sec. 2.5.3). The AVCY model is parametrized in a way that it covers mainly stationary processes, while the AVAY model is designed for more dynamic maneuvers, especially turn maneuvers. At this, the AVCY model assumes a constant yaw acceleration of zero, while the AVAY filter enables *directed* changes of the yaw rate via the yaw acceleration. The parametrization of the system noise matrix of the AVCY model, $C_{ww}^{(stat)}$, allows for slow changes in acceleration ($\sigma_{\dot{v}} = 0.1$ m/s^2) and minor changes of the yaw rate ($\sigma_{\dot{\psi}} = 0.01$ rad/s).

[1] http://www.povray.org/

		σ_X	σ_Z	$\sigma_\psi, \sigma_\beta$	σ_v	$\sigma_{\dot{\psi}}$	$\sigma_{\dot{v}}$	$\sigma_{\ddot{\psi}}$
		[m]	[m]	[rad]	[m/s]	[rad/s]	[m/s^2]	[rad/s^2]
IMM-1:	(stat)	0.01	0.01	0.01	0.01	0.01	0.1	-
	(mnv)	0.01	0.01	0.01	0.1	**1**	2	-
IMM-2:	(stat)	0.01	0.01	0.01	0.01	0.01	0.1	-
	(mnv)	0.01	0.01	0.01	0.1	**0.1**	2	**0.5**
IEKF:		0.01	0.01	0.01	0.01	**0.1**	**1**	-

Table 4.3.: System noise matrix configurations used in the following experiments. Differences between the different configurations are highlighted.

Upon starting, vehicles accelerate with approximately 1.5 to 3 m/s^2, and stop with -1.5 to -5 m/s^2. We thus allow changes of the acceleration by 2 m/s^2 via the AVAY system noise matrix $C_{ww}^{(mnv)}$. This is a compromise between the maximum expected acceleration and the objective to yield smooth velocity estimates (constant acceleration). The allowed changes of the yaw rate are increased to 0.1 rad/s in the dynamic model. In addition, the yaw rate depends on the estimated yaw acceleration, which is able to change by 0.5 rad/s^2 in this configuration.

The **single filter baseline** (IEKF with 3 iterations) has been manually optimized for the considered scenes to yield a good compromise in tracking performance and smoothness in the state estimate.

All filters are additionally provided with rotation point measurements in the following experiments (stereo-profile method, cf. Sec. 3.5.4). As in the simulation results, a constant measurement noise of $C_{vv}^{(P)} = \mathrm{Diag}(\sigma_u^2, \sigma_v^2, \sigma_d^2) = \mathrm{Diag}(0.5^2, 0.5^2, 0.5^2)$ is assumed for all points. This choice is motivated from error statistics comparing the computed feature tracks and stereo disparities on artificial scenes with ground truth available.

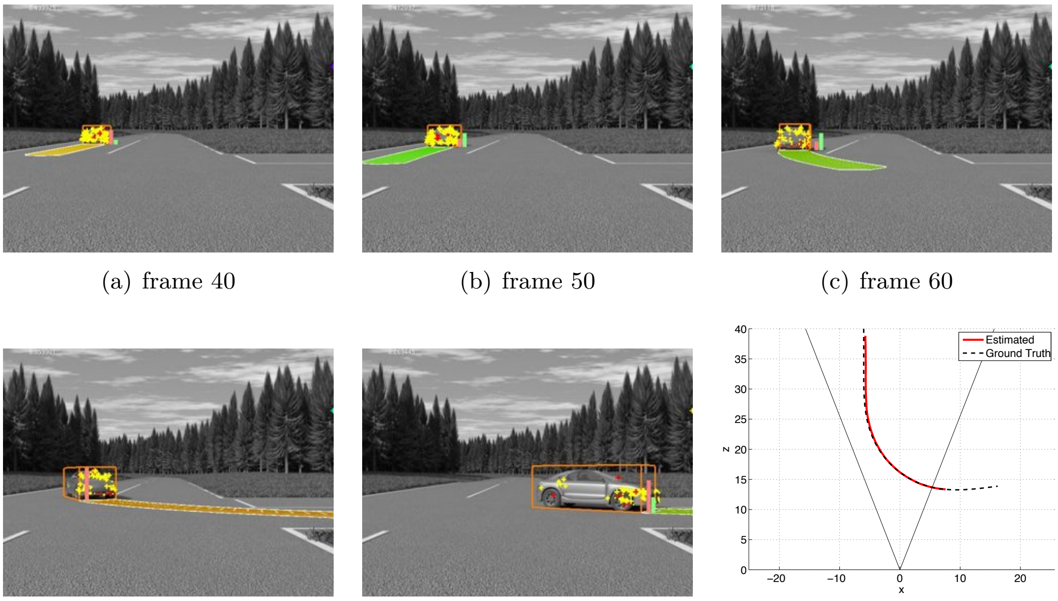
4.3.2. Intersection Left Turn

In the first experiment, the IMM filter tracking performance is evaluated based on an artificial scene containing a fast left turn maneuver. Depth and motion information is extracted from the image sequences with the same algorithms as in the real-world system (SGM stereo + KLT feature tracker).

Scenario: An oncoming vehicle is approaching from 60 m distance and quickly turning to the left at approximately 15 m distance in front of the stationary ego-vehicle. The velocity ($\approx$ 10 m/s) is slightly decreased during the turn maneuver.

Fig. 4.11 shows selected frames of the sequence with exemplary estimation results (IMM-2) superimposed. The estimated object pose and dimension is indicated by the bounding box, the motion parameters are encoded in the predicted driving path for the next second, visualized as a carpet on the ground. The tracked feature points, building the object shape model, are visualized by yellow crosses (red crosses=outliers).

The object is detected at about 40 m distance at frame 29 using the image-based method. The maneuver starts at frame 50. The predicted driving path already starts to



(a) frame 40 (b) frame 50 (c) frame 60

(d) frame 70 (e) frame 90 (f) estimated trajectory

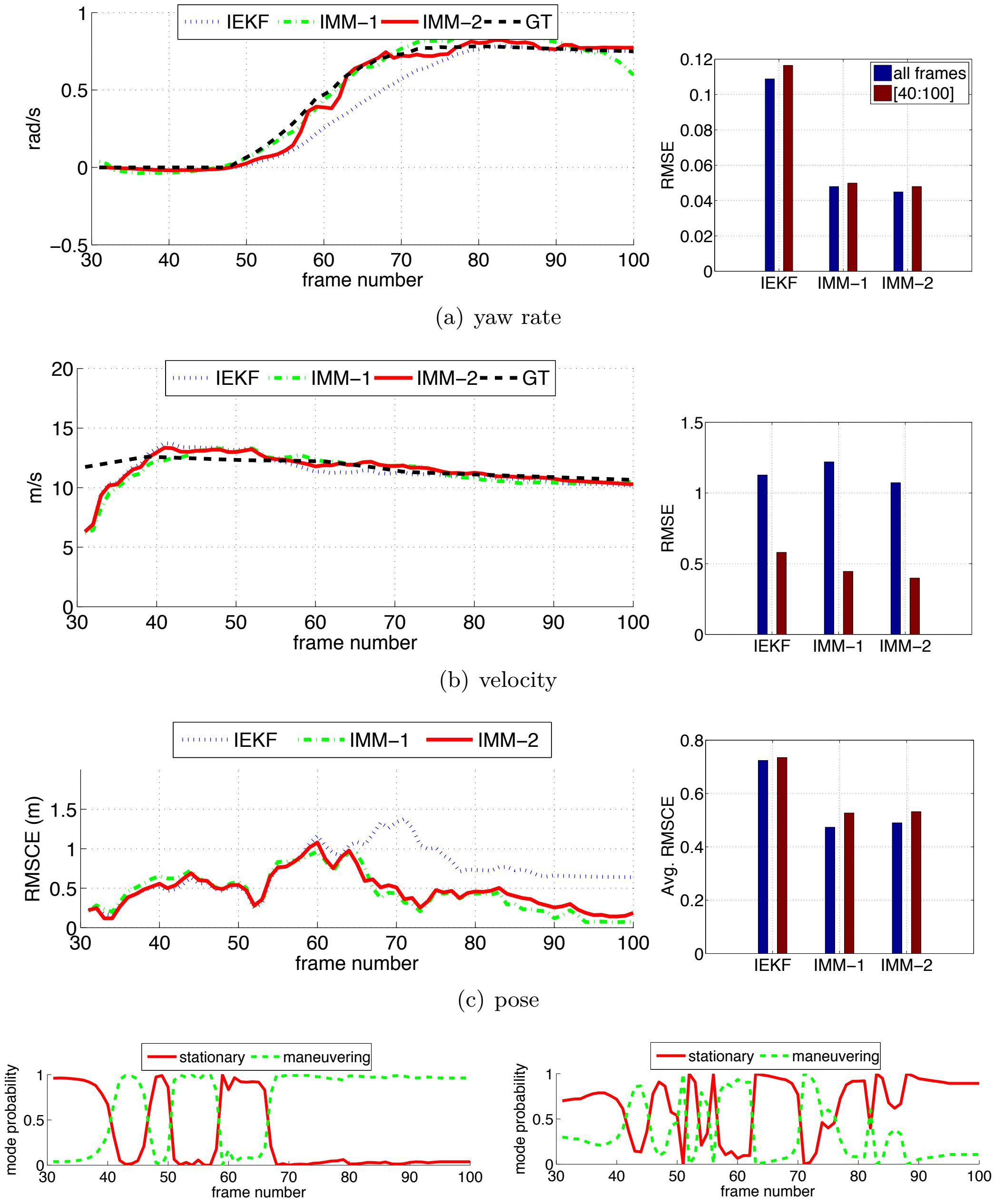
Figure 4.11.: Tracking results of a synthetic left-turn intersection scene (here with IMM-2 approach). The estimated motion state allows for accurate prediction of the object's driving path for the next second. The estimated trajectory of the cuboid center position (red solid line) precisely fits the ground truth (dashed black line) as shown from bird's eye view on the bottom right.

bend at this time step, although the vehicle has not significantly changed its orientation. Ten frames ahead, the predicted driving path clearly indicates the vehicle is turning to the left. Its destination is accurately predicted at frame 70 as shown in frame 90.

Evaluation Criteria: Analog to the simulation experiments, the tracking accuracy is evaluated based on the RMSCE, i.e., the distance of the four corners of the estimated object bounding box compared to the ground truth corner positions in ego- coordinates. The estimation accuracy of the motion parameters velocity and yaw rate is evaluated in terms of the RMSE$_{\dot{\psi}}$ and RMSE$_v$, respectively (cf. Sec. 4.1).

Observations: The evolution of the estimated and true motion parameters over time as well as the mean RMSE over two time intervals are shown in Fig. 4.12(a)-(b). The RMSCE as a function of time is shown in (c). The mode probabilities of the two compared IMM approaches are given in (d) and (e).

As can be seen, the estimated motion parameters of the IMM approaches approximate the ground truth very well. During the straight motion phase the single filter with AVCY motion model performs equally well as the IMM filters, however, it cannot follow the sudden increase of the yaw rate as the maneuver starts. The RMSE$_{\dot{\psi}}$ of the yaw rate compared to the ground truth (outer precision) is 0.048 for the IMM-1, 0.044 for the IMM-2, and 0.107 for the single filter approach. The RMSE$_v$ in velocity is 1.219



(a) yaw rate

(b) velocity

(c) pose

(d) IMM-1

(e) IMM-2

Figure 4.12.: Estimation results and RMSE of IMM filter variants for (a) yaw rate and (b) velocity compared to ground truth (dashed black lines) and a single filter approach (dotted blue line). The pose accuracy is evaluated based on the root mean squared corner error in (c). The IMM approaches clearly outperform the single filter approach. The mode probabilities are shown in (d) and (e).

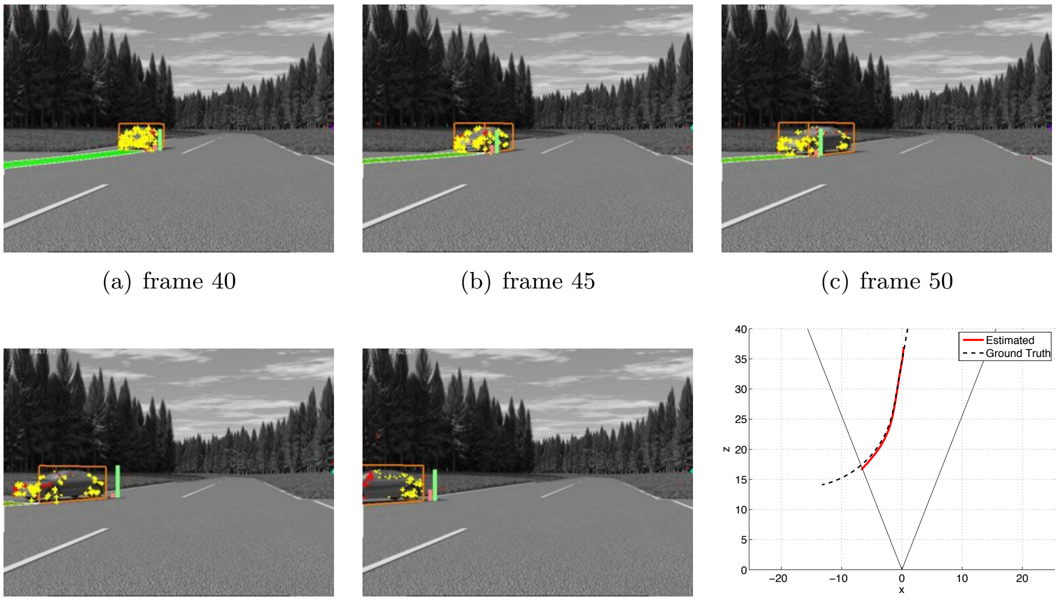
(0.445) for the IMM-1, 1.072 (0.399) for the IMM-2 and 1.127 (0.58) for the single EKF respectively. The values in brackets indicate the RMSE if the first 10 frames are ignored, reducing the influence of the significantly underestimated velocity at initialization.

The RMSCE decreases over time to below 0.08 (average 0.47 m) for the IMM-1 and below 0.15 m (average 0.49 m) for the IMM-2. The single filter approach results in a significantly larger error during the whole turn maneuver (average 0.72 m).

The IMM mode probabilities in Fig. 4.12(d) and (e) indicate the **interaction** of the two modes. The IMM-1 filter directly switches to dynamic mode at frame 40 (maneuver start) to be able to quickly adapt the yaw rate. Then, the filter toggles a few times between the two modes, indicating that there are phases of constant yaw rate (stationary mode). At the end of the sequence, the IMM-1 filter shows some overshooting and the filter remains in dynamic mode until the car leaves the visual field of the camera.

The mode probabilities of the IMM-2 approach are not that discriminating for the first 10 frames. As a result, the filter switches slightly later to the dynamic mode. However, the error of the yaw rate can be quickly compensated by estimating the yaw acceleration. To prevent an overshooting, the filter switches back to stationary mode (yaw acceleration zero). Climbing the ramp in a few *stairs*, induced by short phases of yaw acceleration followed by a constant yaw rate, is much faster compared to the single filter approach assuming a constant yaw rate for all frames. The advantage of the IMM-2 approach is that is shows almost no overshooting in the yaw rate estimate. However, the results are significantly less smooth compared to the IMM-1 approach.

Conclusions: From this experiment one can conclude that the good performance of the IMM approach in the simulation can be confirmed for the artificial sequences. In this sequence, there is only a minor difference between the two IMM variants. The IMM-2 yields a slightly improvement in the estimated motion parameters, while the IMM-1 approach yields more smooth results and a smaller average error in the estimate pose.



(a) frame 40

(b) frame 45

(c) frame 50

(d) frame 55

(e) frame 60

(f) estimated trajectory

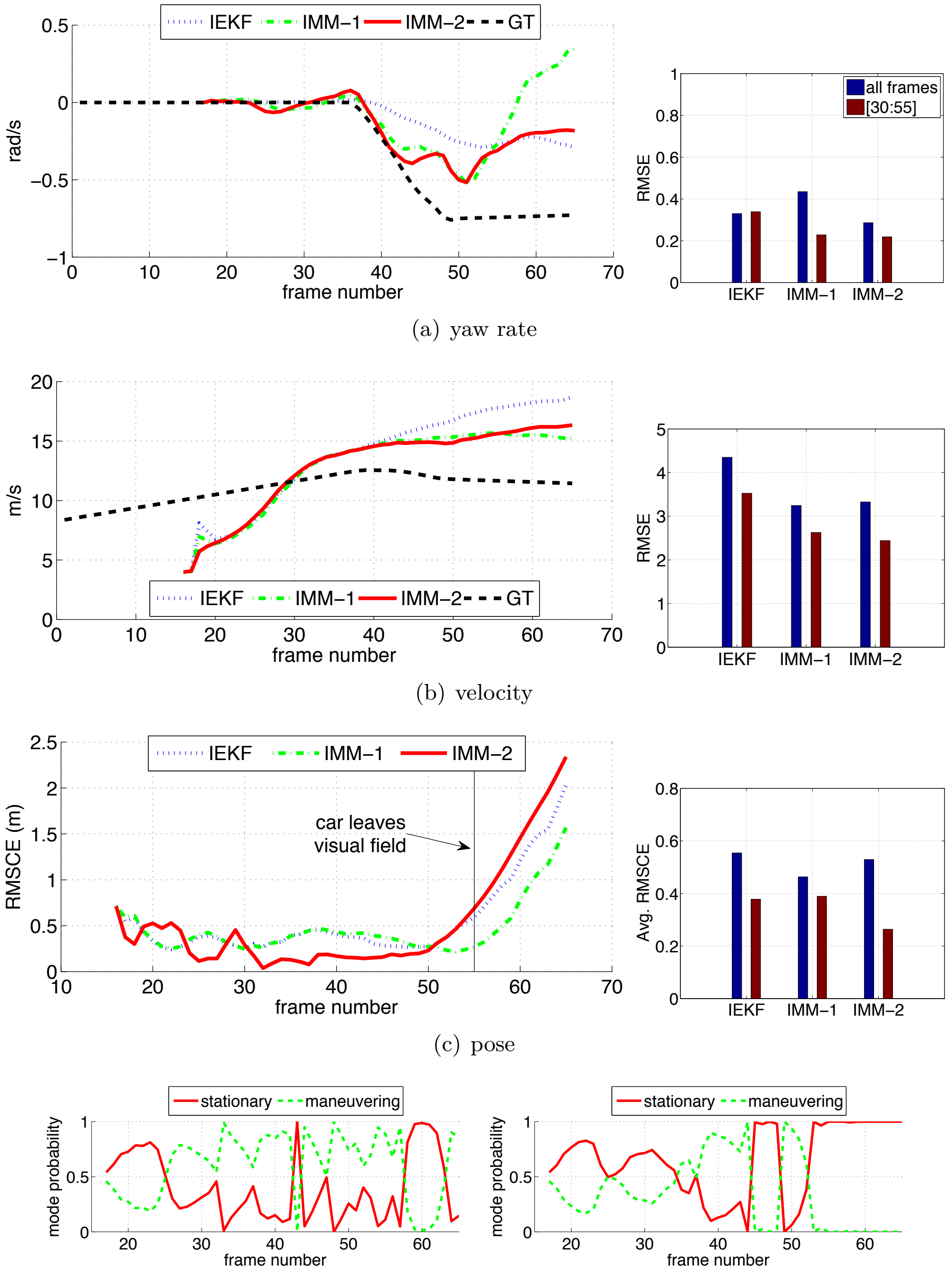
Figure 4.13.: Tracking results of synthetic right-turn intersection scene (here with IMM-1 approach and ground truth stereo measurements, see text).

4.3.3. Intersection Right Turn

The second artificial scene addresses a fast right-turn maneuver. The filter configuration and the evaluation criteria are analog to the previous experiment (see Table 4.3), i.e., a single filter approach is compared to the two IMM variants.

Scenario: An oncoming car is driving straight in the opposite lane towards the intersection for about 1.5 s (36 frames), and then turns quickly to its right (to the left from perspective of the camera). Fig. 4.13 shows selected images and a bird's eye view on the ground truth trajectory of this scene with exemplary estimation results superimposed. The velocity increases from 8.4 to 12.5 m/s until the beginning of the turn maneuver. While turning, the car slightly decelerates. The yaw rate quickly decreases in terms of an approximately linear ramp (constant yaw acceleration). At frame 55, the car starts leaving the visual field of the stereo system. The ego vehicle does not move in this scenario.

Observations: The object is detected at frame 16 in this experiment. The estimation results are visualized in Fig. 4.14 analog to the plots for the left-turn maneuver. During the straight line motion, the zero yaw rate is estimated well by all filters. The IMM-2 filter shows some minor overshooting. At the beginning of the turn maneuver at frame 36, the IEKF filter is too slow to follow the sudden yaw acceleration. The IMM filters differ only slightly in the response to the fast decrease of the yaw rate, but follow the ground truth (black dashed line) quickly for approximately half of the maneuvering phase. Then, all filters stagnate and the difference between estimate and ground truth



(a) yaw rate

(b) velocity

(c) pose

(d) IMM-1

(e) IMM-2

Figure 4.14.: Estimation results and IMM mode probabilities for the right-turn maneuver analog to Fig. 4.12. Here, SGM stereo disparities are used as measurements.

increases until the car leaves the visible field. The RMSE$_{\dot{\psi}}$ of the yaw rate estimate over the whole sequence is lowest for the IMM-2 approach in this experiment, but differs only marginally if only the time steps between frame 30 and 55 are considered.

The velocity is again initialized too low. The acceleration required to compensate for the error in velocity leads to an overshooting, which coincides with the beginning of the maneuver. Other than for the left-turn maneuver, the estimated velocity does not converge toward the real value in this experiment for any filter. The RMSE$_v$ is approximately equal for the two IMM variants, while the RMSE$_v$ of IEKF is slightly larger.

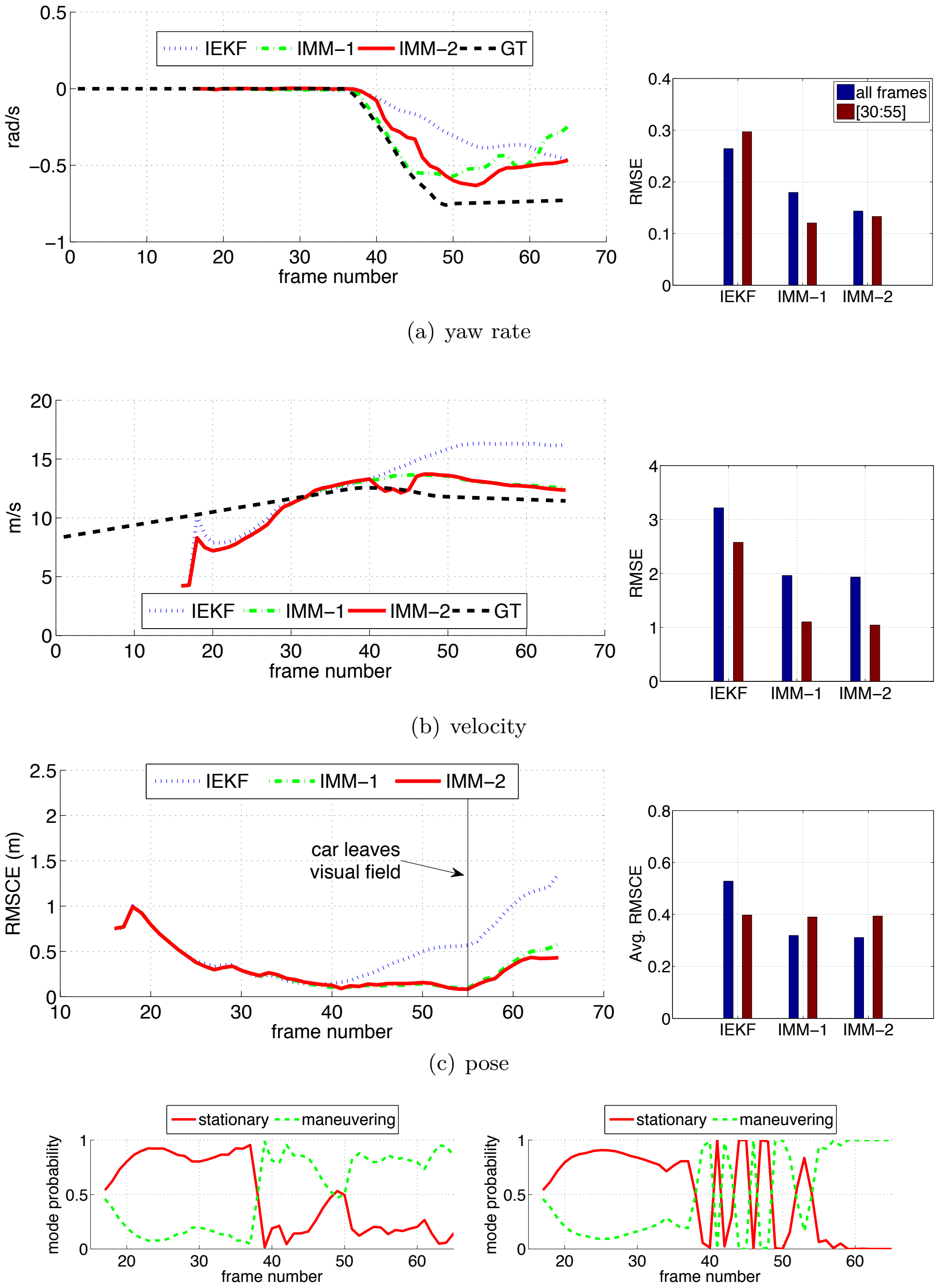
The inaccuracies of the motion parameters affect the estimated poses only of little account, indicated by the RMSCE, as long as the car is completely within the visible field of the camera. One explanation for that is that the pose depends on the object orientation (moving direction) and the position of the rotation point (center rear axle), which is geometrically constrained by additional measurements. As has been shown in Sec. 3.5.4, there are two ways to minimize the residual between prediction and measurements, i.e., altering the pose via the motion parameters or changing the rotation point position.

The IMM mode probabilities of the IMM-1 filter indicate that the stationary mode is more probable at the beginning, although the dynamic mode, allowing for a larger acceleration, has still a certain influence on the result. This is also the reason why the filter switches to the dynamic mode before the turn maneuver starts. It then remains in the dynamic mode, except for a short interruption, until the car leaves the visual field of the camera. The IMM-2 filter remains in the stationary mode until the onset of the turn maneuver. Switching to the dynamic mode at this time step allows for incorporating the yaw acceleration. As the filter returns to stationary mode about half of the maneuver, the constant yaw rate assumption leads to the stagnation from which the filter cannot recover.

Repetition with ground truth stereo: To investigate the source of these problems, the experiment is repeated with ground truth depth measurements, i.e., the measured distances correspond to the actual distances. The point tracks are still computed from the image sequences by feature tracking based on the KLT method. The assumed measurement noise for the disparity is not changed compared to the previous run, i.e., it is still assumed that the standard deviation of the disparity is 0.5 pixels. The results of the second run are summarized in Fig. 4.15.

As can be seen, the estimated parameters differ from the run with SGM disparity measurements. During the straight-line movement, the zero yaw rate is reconstructed even more accurately and the yaw rate RMSE$_{\dot{\psi}}$ is decreased by almost 50 %, although there is a certain delay in the yaw rate estimate of the IMM-2 filter, which switches to dynamic mode a few frames later (see mode probabilities). The error in the velocity estimate is also significantly reduced. This time, the estimates tend toward the ground truth (at least for the IMM-variants), reducing the RMSE to about 1 m/s if only frames 30 to 55 are considered. The RMSCE decreases continuously until the car leaves the visual field of the camera.

However, both the IMM-1 and the IMM-2 still show a certain stagnation at the middle



(a) yaw rate

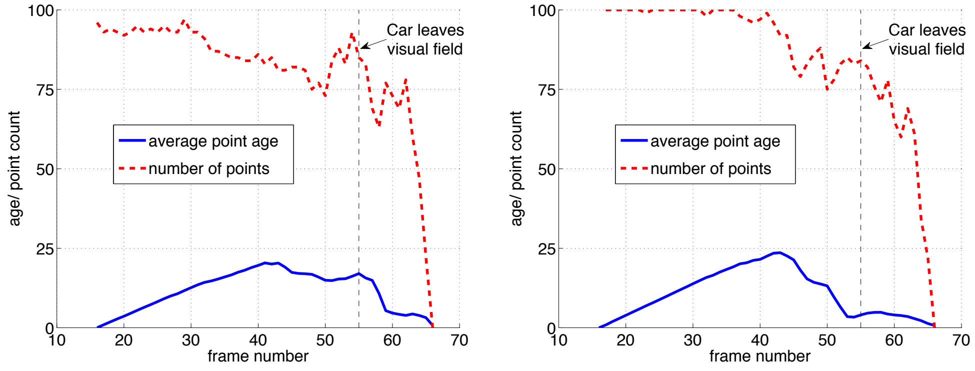
(b) velocity

(c) pose

(d) IMM-1

(e) IMM-2

Figure 4.15.: The estimation results are significantly improved if the depth measurements are replaced by ground truth stereo information.



(a)

(b)

Figure 4.16.: Development of the average point age over time as well as the total number of points contributing to the state update, (a) for the run with ground truth stereo disparities and (b) for the run with computed stereo disparities.

of the turn maneuver. This effect has also been observed in the previous run with real stereo measurements and is not fully understood by now. One possible explanation for that effect can be derived from the number of points contributing to the object model. Fig. 4.16 depicts the average point age for a given time step, i.e., the average over all points that have contributed to update the current filter state, as well as the total number of these points.

The plot shows that until about frame 42, the average point age (number of consecutive time steps this point has been part of the model) increases linearly with time, while the total number of points decreases slightly. This means there are a few outliers in the point data, but most points are observed and confirmed over the whole time interval. After this point in time, the average feature age decreases significantly, indicating that there have been several new points added to the shape model. These points are added during the turn maneuver. This is not critical as long as the estimated object pose is correct. Any deviations lead to the effect that points are registered to a wrong position within the object model. This is still not a big problem, as long as these points have a certain uncertainty. However, if many points get lost at one time step that have been observed for a longer time period, and these points are then replaced by new, uncertain points, these points get a non negligible influence on the estimation result. This problem is enforced in case of real (noisy) 3D point measurements as has be seen at the results with SGM stereo.

Conclusions: Loosing many points at the same time during a maneuvering phase is critical and leads to larger errors in the state estimate. Both the IMM-1 and the IMM-2 approach have this problem. However, the multi-filter approaches still outperform the single filter approach significantly at this experiment.

4.3.4. Summary of Artificial Scene Results

The experiments with artificial scenes have shown that the proposed system is able to reliably estimate the pose and motion parameters of a given object under realistic conditions. This includes depth and motion information extracted from the image sequences with more realistic error distributions, outliers, and a varying number of object points. The object hypothesis have been generated in an automated fashion, which leads to larger deviations of the initial object state compared the ground truth, especially in the velocity estimate. This is a significant difference to the starting conditions of the simulation experiments in Sec. 4.2.

The two IMM variants outperform the single filter approach as in the simulation at the considered scenarios. The IMM-2 approach, incorporating also the yaw acceleration in the dynamic mode, yields slightly better results compared to the IMM-1 approach, in which the different modes differ only in the variances of the yaw rate. However, the IMM-1 filter results in more smooth changes of the mode probabilities, which directly transfers to the estimated parameters which are also more smooth.

The problems arising in the right-turn scene are acceptable in practice, since the car is tracked very accurately as it is approaching the intersection and the estimated yaw rate clearly indicates that the car is turning to the right. This is a very useful information for predicting the driving path of the oncoming car. Furthermore, the pose error is quite small as long as the car is in the visible field, i.e., the object boundaries are reconstructed very well under consideration that the given turn maneuver is in fact racy.

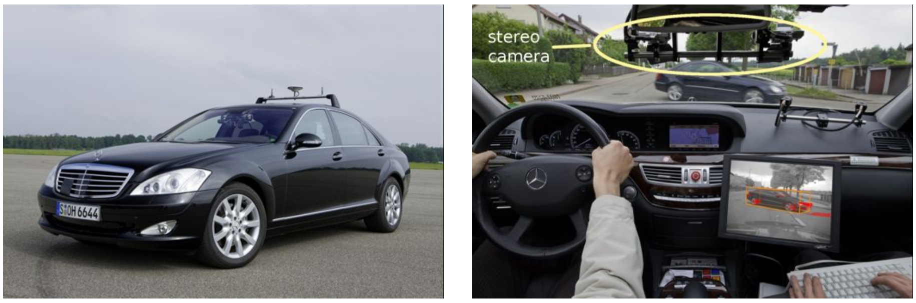
4.4. Real World Results

This section presents results from real world stereo sequences. The following questions should be answered:

- How accurate are the resulting pose and motion estimates at controlled real-world scenes? Are these results comparable to the artificial scenes?
- How does the system perform in real traffic scenes?
- What influence has the initialization method on the tracking result?
- What are the limits of the approach?
- What are the computation times of the real-world system?

The section is divided into three parts:

First, the system is tested based on three maneuvers that have been recorded in a **controlled test drive environment**. The advantage of these scenes is that both the object vehicle as well as the ego-vehicle are driven by a robot and, thus, there is ground truth information on the objects pose and motion state available allowing for a quantitative evaluation as in the previous sections.



(a) (b)

Figure 4.17.: (b) Mercedes-Benz demonstrator car used for testing of the proposed methods. (b) The stereo-system is mounted behind the windshield.

In the second part, exemplary tracking results at **real traffic scenes** (urban intersections, country road, high way,...) are shown and evaluated in a qualitative sense due to the absence of ground truth information.

Finally, a critical discussion on the challenges and limits of the current system is given, including examples of scenes were the system failed to perform correctly, as well as an analysis of the computation time.

Opposed to previous experiments, in this section the ego-vehicle is also moving in several scenes. All results presented below are achieved in real-time, i.e., the state estimates can be computed on-line in a demonstrator car. The configuration of this test vehicle is briefly introduced in the following.

4.4.1. Demonstrator Car

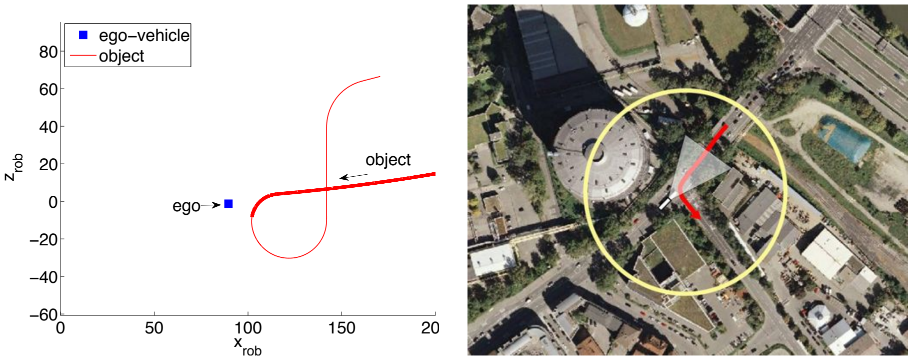
A Mercedes-Benz S-class car (see Fig. 4.17(a)) has been the platform for developing and testing of the proposed methods. This research car is a demonstrator vehicle for innovative driver assistance and safety systems.

It is equipped with a 0.3 m baseline stereo system behind the windshield that has a viewing range of approximately 42 degrees with 6 mm focal length lenses. The CMOS imagers have a resolution of 640×480 (VGA) and provide 12 bit gray-scale images. The stereo sensor is the main sensor used in the present work. In addition, a far range radar sensor and several near range radar sensors are available. Information of inertial sensors, such as wheel speed or the yaw rate, is available over a CAN interface. Furthermore, a differential GPS (DGPS) sensor allows for precise localization of the ego-vehicle.

The main processing unit is a recent high-end off-the-shelf Quad Core PC with a programmable graphics card. The stereo computation is done on a separate FPGA and does not require processing time on the CPU. See Sec. 2.3 for details.

4.4.2. Robot Experiments

In a controlled testing environment, the ego-vehicle (demonstrator car), as well as one *object* vehicle (Mercedes E-class) are driven by a robot along predefined trajectories. The actual motion state (velocity, acceleration, yaw rate), orientation, and the position



(a)

(b)

Figure 4.18.: Scenario 1: Slow turning vehicle in front of the stationary ego-vehicle. (a) Trajectory from bird's eye view. (b) Satellite view on the original intersection.

of a reference point of both cars is recorded during the maneuvers. The positions are obtained from a differential GPS (DGPS) system, i.e., a ground-based reference station broadcasts the difference between the less accurate satellite position and a fixed local reference system on the ground to yield a much more precise position. This *ground truth* data is available at 100 Hz and is synchronized between both cars. This setup enables testing of driver assistance systems reliably and reproducible, without risking human health in critical scenarios.

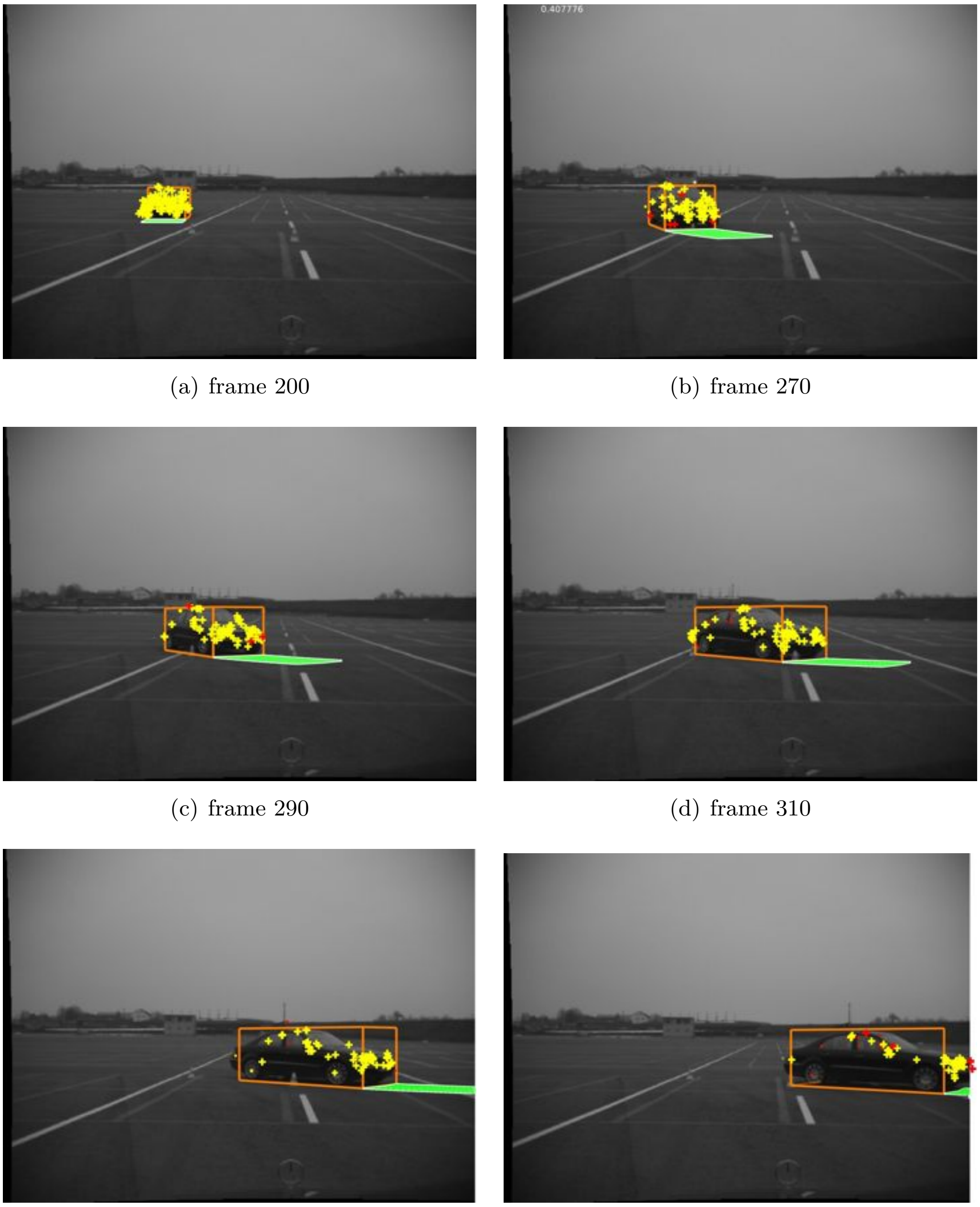
For evaluation of the proposed vehicle tracking system, three maneuvers have been defined. These maneuvers correspond to real intersection scenarios, i.e., the cars are moved on virtual roads and intersections that exist in reality. The particular road geometry has been acquired based on data available from the local land-registry. The first scenario is similar to the left-turn maneuver of the artificial scenes. In the second scenario, both the ego-vehicle and the oncoming object vehicle pass straight at a relative velocity of 80 km/h. In the third scene, the ego-vehicle is quickly turning to the left in front of the oncoming car which is stopping shortly before the ego-vehicle.

In all robot experiments, the IMM-1 filter configuration as proposed before is used for evaluation. A measurement noise of $\sigma_u = 1$, $\sigma_v = 3$, and $\sigma_d = 1$ is assumed. The slightly larger uncertainties are required to make the system less sensitive to errors in the ego-motion, e.g. non-compensated camera pitch movements lead to larger prediction errors of the vertical image coordinate.

Scenario 1: Left-Turning Object

In the first scenario, the oncoming object is driving as slow as possible (about 12 km/h) and turns to the left in front of the stationary ego-vehicle. It starts leaving the visual field of the stereo system at frame 350. Fig. 4.18(a) depicts the object trajectory and the ego-vehicle position with respect to the stationary local robot coordinate system. The original intersection, which has been the model for this scenario, is shown in (b).

Fig. 4.19 shows selected frames of this sequence with the tracking results superimposed. As can be seen, the estimated bounding box matches the actual object almost perfectly. The predicted driving path is slightly bend, indicating the vehicle is driving on a circular path. The estimated pose and motion parameters are compared to the ground truth analog to previous experiments in Fig. 4.20. The RMSE$_{\dot{\psi}}$ of the yaw rate estimate is 0.03 rad/s. Some problems occur as the object leaves the visible field, i.e., many points are lost at once. This effect has already been observed in the artificial scenes. The velocity estimate shows some overshooting. However, the slight deceleration at the beginning of the turn maneuver as well as the acceleration after reaching the maximum curvature point, is clearly visible in the velocity estimate. The velocity RMSE$_v$ is 0.39 m/s. The initial pose error quickly decreases at the beginning of the scene due to the geometric measurements of the rotation point position. It is slightly larger compared to previous experiments, which can be explained by the estimated box size. In this scene, the estimated object length is slightly too small, since parts of the heck are not completely included in the bounding box, which directly affects the RMSCE.



(a) frame 200

(b) frame 270

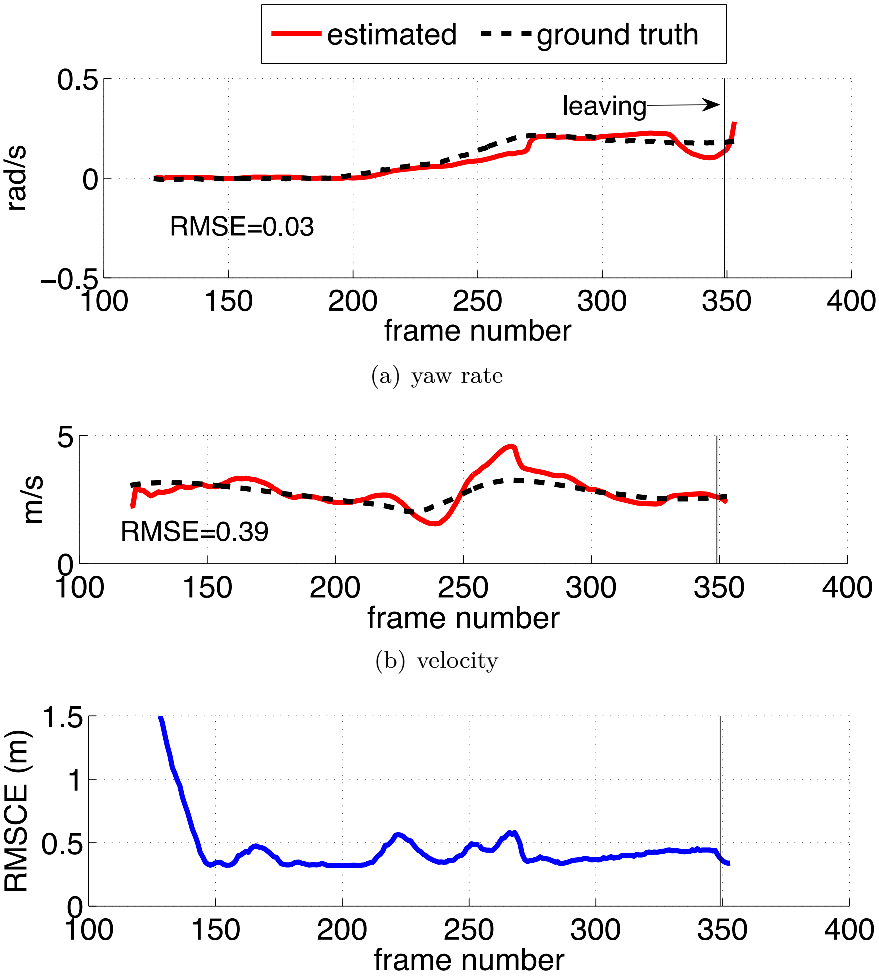
(c) frame 290

(d) frame 310

(e) frame 330

(f) frame 350

Figure 4.19.: Scenario 1 from the perspective of the ego-vehicle. The tracking results are superimposed. The bounding box indicates that the system is able to reconstruct the object pose very precisely during the maneuver.

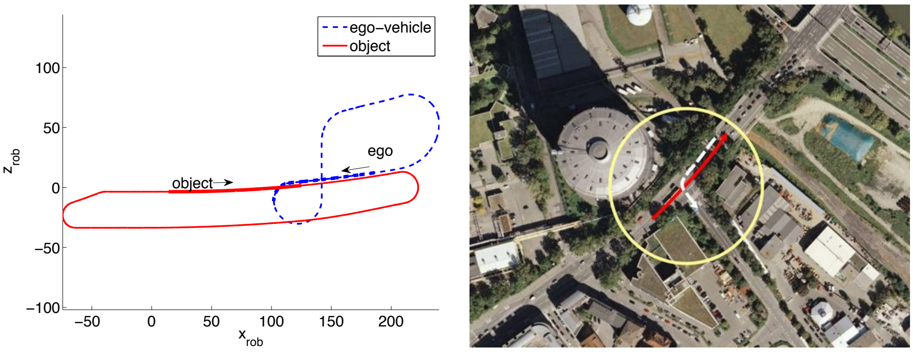


(a) yaw rate

(b) velocity

(c) pose

Figure 4.20.: Estimated motion and pose parameters for Scenario 1 compared to ground truth. The RMSCE shows that the pose estimate is systematically improved over time until the oncoming car starts leaving the visual field of the camera.



(a)

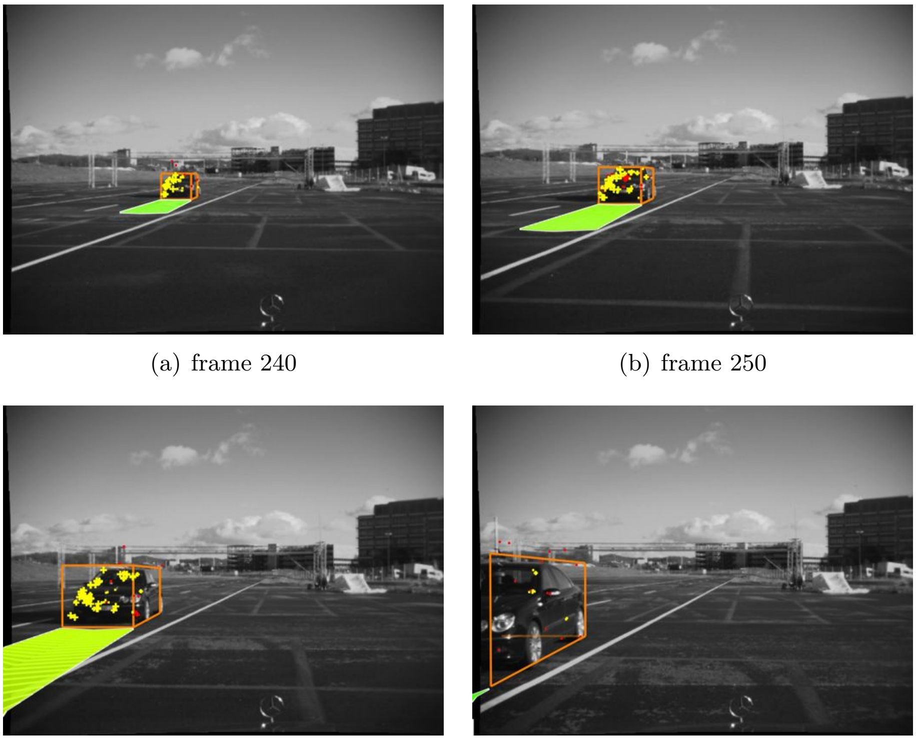
(b)

Figure 4.21.: Scenario 2: The ego-vehicle turns to the left at the intersection **after** the object has passed (straight-line passing). (a) Recorded ground truth trajectories in fixed local coordinate system. (b) Satellite view on the original intersection simulated in this experiment.

Scenario 2: Straight-line Passing

This experiment addresses a typical real-world scenario in which the oncoming vehicle passes the ego-vehicle while both vehicles move straight in their (virtual) lanes. The complete trajectories within the fixed local robot system as well as the original intersection are shown in Fig. 4.21. The relevant parts of the trajectories to be evaluated are marked in bold. In this scenario, the ego-vehicle is turning to the left after the oncoming vehicle has passed, i.e., the ego-vehicle turns to the left behind the oncoming vehicle.

Fig. 4.22 shows selected frames of the captured image sequence with the tracking results superimposed. The ground truth shows that the yaw rate of the oncoming vehicle is slightly larger than zero, i.e., the car is not exactly driving on a straight line. The filter also estimates a positive yaw rate. Between frame 250 and 262 the filter slightly overestimates the yaw rate. The $\mathrm{RMSE}_{\dot{\psi}}$ is 0.01 rad/s. The initial velocity estimate is slightly too large and converges toward the ground truth after eight frames. Shortly before passing, the filter starts to underestimate the velocity, which cannot be corrected anymore, since the vehicle leaves the visible field. The RMSE_{v} for the whole sequence is 0.9 m/s. The pose error is systematically decreased while the oncoming car is fully visible to below 0.15 m.



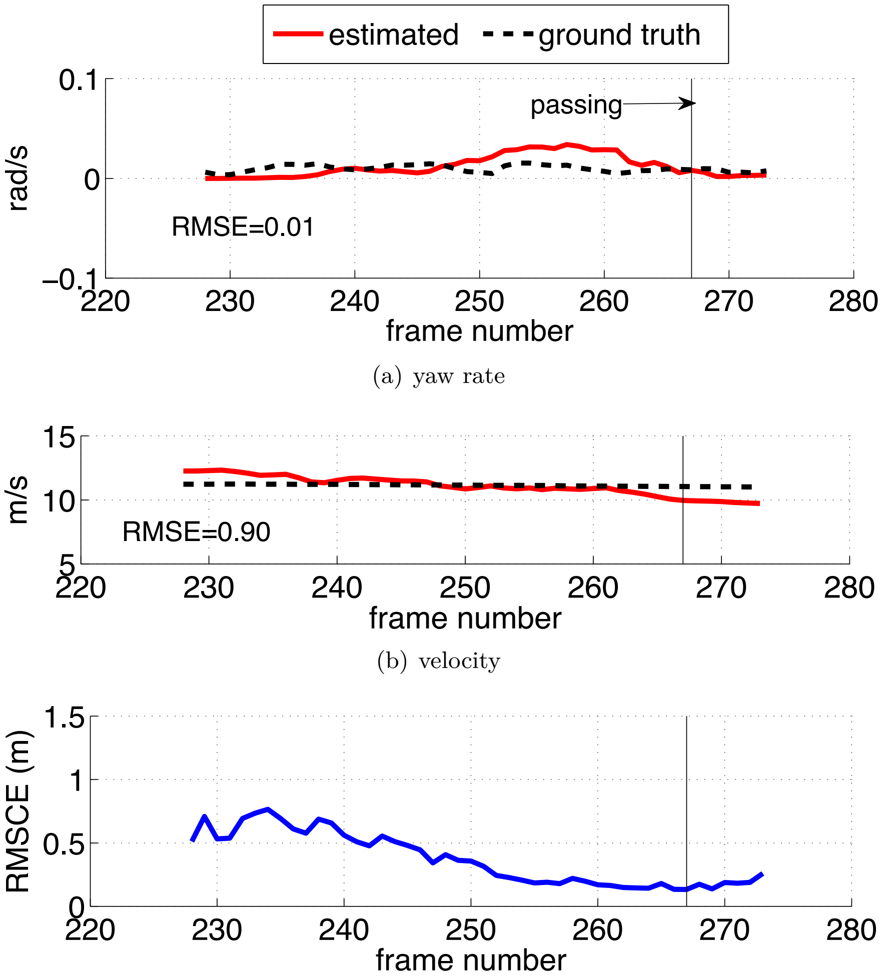
(a) frame 240

(b) frame 250

(c) frame 260

(d) frame 270

Figure 4.22.: Scenario 2 from the perspective of the ego-vehicle. The tracking results are superimposed. The bounding box indicates that the system is able to reconstruct the object boundaries precisely in this experiment.

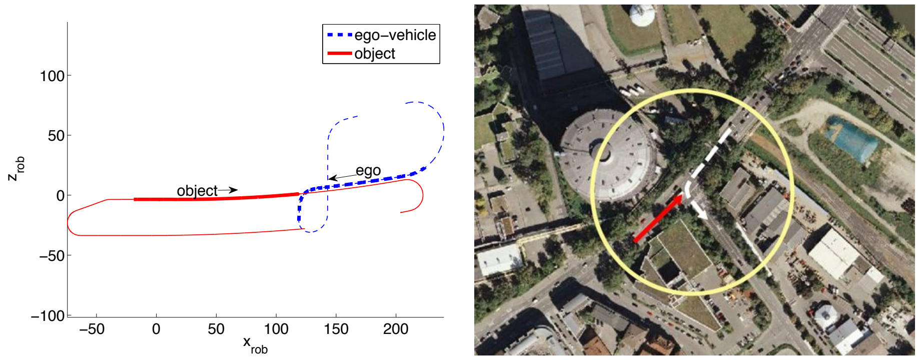


(a) yaw rate

(b) velocity

(c) pose

Figure 4.23.: Estimated motion and pose parameters for Scenario 2 compared to ground truth for scenario 2. The RMSCE shows that the pose estimate is systematically improved over time until the oncoming car starts leaving the visual field of the camera at frame 267.



(a)

(b)

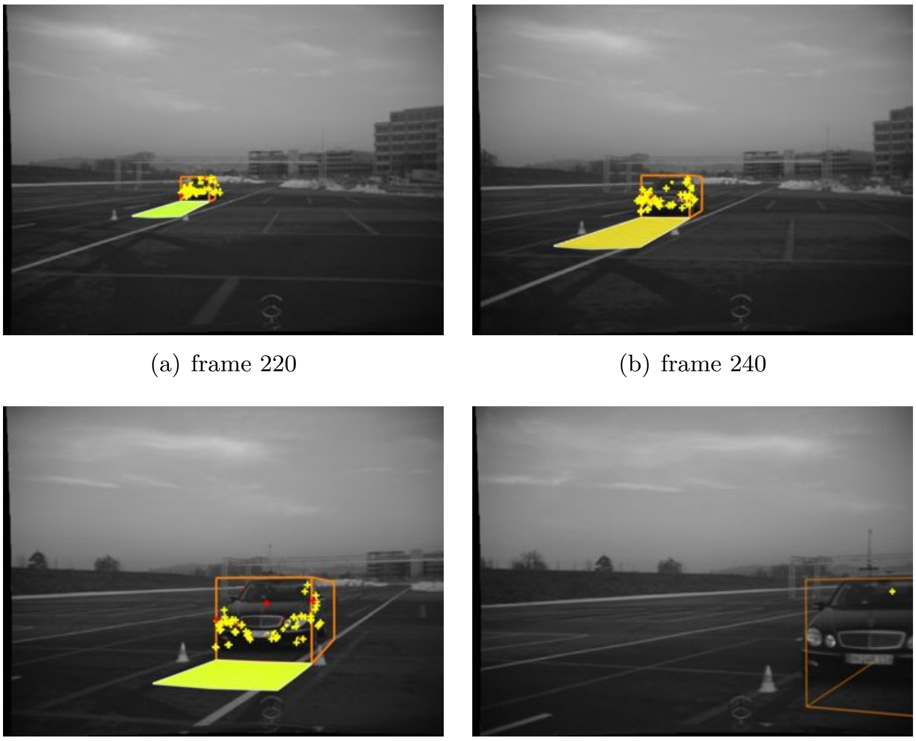
Figure 4.24.: Scenario 3: The ego-vehicle is turning to the left in front of the oncoming vehicle, which stops at the (virtual) intersection. (a) Recorded ground truth trajectories in fixed local coordinate system. (b) Satellite view on the original intersection simulated in this experiment.

Scenario 3: Left-Turning Ego-Vehicle

The third scenario contains a quick turn maneuver of the ego-vehicle in front of a stopping oncoming vehicle. The oncoming car approaches straight and decreases its velocity from 50 km/h to 0 km/h in about two seconds (mean acceleration of -5.2 m/s^2). The ego-vehicle has an average velocity of 20 km/h during the maneuver. The maximum yaw rate is 0.5 rad/s. Example frames of this sequence with the tracking results superimposed are shown in Fig. 4.25. The length of the predicted driving path clearly indicates that the vehicle is stopping. At frame 270, all feature tracks but one are lost due to the large displacements induced by the camera motion. Accordingly, the object pose at frame 270 is only updated based on the geometric measurements for the rotation point.

The ground truth yaw rate of the oncoming car in Fig. 4.26 is less smooth than the filtered yaw rate estimate. As in scenario 2, the slightly positive yaw rate indicates that the original road is slightly bend at this section, i.e., the car is not driving on an exact straight line. The RMSE$_{\dot{\psi}}$ is 0.02 rad/s over the whole sequence, although this scenario is highly dynamic. The velocity is initialized close to the ground truth. Between frames 210 and 230, the velocity is slightly underestimated, but follows the sudden deceleration very accurately. The total RMSE$_v$ is 0.89 m/s in this sequence. The error in the velocity also affects the pose error as indicated by the RMSCE. However, after frame 220, the RMSCE is systematically decreased to below 0.3 m. Since both the ego-vehicle and the oncoming car are highly dynamic in this scenario, this accuracy is substantial.

Conclusions of Robot Experiments The experiments have shown the real-world system reaches a comparable performance as on the artificial scenes. The resulting error measures are in the same order of magnitude.



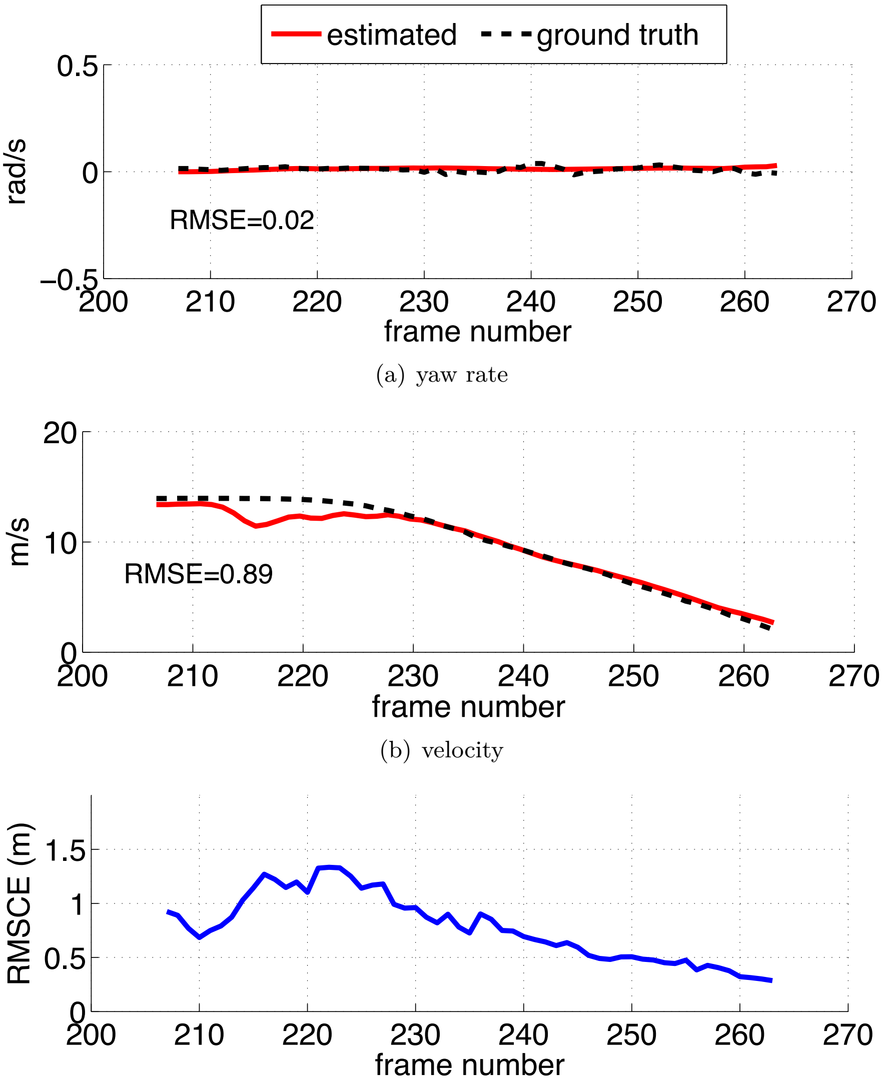
(a) frame 220

(b) frame 240

(c) frame 260

(d) frame 270

Figure 4.25.: Scenario 3 from the perspective of the ego-vehicle. The ego-vehicle is quickly turning in front of the stopping oncoming car. The tracking results are superimposed. The shortening driving path indicates that the object decelerates.

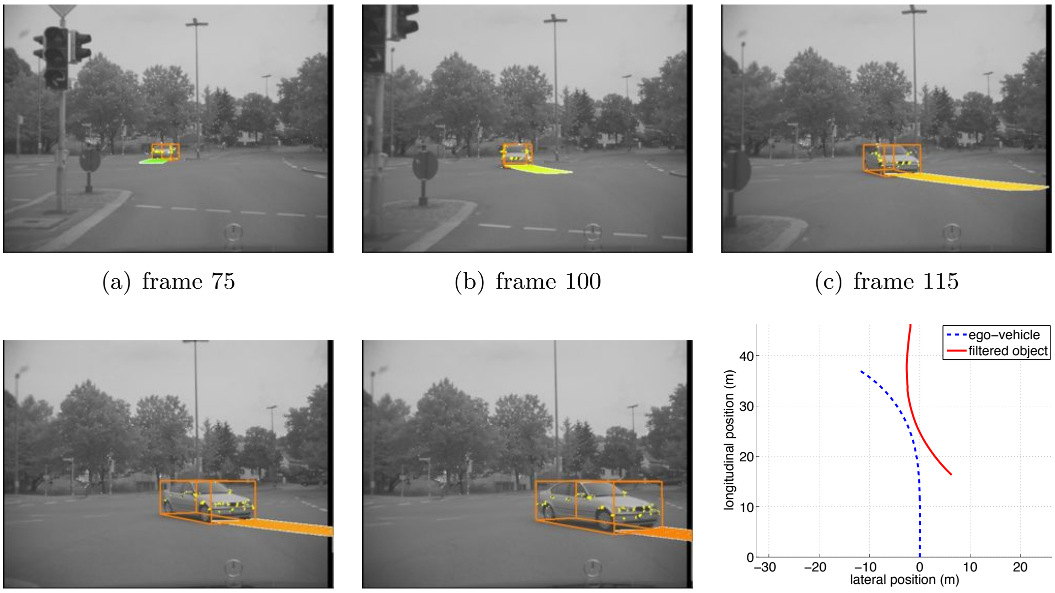


(a) yaw rate

(b) velocity

(c) pose

Figure 4.26.: Estimated motion and pose parameters for Scenario 3 compared to ground truth. The RMSCE shows that the pose estimate is systematically improved over time.



(a) frame 75

(b) frame 100

(c) frame 115

(d) frame 125

(e) frame 130

(f) trajectory

Figure 4.27.: Tracking result of a left turning oncoming vehicle at an urban intersection. The crosses superimposed mark the feature positions used in building the object model. The resulting trajectories of the ego-vehicle and the object are shown in (f).

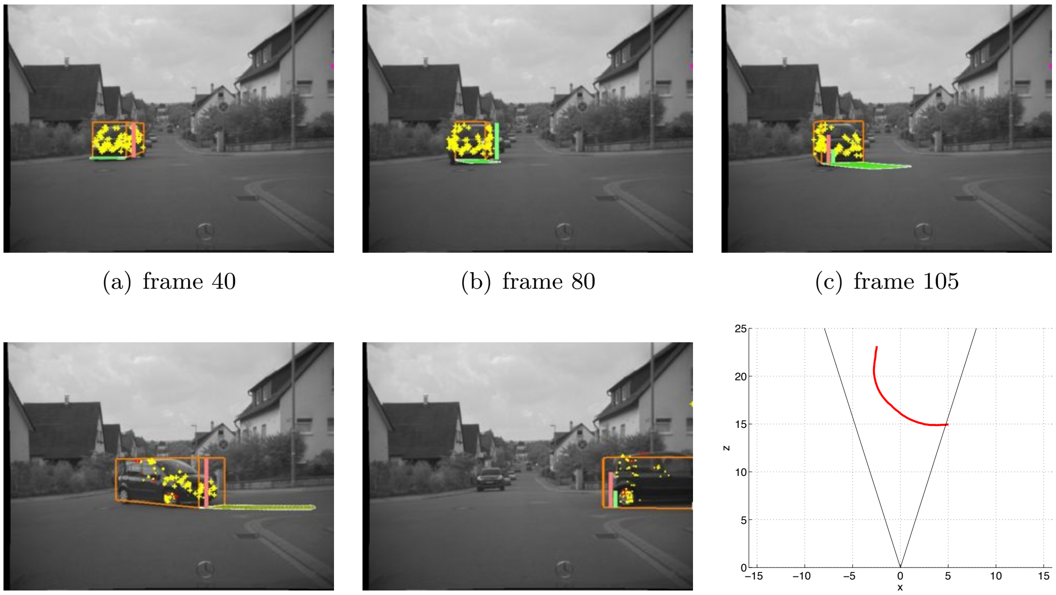
4.4.3. Oncoming Traffic at Intersections

Intersection scenarios can be arbitrary complex and challenging. In this section it will be focused on two representative scenes with turn maneuvers. Only scenarios with isolated object are considered here. The difficulties arising at dense traffic scenes will be addressed in Sec. 4.4.6.

American Left Turn Maneuver

Fig. 4.27 shows selected frame of a scene where both the ego-vehicle and the oncoming vehicle turn left at the same time (American left turn rule). Similar to the previous results, the image overlay shows the filtered object state in terms of pose (bounding box), object model (marked features on the object surface), and motion state (carped on the ground indicating the predicted driving path for the next second). The resulting object trajectories are shown in Fig. 4.27(f). As can be seen, both the driving path as well as the object pose and boundaries are reconstructed extremely well.

Assuming the driver in the ego-vehicle wants to drive straight ahead in the example scene above, a free-space analysis based on static occupancy grids, for example [Badino et al. 2008], would result in an obstacle-free driving corridor for the ego-vehicle up to frame 125 (see Fig. 4.27(a)-(d)). Obviously, ignoring there is another vehicle at the intersection that will be crossing the ego-vehicle's path can lead to potentially critical situations. Thus, the proposed system provides valuable information for predicting



(a) frame 40 (b) frame 80 (c) frame 105

(d) frame 130 (e) frame 160 (f) estimated trajectory

Figure 4.28.: Tracking results of a vehicle turning through a small radius after stopping.

collisions.

Left Turn After Stopping

The next scene contains an oncoming vehicle stopping at the intersection before it turns to the left. This scene is particularly challenging, since the vehicle turns through a small radius and there is both an acceleration in the longitudinal direction as well as a sudden increase of the yaw rate. As in the simulation, there are also self occlusions during the turn maneuver. The tracking results (IMM-2 filter) are shown in Fig. 4.28.

As can be seen, the system is able to accurately track the object during the whole maneuver. The estimated pose parameters are shown by the bounding boxes, the estimated motion parameters are shown in Fig. 4.29.

At the beginning, the filter is in stationary mode (constant negative acceleration) and toggles to maneuvering mode only shortly at frame 60 to reduce the acceleration as the vehicle stops. At the onset of the turn maneuver, the filter switches again to maneuvering mode at about frame 75 to be able to quickly develop an acceleration in the longitudinal direction as well as with the yaw rate. At about frame 105, the constant yaw rate model outperforms the constant yaw acceleration model, leading to another mode change. This is typically the point where the driver shifts up into the next gear, interrupting the acceleration for a short period (about one second). The stationary filter only allows for slow changes in acceleration, thus, the velocity is slightly underestimated for a couple of frames. At frame 160, the probability for maneuvering mode increases again, on the one hand to increase the velocity by increasing the acceleration, on the other hand to reduce the yaw rate to zero as the vehicle is going back to straight motion while it leaves the viewing field of the camera.

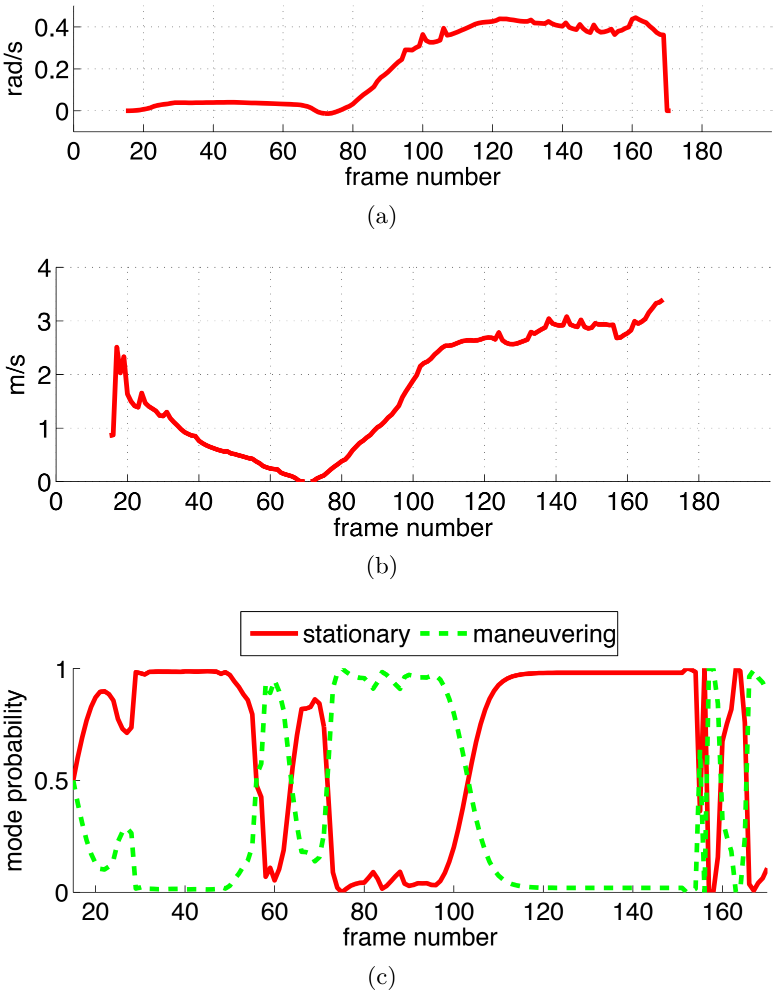


Figure 4.29.: Estimated motion parameters and IMM mode probabilities of the scene shown in Fig. 4.28. (a) yaw rate, (b) velocity, (c) IMM mode probabilities.

4.4.4. Oncoming Traffic at Country Roads

This experiment focuses on the different initialization methods (image-based and radar-based) proposed in Sec. 3.8 and the ability of the system to quickly yield reasonable estimates of the motion parameters.

Estimating the driving path of oncoming vehicles on country roads is extremely challenging due to relative velocities of 100-240 km/h, narrow curved roads as well as limited range of sight. In everyday situations approaching vehicles pass very closely, making it difficult to identify real critical situations. In country road curves, an estimate of the yaw rate of other traffic participants is essential as can be seen in Fig. 4.30. Using a common linear motion model, a collision between an oncoming vehicle and the ego-vehicle would be predicted. Only when the yaw rate of the oncoming vehicle is estimated, in fact, the situation can be correctly interpreted as uncritical.

Fig. 4.31 shows selected frames of a sequence with an oncoming vehicle, a Mercedes-

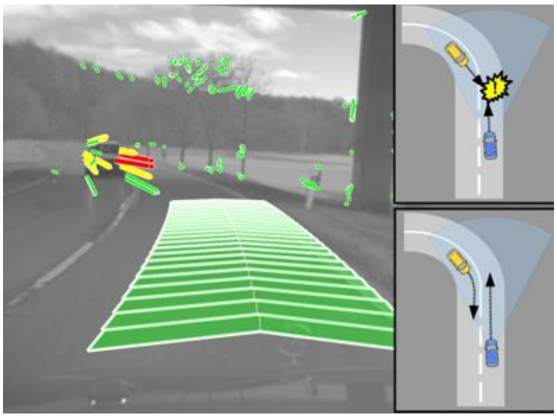


Figure 4.30.: Linear prediction of the driving path of an oncoming vehicle leads to false alarms in curves or turning maneuvers. Using an estimate of the other vehicle's yaw rate, it is possible to predict its real driving path accurately, and to prevent such false alarms.

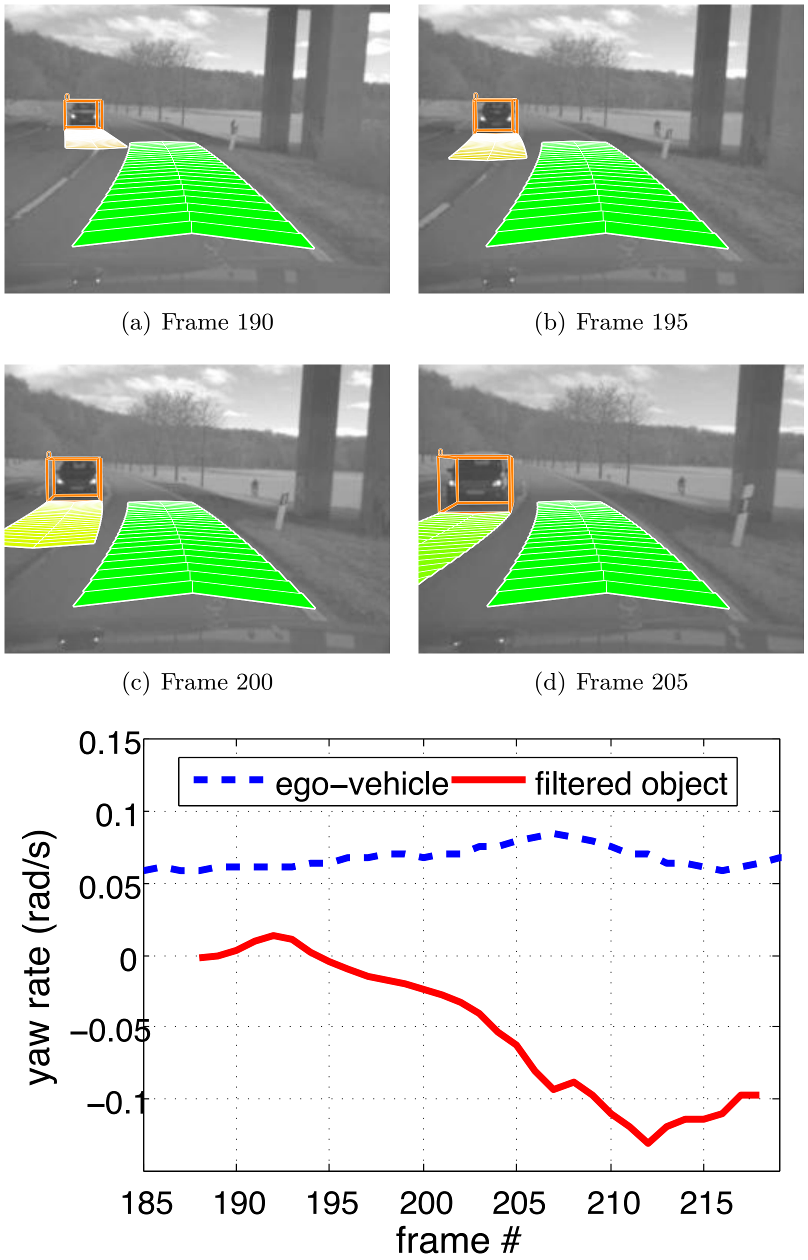
Benz B-class without any preparations, on a curved country road. It illustrates how the Kalman filter sequentially corrects the inaccurate initialization within the first 500 ms of tracking. In this experiment, a single filter IEKF with 3 iterations has been used. The overlaid carpets represent the predicted driving corridor of the ego-vehicle based on inertial sensors and the estimated driving path of the oncoming vehicle respectively.

Image-Based Initialization: At frame 188, the oncoming vehicle is detected at about 38 m distance using the purely image-based method as introduced in Sec. 3.8.1. The relative velocity v_{rel} is about 30 m/s.

Tracking the object over a few frames indicates that the object is not driving straight ahead (critical), but rather on a curved path (uncritical). With the assumed motion model, the motion of all observed points on the object can be best explained as a vehicle with a negative yaw rate. The estimated yaw rate of the oncoming vehicle is plotted in Fig. 4.31(e). It can be seen that the absolute value of the estimated yaw rate of the oncoming vehicle is slightly larger than that of the ego-vehicle, since the oncoming vehicle is driving on the inner lane of the curve on a smaller radius. It is important to state the system does not include any lane recognition component. The estimated yaw rate depends purely on the point motion and the underlying motion model.

Critically reflecting the results, it can be observed that there is not much more than a second left between object detection and passing. The filter requires 10-15 cycles (0.4-0.6 s) until the driving path is estimated well. In this experiment, both vehicles are driving at approximately 50-60 km/h. The speed limit on this road is 70 km/h. In practice, especially in potentially critical situations, one can expect even higher relative velocities.

Since the Kalman filter always requires a couple of cycles until convergence, initializing the filter as early as possible becomes even more important.



(a) Frame 190

(b) Frame 195

(c) Frame 200

(d) Frame 205

(e)

Figure 4.31.: (a) - (d) Selected frames of a real world scene with one oncoming vehicle captured in a left curve ($v_{\mathrm{rel}} \approx 30$ m/s) are shown. The driving corridor of the ego-vehicle based on inertial sensors is visualized in all images. It can be observed how the filter successively corrects the erroneous initial assumption of the oncoming object's driving path. (e) The filtered yaw rate indicates the oncoming vehicle is steering to the right, while the yaw rate of the ego-vehicle is consistent with a slight left turn.

Radar-Based Initialization: We have repeated the country road curve experiment above with a relative velocity of 44 m/s, i.e., each vehicle is driving at approximately 80 km/h. This time the radar-based initialization method as proposed in Sec. 3.8.2 is used and compared to the image-based method. In addition, radar velocity measurements are provided at runtime.

With the image-based method, this time the object is detected at 36-m distance. Due to the relative velocity only 13 cycles remain until the vehicle leaves the field of view of the cameras. The radar detects the object at a 52 m distance, which is detected 16 m earlier compared to the image-based method, corresponding to 7 additional image pairs to evaluate. In this scene the object is not in the field of view of the radar earlier due to the curve. On straight roads one can expect oncoming objects to be detected at significantly larger distance by the radar.

The estimated yaw rates are shown in Fig. 4.32(a). Both runs have been initialized with $\dot{\psi} = 0.0$ rad/s and converge toward approximately -0.1 rad/s. However, the yaw rate of the filter initialized by radar clearly indicates that the object is driving on a circular path while the filter initialized using the image-based method is still indicating linear motion. Fig. 4.33 visualizes the estimated motion state at frame 151, i.e. shortly after the object has been detected by the image-based method.

Besides starting the tracking earlier, initializing the object's motion state with the radar velocity also improves the result as can be seen in Fig. 4.32(b). Using the image-based initialization method, the initial velocity is underestimated. It takes about 5 cycles until the real object velocity is estimated accurately by the filter. A good estimate of the object's velocity also provides a strong cue for separating points on the object's surface from static scene points.

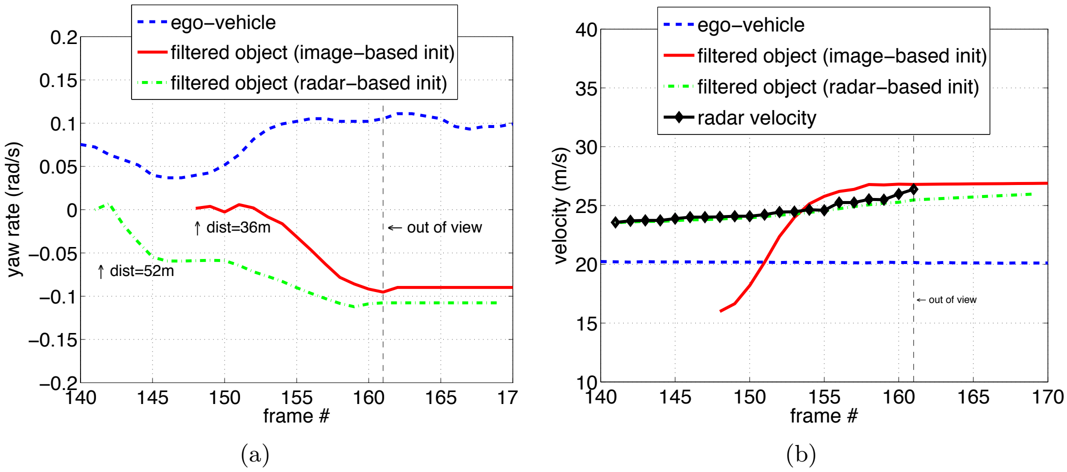
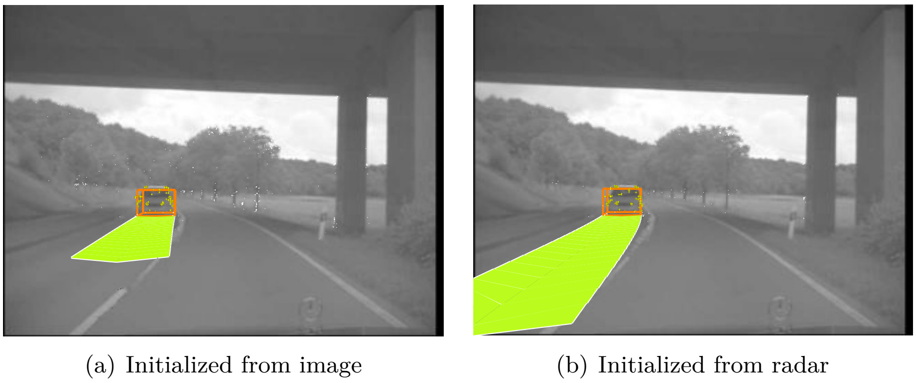


Figure 4.32.: Effect of different initialization methods on (a) yaw rate and (b) velocity estimate for country road curve scenario, with $v_{\text{rel}} \approx 44$ m/s, is shown. Initializing the object's motion state with the radar velocity significantly improves the velocity estimate.



(a) Initialized from image

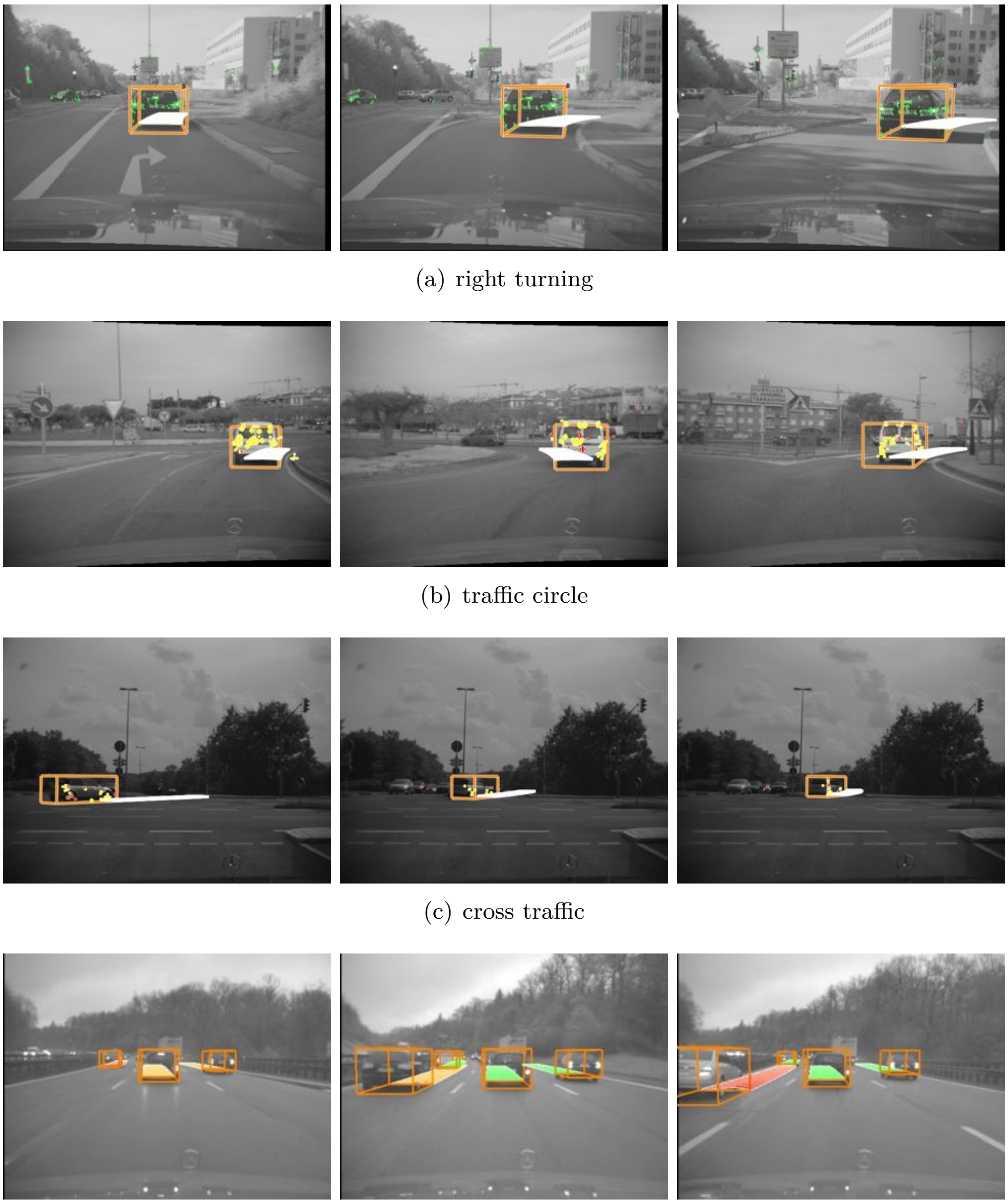
(b) Initialized from radar

Figure 4.33.: Frame 151 of the country road scene with the predicted driving path superimposed is shown. Due to earlier object detection, the yaw rate estimate of the filter initialized by radar already indicates a curved path as the image-only filter still assumes linear motion. The velocity in (a) is also underestimated.

4.4.5. Leading Vehicles and Cross Traffic

Tracking leading vehicles is not the focus of this thesis, since a lot of work has been done in this field before. However, the proposed method can be used to track vehicles driving in any arbitrary direction. In many cases tracking leading vehicles is even less difficult compared to oncoming traffic due to lower relative velocities. Thus, there is more time for filtering. Furthermore, leading vehicles are usually detected at closer distances, i.e., the 3D position of points can be extracted more precise due to lower stereo uncertainty, which directly improves the initial object model.

At this point, only a few examples are presented in Fig. 4.34 to give an idea how the system performs in tracking the vehicles that are driving in the same direction as the ego-vehicle or entering the scene from the side. The scenarios include a right turning car, a leading car in a traffic circle, cross traffic from the left, and a highway scene with four vehicles tracked at the same time.



(a) right turning

(b) traffic circle

(c) cross traffic

(d) high way

Figure 4.34.: The proposed method is not limited to oncoming traffic and single object tracking. The examples show different tracking results of leading vehicles and cross traffic. The curvature of the predicted driving paths, based only on the movement of the 3D point cloud, fits the lane markings or road course very well.

4.4.6. Challenges and Limits

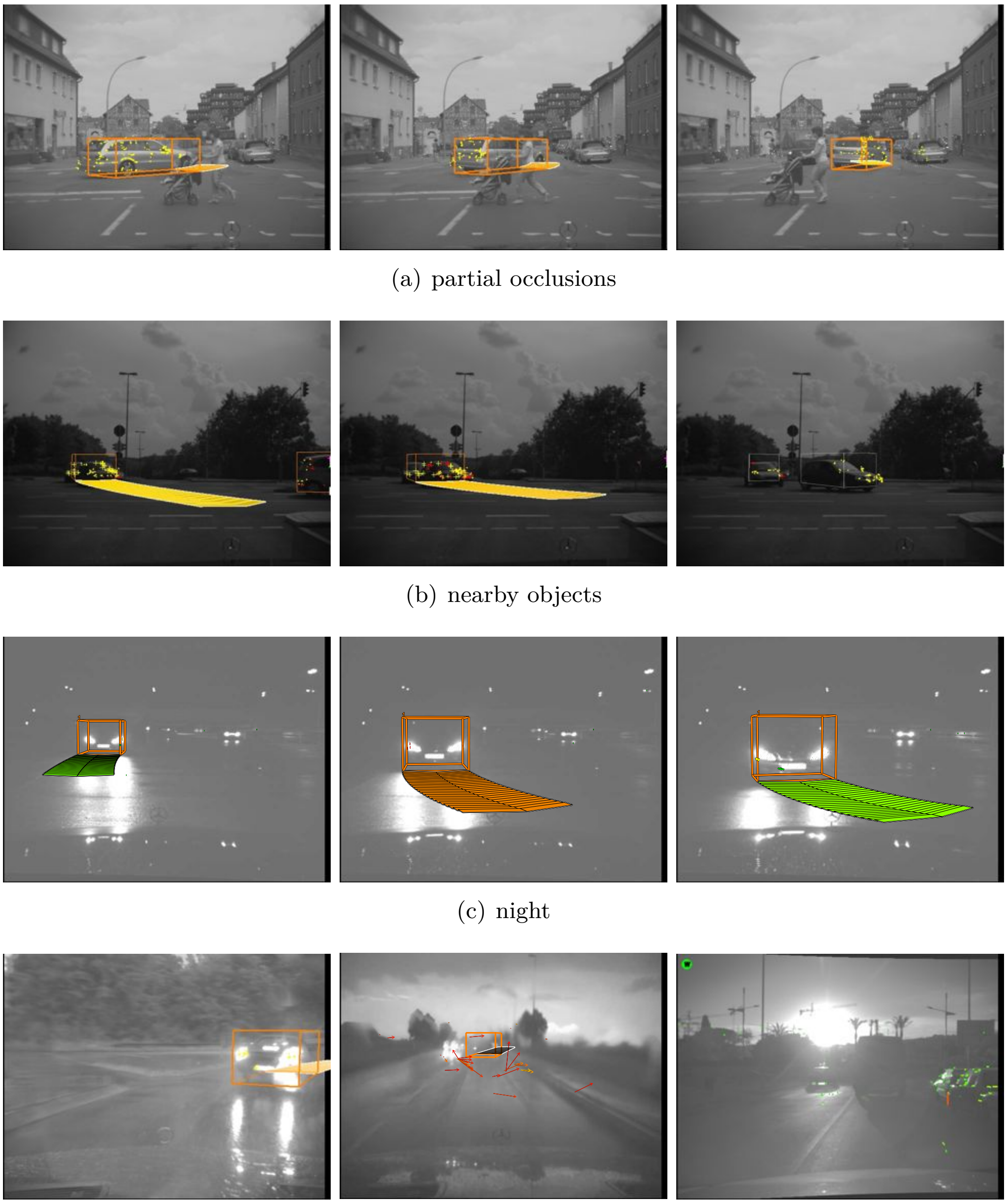
In the following some classes of challenging scenes that generate potential problems are presented and discussed. At some challenging situations, the current system performs already very well, while other scenarios and scene conditions depict the current limits of the proposed approach.

Partial Occlusions: Modeling an object as 3D point cloud is beneficial if partly occlusions occur as can be seen in Fig. 4.35(a). In this example the pedestrian with the stroller is crossing the street in front of the ego-vehicle, while a vehicle turns into the street behind the pedestrian from the left (this scene has already been addressed in Fig. 3.15). Due to a minimum velocity threshold, an object hypothesis is only created for object (2).

The object has been detected before the partial occlusion. If sufficient traceable points on the object's surface are present, the object motion state can be estimated accurately, independent of partly occlusions. In this example, the incompatible motion and distance of feature points found on the pedestrian prevents wrongly adding these points to the object model. Model knowledge of the object's dimension from earlier time steps without occlusion helps in finding new points on the car's front even though they are separated from the majority of points at the car's rear, and prevents the object's model from being split into two parts. Without this prior knowledge or the pedestrian moving in the other direction, it would be extremely difficult to yield the same accurate tracking results in this scene.

Nearby Moving Objects: If two nearby cars are moving in the same direction with approximately the same velocity, these cars are likely to become merged to one single object, as can be seen in Fig. 4.35(b). In this example, the point clouds as well as the stixel silhouette of both cars are merged. Without any further model knowledge on the object dimension or appearance, this problem cannot be solved with the present approach. However, as long as the objects are close and move in the same direction, the merged large object is one potentially dangerous obstacle that should not collide with the ego-vehicle anyway and could be integrated into a situation analysis accordingly. Tracking such object becomes critical at highly dynamic turn maneuvers, since the rotational center of the two cars may lie to some amount away from our geometric expectation. Any larger deviations from the expectation, however, lead to a invalidation at the verification step and the object tracking is reinitialized.

Night Scenes: At night scenes, typically the contrast is very poor and only illuminated parts of the scene become visible. On the other hand, glaring and specular reflections occur that violate several assumptions at stereo disparity computation and feature tracking. Fig. 4.35(c) shows an oncoming vehicle detected and tracked in a rainy night. The vehicle moves toward the ego-vehicle and suddenly turns left to drive around the obstacle. Only the headlights and the number plate illuminated by the ego-vehicle are visible. The grouped information of the few tracked feature points, however, is sufficient to predict the driving path of the vehicle very well, providing the capabilities of the



(a) partial occlusions

(b) nearby objects

(c) night

(d) bad weather and illumination conditions

Figure 4.35.: (a) Partly occlusions do not disturb the tracking as long as enough points are contributing to the object model. (b) Nearby objects moving in the same direction cannot be separated by the proposed method without incorporating model knowledge on the expected object dimension. (c) Although only points on the headlights can be tracked, the driving path is estimated very well. (d) The tracking approach reaches its limits if reliable feature tracking is not possible anymore, e.g., due to heavy rain or extreme backlight.

proposed method. This shows the advantage of the constraining motion model as well as the feature based approach which does not require completeness of the point cloud.

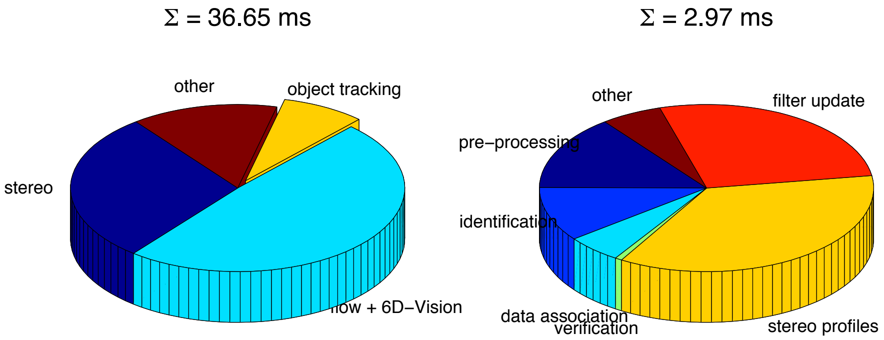
Bad Weather and Illumination Even at daylight conditions outer influences such as heavy rain, fog, or extreme sun light can make a reliable object tracking impossible. In Fig. 4.35(d) at the left, the lead vehicle is still tracked although there are extreme reflections on the object. However, a few frames after this screen-shot, the track is lost since the system fails to find sufficient features to track on the moving object. The image in the middle shows an object that has been detected a few frames earlier in this heavy rain scene. Due to the considerable accumulation of water on the windshield, the images get blurred and the contrast is very poor. In such scenes the image brightness constancy assumption, used for feature tracking, usually does not hold. The obtained few flow vectors (ego-motion compensated), as shown superimposed in the image, are error-prone and do not contain any valuable information for object tracking. At the right hand side, the contrast on the object's surface as well as between the object and the surrounding background is very poor due to the extreme backlight. The system again fails to find sufficient traceable features that would support the object track. In addition, strong reflections with different influence on the left and right image impair the depth computation ability from stereo. However, in most situations that the system fails, humans also have difficulties accurately interpreting the scene.

4.4.7. Computation Time Analysis

The computation time of the real-world system on a Intel Core 2 Extreme CPU (3 GHz) with 4 GB RAM is exemplarily analyzed for a test sequence (scenario 3 of the robot experiments, see Sec. 4.4.2). The object shape model is limited to 100 points. The rotation point measurements are derived from the stereo profiles. The images are acquired and processed at a constant cycle time of 40 ms. The average processing time over 55 frames of the complete system, including stereo and motion computation, is 36.65 ms in this experiment. However, the main interest is on the ratio of the different system components.

As can be seen in Fig. 4.36(a), the motion component is currently the bottleneck of the system (49 %). Here, 2000 feature points are tracked and filtered using the 6D-Vision method. The stereo computation is mainly done on an FPGA, however, transferring the data requires about 10.6 ms (28.9 %). The third largest part of the processing time (14.1 %), denoted as *other* in this chart, includes several basic functionalities such as camera pitch angle estimation, ego-motion estimation, or reading in radar information. The actual object tracking covers only 8 % of the total processing time (< 3 ms) for a single object.

The object tracking is further partitioned into the different stages of processing in Fig. 4.36(b), including a data preprocessing step (14.5 %), the image-based identification of new objects (10.5 %), the data association (5.4 %), or object verification (< 1 %). Analyzing the stereo profiles, as proposed in Sec. 3.5.4, is the computationally most expensive part of the object tracking approach (35.9 %), as it contains the transformation of many hundreds of points from the image domain into vehicle coordinates. The actual Kalman filter update step requires 27.1 %, including the state and measurement



(a) total

(b) object tracking

Figure 4.36.: Average processing time over 55 frames for a single oncoming object with 100 points. (a) Total system. (b) Different object tracking steps.

prediction, linearization of the corresponding system and measurement matrix, as well as the robust outlier detection method. The remaining about 5.8 % of the processing time originate from post-processing steps.

All parts beside the preprocessing and the identification component have to be computed for each object individually, i.e., the total processing time for object tracking scales approximately linearly with the number of objects in the scene.

If the stixel silhouettes are used to derive the rotation point measurements, the overall processing time increases to about 80 ms, since the stixel computation and analysis takes currently about 40 ms. This means, only every second frame can be processed. In practice, this is still acceptable if the relative velocity between ego-vehicle and object is not too fast.

4.4.8. Summary of Real World Results

The robot experiments have indicated that the proposed approach is able to accurately estimate the pose and motion state of oncoming vehicles during straight-line motion with constant velocity, turn maneuvers as well as strong deceleration, while the ego-vehicle is turning quickly. Due to the available ground truth data, these findings could be proven in a quantitative sense. The resulting accuracy measures for the real-world scenes are comparable to the simulation results.

Experiments in real traffic have addressed different scenarios, including intersection turn maneuvers, oncoming traffic at country roads, and tracking the lead vehicle in inner cities or at highways. This emphasizes the **generality** of the approach and the huge field of applications. In all scenarios, both the object pose and dimension have been extracted very well as indicated exemplarily by the object bounding boxes. Furthermore, the quality of the estimated motion parameters has been demonstrated based on the predicted driving paths, superimposed in the images, which are computed based on the estimated motion state. All real-world experiments have been achieved with the **real-time variant** of the approach, i.e., the points have been updated outside the filter.

Both the image-based and the radar-based **initialization** method have been successfully applied to detect objects. The radar initialization is beneficial especially at country roads, due to the large relative velocity between object and ego-vehicle. It has significantly improved the estimation results, since objects are detected at larger distance and the initial velocity is much more accurate than the image-based velocity. As a result, the estimated object yaw rate indicates much earlier that the oncoming vehicle is driving on a curved trajectory.

The point cloud representation is **robust** to error-prone point tracks, partial occlusions, or self-occlusions as long as there are enough reliable point tracks supporting the object. The data association mechanism successfully compensates for lost feature tracks at run-time by adding new points to the model. Promising results at challenging scenes have been achieved with this approach, for example, at the night scene, where the grouped information of all available points on the object, in combination with the vehicle motion model, allows for correctly estimating the rotational velocity of the object.

The system reaches its **limits** if reliable feature tracking or stereo computation is not possible anymore due to bad illumination or weather conditions. Furthermore, the current approach cannot separate two nearby objects moving in the same direction with the same velocity.

5. Conclusions and Future Work

In this document, a new approach for real-time vehicle tracking from a moving platform using dense stereo vision has been proposed and discussed. This approach allows for simultaneous estimation of pose, motion, and shape parameters of a given object by means of an extended Kalman filter. The proposed system is able to accurately track both oncoming and leading vehicles as well as cross traffic, which have been shown both on synthetic and real-world data in a quantitative and qualitative sense. The applicability of this method to various driving situations separates the proposed method from many other existing vehicle tracking approaches.

Objects are modeled by a rigid 3D point cloud, representing the object's shape, and a surrounding cuboid approximating the object dimensions. It has been shown that decoupling the object dimension from the point cloud is especially beneficial at intersections, since the observable object point cloud might be incomplete, due to factors such as visibility constraints or partial occlusions; this decoupling further enables opportunities for sensor fusion of extended objects.

The movements of an object have been restricted to circular path motion using a variant of the well-known bicycle motion model, parametrized by velocity, acceleration, and yaw rate. A reliable estimate of the **yaw rate** of other traffic participants, containing valuable information for trajectory prediction, is the main contribution of this thesis document. To the knowledge of the author, there is currently no other system running in real-time in a demonstrator car that yields the same performance.

Real-time computation time is reached by separating the shape model update from estimating the pose and motion parameters. It has been concluded that the resulting loss in accuracy, compared to a combined estimate model, is acceptable and small in practice.

Besides real-time applicability, two main challenges had to be resolved. First, the system must be robust enough to handle noisy and error-prone data. This includes outlier detection as well as an online assignment of new point tracks to an existing object. Secondly, the wide range of possible vehicle dynamics, reaching from straight-line motion with constant velocity to sudden turn maneuvers at intersections, requires special consideration at the filtering approach.

An iterative robust reweighing of point measurements in combination with an adaptive outlier threshold allows for dealing with, theoretically, up to 50% of outliers in the data. This has been proven both on simulation data and on challenging real world scenes, for example, a rainy night scene. However, limits are reached if reliable feature tracking or stereo computation is not possible anymore, i.e., if there are more outliers than feasible feature tracks in the scene.

A probabilistic data association mechanism allows for assigning new tracked feature points to an existing object track at runtime based on depth and motion compatibility measures. New points are registered to the object shape model and compensate for point tracks that have been lost or detected as outlier. This mechanism is essential for tracking turning vehicles, since during turn maneuvers some parts of the object become occluded while others become visible.

Highly dynamic turn maneuvers require the filter to be flexible enough to follow objects changing their current states quickly. However, the filter should only be reactive if required, and restricting otherwise, to yield smooth trajectories and to be less sensitive to noisy data; which is a significant trade-off. Three different approaches have been proposed to improve the tracking of highly dynamic turn maneuvers. Among these approaches, a multi-filter solution based on the IMM-framework has turned out to be the best compromise between computation time and tracking performance. The slightly better performance of a single filter approach with adaptive system noise, which is controlled via a model-free *oracle*, is computationally much more expensive but has the advantage of being able to also detect unmodeled driving situations such as skidding.

Tracking oncoming vehicles means little reaction times between detection and passing, especially at country roads due to the large relative velocities. Accordingly, a good initialization is essential. Two different initialization methods have been proposed. An image-based method based on 6D-Vision as well as a radar-based initialization strategy that increases the distance range of the system and yields much more accurate initial velocity estimates. In country road scenarios, the radar-based initialization method has outperformed the image-based method due to faster object detection.

Dense stereo data is used to stabilize the rotation point, located at the center rear axle of the vehicle in a geometrical sense. The rotation point position is essential to correctly predict the point positions at rotational movements. However, its position can hardly be reconstructed based on the point movements only. Thus, two methods for deriving additional geometric measurements for the rotation point position have been proposed. One is based on stereo profile histograms and the other based on stixel silhouettes. The additional measurements prevent the rotation point from drifting, for example, outside the vehicle's bounding box.

Throughout this thesis, mainly isolated objects have been considered. The current approach cannot distinguish between two nearby objects moving in the direction, if the corresponding point clouds overlap. In this case, it is very likely that the two objects get merged. Separating two overlapping point clouds requires more specific model knowledge on the object dimensions as well as utilization of appearance features such as gray value, edge, or texture information, which have been completely excluded in the current approach.

Outlook

Dense Scene Flow: Recent advances in *dense scene flow* computation enable the reconstruction of dense 3D motion fields from stereo image sequences in real-time (cf. Sec. 2.4.2). With such methods, depth and motion information is available at almost every pixel in the image, providing new opportunities for object detection and scene segmentation.

The point cloud model allows for predicting the scene flow, induced by a tracked object, at a number of tracked feature points. To be able to predict a dense scene flow

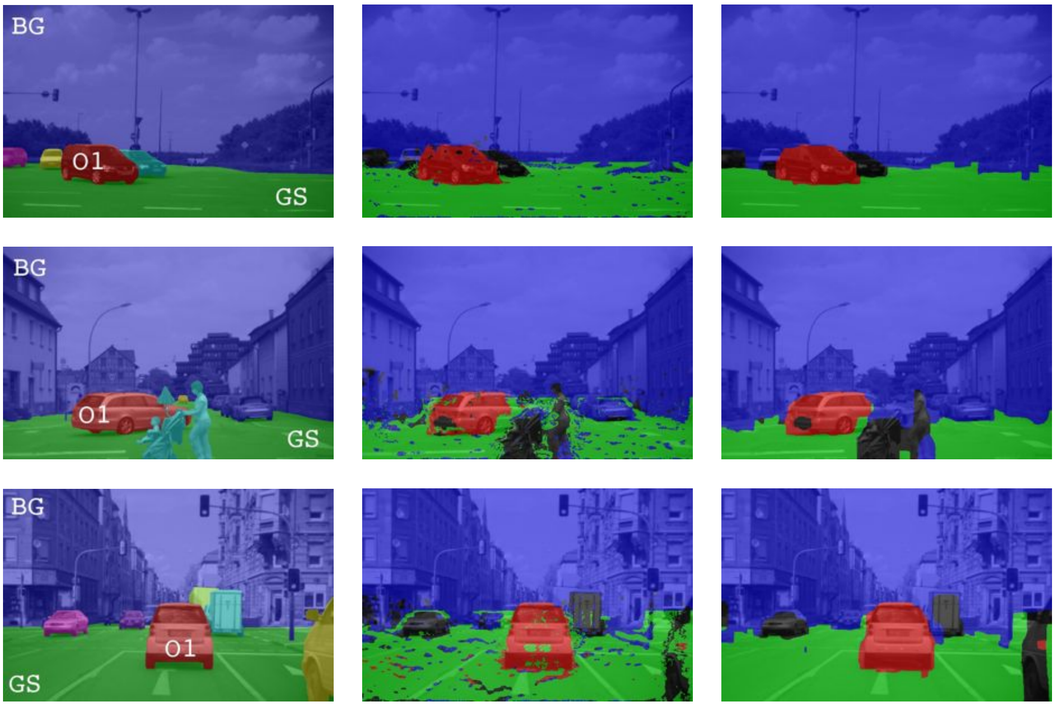


Figure 5.1.: Scene flow segmentation results for different traffic scenes. Knowledge on tracked objects, derived from the presented vehicle tracking approach, is incorporated. Left: Manual ground truth labeling. Middle: Result if the data is evaluated for each pixel independently. The colors encode the maximum class potentials at a given pixel (blue=static background, green=ground surface, red=tracked object, black=unknown moving object). Right: Result if local smoothness and global ordering constraints are incorporated via local neighborhood inference (result after 40 iterations of loopy belief propagation).

for a given object, the object shape model has to be extended to a continuous surface description, e.g., composed of higher order 3D surfaces that represent the actual object shape much more accurately than the flat cuboid sides. Such models are also benefical in predicting partial occlusions.

Scene Labeling: Knowledge on the pose and motion state of tracked objects in the scene can be directly integrated into a globally consistent scene segmentation and *understanding* scheme, that also incorporates other classes besides moving objects such as the ground surface or static obstacles. We have proposed a probabilistic approach for multi-class traffic scene segmentation based on dense scene flow data in [Barth et al. 2010]. The segmentation problem is formulated as a conditional random field (CRF) [Bishop 2006]. The per-pixel potentials are derived purely based on the available depth and motion information from the scene flow, the corresponding uncertainties, as well as prior knowledge on tracked objects in the scene. The basic concepts are similar to the probabilistic data assignment as proposed in Sec. 3.9. The approach differs in a way that an additional class for the ground surface is integrated and that the used scene flow data is available at almost every pixel in the scene. Furthermore, local smoothness as well as global ordering constraints are locally integrated by consideration of neighboring pixels in the CRF. The inference is solved via loopy belief propagation [MacKay 2003]. The first segmentation results are very promising, as can be seen exemplarily in Fig. 5.1. Note that these results do not incorporate any appearance features.

The segmented scene flow data can be used to further improve the object pose and motion estimation. Such feedback loops, e.g., in an expectation maximization sense, will be part of future work.

Long Term Trajectory Prediction: The estimated motion parameters allow for prediction of the driving path for about the next second. For long-term prediction, i.e., $2-3$ s ahead, the current motion state alone is not sufficient enough to reliably predict the object trajectory. For example, an intersection turn maneuver consists of different phases of acceleration, deceleration, yaw acceleration, or constant yaw rate. In many driving situations, the chronology of such phases is similar and becomes predictable, if a subsequence of object states is known from previous time steps.

In [Hermes et al. 2009b] we have presented a particle filter based framework for long-term trajectory prediction. In a training step a variety of intersection trajectories, which have been computed based on the proposed vehicle tracking approach, are processed to setup a reference trajectory database. At run-time, a sequence of previously estimated pose and motion states defines a trajectory segment. Given such segment, we want to predict the object's pose and motion state in a future given time step, based on the similarity of this segment to trajectories in the reference database. The uncertainty of the prediction is represented by the population of particles, where each particle corresponds to a certain trajectory hypothesis. The results have shown that this method is able to improve the prediction accuracy at intersection turn maneuvers.

This work has been extended by Hermes et al. in [Hermes et al. 2009a], which is the basis, e.g., for recognition of different motion classes at intersections [Käfer et al. 2010] and other future work in the field of situation analysis.

Artificial systems aim at *measuring* the (absolute) motion of other traffic participants based on local features; which the human brain measures very poorly. Instead, humans bring observations about object motion into a wider context of the traffic scene. For example, intersections are characterized by a finite number of possible *exit points*, i.e., positions at which a vehicle is able to leave the intersection from (see Figure 5.2). In other words, for a simple intersection scenario it is very likely that a vehicle will cross the intersection straight, turn to one's left or right in a quite deterministic way, or perform a U-turn. The latter may have a lower probability than other options. It is very unlikely that a vehicle is steering into the oncoming traffic

The human brain combines a large number of environmental cues such as street topology (e.g. intersection, T-junction,...), street markings, traffic lanes, traffic signs, or static obstacles (curbs, refuges,...) with knowledge about traffic rules and the observed behavior of other moving traffic participants. A blinking turn signal also provides strong evidence for a turn maneuver. All these cues, containing prior information about a potential maneuver, can be extracted even if the oncoming vehicle is stationary.

Furthermore, drivers usually do not drive at high speed if turning at an urban intersection. There are physical limits restricting the number of possible driving paths a vehicle can take if trying to reach a certain destination. Besides the physical limits there are more or less likely motion patterns, i.e., those allowing for a comfortable and safe reaching of a destination are usually more likely than those exhausting what is physically possible.

Summarized, we have two main types of a priori information humans use for motion prediction: The destination an object is trying to reach (intention) and an expectation about the motion pattern required to reach that destination. Integrating such expectations into the state estimation is a challenging task in future studies.

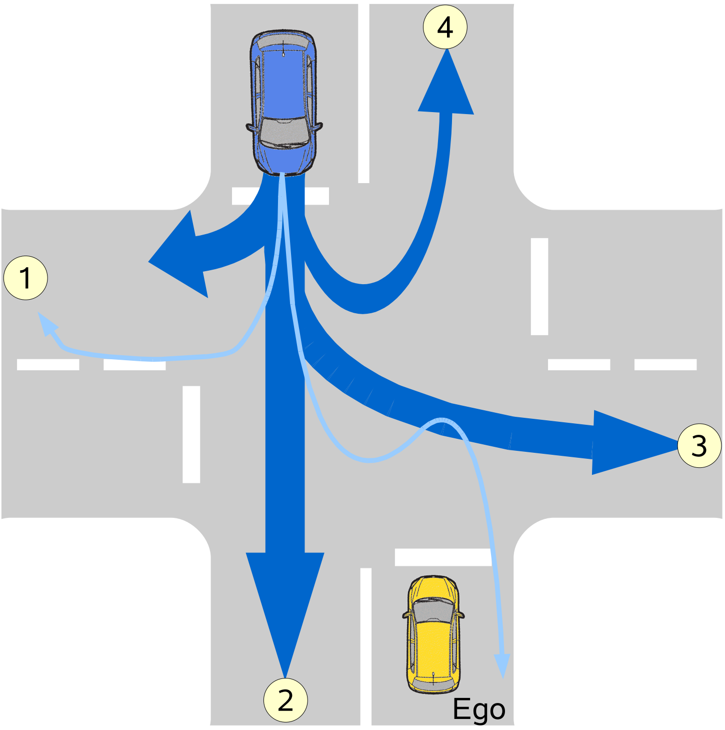


Figure 5.2.: Intersections have a finite number of exit points. In this example there are 4 exit points (yellow circle). The arrows show potential driving paths of the oncoming vehicle, while the thickness of an arrow indicates the fictitious a priori probability of the corresponding driving path. The trajectories indicated by the thin arrows (light blue) are less probable compared to the remaining trajectories.

A. Appendix

A.1. The Unscented Kalman Filter

The unscented Kalman filter, proposed by Julier and Uhlman in [Julier and Uhlmann 1997], contains a different strategy to propagate the mean and covariance matrix of the parameters compared to the EKF (see Sec. 2.6.5).

Prediction: The $2N+1$ sigma points are chosen as follows:

$$\widehat{\boldsymbol{x}}^{(n)}(k-1) = \begin{cases} \widehat{\boldsymbol{x}}^{+}(k-1) + \left(\sqrt{(N+\lambda)C^{+}_{\widehat{x}\widehat{x}}(k-1)}\right)^{\mathsf{T}}_{n} & ,n=1\ldots N \\ \widehat{\boldsymbol{x}}^{+}(k-1) - \left(\sqrt{(N+\lambda)C^{+}_{\widehat{x}\widehat{x}}(k-1)}\right)^{\mathsf{T}}_{n} & ,n=(N+1)\ldots 2N \end{cases} \tag{A.1}$$

and $\widehat{\boldsymbol{x}}^{(0)}(k-1) = \widehat{\boldsymbol{x}}^{+}(k-1)$. The parameter $\lambda = \alpha^2(N+\kappa)$ controls the sampling interval, with α and κ user-defined values depending on the expected distribution, e.g., $\alpha = 1.0$ and $\kappa = 3-N$ for a normal distribution. The *a priori* state estimate is then computed as the weighted sum over all sigma points, propagated through the nonlinear system model f:

$$\widehat{\boldsymbol{x}}^{-}(k) = \sum_{n=0}^{2N} w_n \widehat{\boldsymbol{x}}^{(n)}(k) \tag{A.2}$$

with

$$\widehat{\boldsymbol{x}}^{(n)}(k) = f\left(\widehat{\boldsymbol{x}}^{(n)}(k-1), \boldsymbol{u}(k)\right) \tag{A.3}$$

and w_n a weighting factor defined as

$$w_0 = \frac{\lambda}{N+\lambda} + (1-\alpha^2+\beta) \quad \text{(for covariance computation)} \tag{A.4}$$

$$w_0 = \frac{\lambda}{N+\lambda} \quad \text{(otherwise)} \tag{A.5}$$

$$w_n = \frac{1}{2(N+\lambda)} \quad 1 \leq n \leq 2N. \tag{A.6}$$

The parameter β, with $\beta > 0$, ensures the positive semi-definiteness of the resulting covariance matrices in case of a negative λ. Accordingly, the *a priori* state covariance matrix is predicted from the transformed sigma points:

$$C^{-}_{\widehat{x}\widehat{x}}(k) = \sum_{n=0}^{2N} w_n \left(\widehat{\boldsymbol{x}}^{(n)}(k) - \widehat{\boldsymbol{x}}^{-}(k)\right)\left(\widehat{\boldsymbol{x}}^{(n)}(k) - \widehat{\boldsymbol{x}}^{-}(k)\right)^{\mathsf{T}} + C_{ww}. \tag{A.7}$$

Correction: Before incorporating the measurements, the sigma points are resampled based on the *a priori* state estimate by replacing $\widehat{\boldsymbol{x}}^{+}(k-1)$ by $\widehat{\boldsymbol{x}}^{-}(k)$ and $\mathsf{C}^{+}_{\widehat{x}\widehat{x}}(k-1)$ by $\mathsf{C}^{-}_{\widehat{x}\widehat{x}}(k)$ in (A.1). Based on the new sigma points, $\widehat{\boldsymbol{x}}'^{(n)}(k)$, the nonlinear measurement model is applied to yield a set of samples in measurement space:

$$\widehat{\boldsymbol{z}}^{(n)} = h\left(\widehat{\boldsymbol{x}}'^{(n)}(k)\right) \tag{A.8}$$

The final predicted measurement vector is computed by averaging over all transformed sigma points:

$$\widehat{\boldsymbol{z}}(k) = \sum_{n=0}^{2N} w_n \widehat{\boldsymbol{z}}^{(n)}(k). \tag{A.9}$$

The corresponding covariance matrix $\mathsf{C}_{\widehat{z}\widehat{z}}$ is given by

$$\mathsf{C}_{\widehat{z}\widehat{z}} = \sum_{n=0}^{2N} w_n \left(\widehat{\boldsymbol{z}}^{(n)}(k) - \widehat{\boldsymbol{z}}(k)\right)\left(\widehat{\boldsymbol{z}}^{(n)}(k) - \widehat{\boldsymbol{z}}(k)\right)^{\mathsf{T}} + \mathsf{C}_{zz} \tag{A.10}$$

where C_{zz} is an additive measurement noise term as introduced for the EKF. The Kalman gain is then obtained as

$$\mathsf{K}(k) = \mathsf{C}_{\widehat{x}\widehat{z}} \mathsf{C}^{-1}_{\widehat{z}\widehat{z}} \tag{A.11}$$

with the cross-covariance matrix

$$\mathsf{C}_{\widehat{x}\widehat{z}} = \sum_{n=0}^{2N} w_n \left(\widehat{\boldsymbol{x}}'^{(n)}(k) - \widehat{\boldsymbol{x}}^{-}(k)\right)\left(\widehat{\boldsymbol{z}}^{(n)}(k) - \widehat{\boldsymbol{z}}(k)\right)^{\mathsf{T}}. \tag{A.12}$$

Finally, the *a posteriori* state and covariance are calculated as

$$\widehat{\boldsymbol{x}}^{+}(k) = \widehat{\boldsymbol{x}}^{-}(k) + \mathsf{K}(k)\left(\boldsymbol{z}(k) - \widehat{\boldsymbol{z}}(k)\right) \tag{A.13}$$

$$\mathsf{C}^{+}_{\widehat{x}\widehat{x}}(k) = \mathsf{C}^{-}_{\widehat{x}\widehat{x}}(k) - \mathsf{K}(k)\mathsf{C}_{\widehat{z}\widehat{z}}\mathsf{K}^{\mathsf{T}}(k). \tag{A.14}$$

A.2. Approximation of the Object Translation

As proposed in Sec. 3.4, the in plane translation of the vehicle's reference point is defined as:

$${}^{v}\boldsymbol{T}(\Delta t) = \begin{bmatrix} \int_0^{\Delta t}(v + \dot{v}\tau)\sin(\dot{\psi}\tau)d\tau \\ 0 \\ \int_0^{\Delta t}(v + \dot{v}\tau)\cos(\dot{\psi}\tau)d\tau \end{bmatrix} \tag{A.15}$$

The translation of the X- and Z-coordinate can be described in terms of the nonlinear functions $f_x(\tau)$ and $f_z(\tau)$, respectively, with $f_x(\tau) = (v + \dot{v}\tau)\sin(\dot{\psi}\tau)$ and $f_z(\tau) = (v + \dot{v}\tau)\cos(\dot{\psi}\tau)$.

Assuming a constant velocity, acceleration, and yaw rate for the time interval Δt, the integrals solve to

$$\int_0^{\Delta t} f_x(\tau)d\tau = \frac{\dot{v}\sin(\dot{\psi}\Delta t) - (\dot{v}\dot{\psi}\Delta t + v\dot{\psi})\cos(\dot{\psi}\Delta t)}{\dot{\psi}^2} + \frac{v}{\dot{\psi}} \tag{A.16}$$

$$\int_0^{\Delta t} f_z(\tau)d\tau = \frac{(\dot{v}\dot{\psi}\Delta t + v\dot{\psi})\sin(\dot{\psi}\Delta t) + \dot{v}\cos(\dot{\psi}\Delta t) - \dot{v}}{\dot{\psi}^2} \tag{A.17}$$

For $\lim \dot{\psi} \to 0$, the equations are indetermined. To overcome this problem, the nonlinear functions f_x and f_z are approximated via a Taylor series expansion (Maclaurin series) up to third order terms as follows:

$$\tilde{f}_x(\tau) = f_x(0) + f'_x(0)\Delta t + \frac{1}{2!}f''_x(0)\Delta t^2 + \frac{1}{3!}f'''_x(0)\Delta t^3 \tag{A.18}$$

$$\tilde{f}_z(\tau) = f_z(0) + f'_z(0)\Delta t + \frac{1}{2!}f''_z(0)\Delta t^2 + \frac{1}{3!}f'''_z(0)\Delta t^3 \tag{A.19}$$

where f', f'', and f''' correspond to the first, second, and third derivative of f in this context with

$$f'_x(\tau) = \dot{v}\sin(\dot{\psi}\tau) + (v + \dot{v}\tau)\cos(\dot{\psi}\tau)\dot{\psi} \tag{A.20}$$

$$f''_x(\tau) = 2\dot{v}\cos(\dot{\psi}\tau)\dot{\psi} - (v + \dot{v}\tau)\sin(\dot{\psi}\tau)\dot{\psi}^2 \tag{A.21}$$

$$f'''_x(\tau) = -3\dot{v}\sin(\dot{\psi}\tau)\dot{\psi}^2 - (v + \dot{v}\tau)\cos(\dot{\psi}\tau)\dot{\psi}^3 \tag{A.22}$$

and

$$f'_z(\tau) = \dot{v}\cos(\dot{\psi}\tau) - (v + \dot{v}\tau)\sin(\dot{\psi}\tau)\dot{\psi} \tag{A.23}$$

$$f''_z(\tau) = -2\dot{v}\sin(\dot{\psi}\tau)\dot{\psi} - (v + \dot{v}\tau)\cos(\dot{\psi}\tau)\dot{\psi}^2 \tag{A.24}$$

$$f'''_z(\tau) = -3\dot{v}\cos(\dot{\psi}\tau)\dot{\psi}^2 + (v + \dot{v}\tau)\sin(\dot{\psi}\tau)\dot{\psi}^3. \tag{A.25}$$

This yields

$$\begin{aligned}
\tilde{f}_x(\tau) =& \underbrace{(v + \dot{v}\cdot 0)\sin(\dot{\psi}\cdot 0)}_{0} \\
&+ \left(\underbrace{\dot{v}\sin(\dot{\psi}0)}_{0} + (v + \dot{v}\cdot 0)\cos(\dot{\psi}\cdot 0)\dot{\psi}\right)\tau \\
&+ \left(\frac{1}{2}2\dot{v}\cos(\dot{\psi}\cdot 0)\dot{\psi} - \underbrace{(v + \dot{v}\cdot 0)\sin(\dot{\psi}\cdot 0)\dot{\psi}^2}_{0}\right)\tau^2 \\
&- \frac{1}{6}\left(\underbrace{3\dot{v}\sin(\dot{\psi}\cdot 0)\dot{\psi}^2}_{0} + (v + \dot{v}\cdot 0)\cos(\dot{\psi}\cdot 0)\dot{\psi}^3\right)\tau^3 \qquad \text{(A.26)} \\
=& v\dot{\psi}\tau + \dot{v}\dot{\psi}\tau^2 - \frac{1}{6}v\dot{\psi}^3\tau^3 \qquad \text{(A.27)}
\end{aligned}$$

and

$$\begin{aligned} \tilde{f}_z(\tau) =& (v + \dot{v} \cdot 0)\cos(\dot{\psi} \cdot 0) \\ &+ \left(\dot{v}\cos(\dot{\psi} \cdot 0) - \underbrace{(v + \dot{v} \cdot 0)\sin(\dot{\psi} \cdot 0)\dot{\psi}}_{0} \right) \tau \\ &- \frac{1}{2} \left(\underbrace{2\dot{v}\sin(\dot{\psi} \cdot 0)\dot{\psi}}_{0} + (v + \dot{v} \cdot 0)\cos(\dot{\psi} \cdot 0)\dot{\psi}^2 \right) \tau^2 \end{aligned} \tag{A.28}$$

$$- \frac{1}{6} \left(3\dot{v}\cos(\dot{\psi} \cdot 0)\dot{\psi}^2 - \underbrace{(v + \dot{v} \cdot 0)\sin(\dot{\psi} \cdot 0)\dot{\psi}^3}_{0} \right) \tau^3 \tag{A.29}$$

$$= v + \dot{v}\tau - \frac{1}{2} v\dot{\psi}^2\tau^2 - \frac{1}{2}\dot{v}\dot{\psi}^2\tau^3 \tag{A.30}$$

Now the integral of the approximated functions solve to

$$\int_0^{\Delta t} \tilde{f}_x(\tau) d\tau = \tilde{F}_x(\tau)\Big|_0^{\Delta t} \tag{A.31}$$

$$= \frac{1}{2} v\dot{\psi}\tau^2 + \frac{1}{3}\dot{v}\dot{\psi}\tau^3 - \frac{1}{24} v\dot{\psi}^3\tau^4 \Big|_0^{\Delta t} \tag{A.32}$$

$$= \frac{1}{2} v\dot{\psi}\Delta t^2 + \frac{1}{3}\dot{v}\dot{\psi}\Delta t^3 - \frac{1}{24} v\dot{\psi}^3\Delta t^4 \tag{A.33}$$

for f_x, where $\frac{d}{d\tau}\tilde{F}_x(\tau) = f_x(\tau)$, and for f_z as

$$\int_0^{\Delta t} \tilde{f}_z(\tau) d\tau = \tilde{F}_z(\tau)\Big|_0^{\Delta t} \tag{A.34}$$

$$= v\tau + \frac{1}{2}\dot{v}\tau^2 - \frac{1}{6} v\dot{\psi}^2\tau^3 - \frac{1}{8}\dot{v}\dot{\psi}^2\tau^4 \Big|_0^{\Delta t} \tag{A.35}$$

$$= v\Delta t + \frac{1}{2}\dot{v}\Delta t^2 - \frac{1}{6} v\dot{\psi}^2\Delta t^3 - \frac{1}{8}\dot{v}\dot{\psi}^2\Delta t^4 \tag{A.36}$$

For $\lim \dot{\psi} \to 0$ the translation vector reduces to

$${}^{v}\boldsymbol{T}(\Delta t) = \begin{bmatrix} 0 \\ 0 \\ v\Delta t + \frac{1}{2}\dot{v}\Delta t^2 \end{bmatrix}, \tag{A.37}$$

i.e., the object moves on a straight line in the moving direction which is equal to the Z-direction in the vehicle coordinate system. The amount of motion depends only on the current velocity and acceleration.

Acronyms

2D	two-dimensional
3D	three-dimensional
6D	six-dimensional (3D position + 3D velocity)
ABS	anti-lock braking system
ACC	adaptive cruise control
AVAY	accelerated velocity, accelerated yaw rate motion model
AVCY	accelerated velocity, constant yaw rate motion model
CAN	controller area network
CMOS	complementary metal oxide semiconductor
CPU	central processing unit
CRF	conditional random field
CVCO	constant velocity, constant orientation motion model
CVCY	constant velocity, constant yaw rate motion model
DGPS	differential global positioning system
DOF	degrees of freedom
EKF	extended kalman filter
ESP	electronic stability program
FOE	focus of expansion
FPGA	field-programmable gate array
GPS	global positioning system
GPU	graphics processing unit
IEKF	iterated extended kalman filter
IMM	interacting multiple models
ITS	intelligent transportation systems
KLT	Kanada-Lucas-Tomasi feature tracker
MPEG	moving pictures expert group
MSCE	mean squared corner error
MSE	mean squared error
NEES	normalized estimation error squared
NIS	normalized innovation squared
RANSAC	random sample consensus
RMSCE	root mean squared corner error
RMSE	root mean squared error
ROI	region of interest
SAD	sum of absolute differences
SGM	semi-global matching
SLAM	self localization and mapping
SSD	sum of squared differences
UKF	unscented kalman filter

List of Figures

List of Tables

Bibliography

Alexander, J. C. and J. H. Maddocks [1988]. "On the maneuvering of vehicles". In: *SIAM J. Appl. Math.* 48.1, pp. 38–51. issn: 0036-1399.

Atev, S., H. Arumugam, O. Masoud, R. Janardan, and N. Papanikolopoulos [Dec. 2005]. "A vision-based approach to collision prediction at traffic intersections". In: *Intelligent Transportation Systems, IEEE Transactions on* 6.4, pp. 416–423. issn: 1524-9050.

Bachmann, A. [2009]. "Applying Recursive EM to Scene Segmentation". In: *DAGM Symposium on Pattern Recognition*. Jena, Germany: Springer, pp. 512–521.

Badino, H. [2008]. "Binocular Ego-Motion Estimation for Automotive Applications". PhD thesis. University of Frankfurt.

Badino, H., T. Vaudrey, U. Franke, and R. Mester [2008]. "Stereo-based Free Space Computation in Complex Traffic Scenarios". In: *IEEE Southwest Symposium on Image Analysis and Interpretation*, pp. 189–192.

Badino, H., U. Franke, and D. Pfeiffer [Sept. 2009]. "The Stixel World - A Compact Medium Level Representation of the 3D World". In: *DAGM Symposium on Pattern Recognition*, pp. 51–60.

Badino, H. [Oct. 2004]. "A Robust Approach for Ego-Motion Estimation Using a Mobile Stereo Platform". In: 1^{st} *Intern. Workshop on Complex Motion (IWCM04)*. Guenzburg, Germany, pp. 198–208.

Bar-Shalom, Y., K. Chang, and H. Blom [Mar. 1989]. "Tracking a maneuvering target using input estimation versus the interacting multiple model algorithm". In: *Aerospace and Electronic Systems, IEEE Transactions on* 25.2, pp. 296–300. issn: 0018-9251.

Bar-Shalom, Y., X. Rong Li, and T. Kirubarajan [2001]. *Estimation with Applications To Tracking and Navigation*. John Wiley & Sons, Inc.

Barrois, B., S. Hristova, C. Woehler, F. Kummert, and C. Hermes [2009]. "3D Pose Estimation of Vehicles Using a Stereo Camera". In: *Intelligent Vehicles Symposium, IEEE*, pp. 267–272.

Barth, A. and U. Franke [2009]. "Estimating the Driving State of Oncoming Vehicles From a Moving Platform Using Stereo Vision". In: *Intelligent Transportation Systems, IEEE Transactions on* 10.4, pp. 560–571. issn: 1524-9050.

– [2010]. "Tracking Oncoming and Turning Vehicles at Intersections". In: *Intelligent Transportation Systems, IEEE Conference on*. Madeira Island, Portugal, pp. 861–868.

Barth, A., J. Siegemund, A. Meissner, U. Franke, and W. Förstner [2010]. "Probabilistic Multi-Class Scene Flow Segmentation for Traffic Scenes". In: *DAGM Symposium on Pattern Recognition*. LNCS 6376, pp. 513–522.

Barth, A. and U. Franke [2008]. "Where Will the Oncoming Vehicle be the Next Second?" In: *Intelligent Vehicles Symposium, IEEE*, pp. 1068–1073.

Barth, A., J. Siegemund, U. Franke, and W. Förstner [2009a]. “Simultaneous Estimation of Pose and Motion at Highly Dynamic Turn Maneuvers”. In: *DAGM Symposium on Pattern Recognition*, pp. 262–271.

Barth, A., D. Pfeiffer, and U. Franke [Nov. 2009b]. “Vehicle Tracking at Urban Intersections Using Dense Stereo”. In: *3rd Workshop on Behaviour Monitoring and Interpretation, BMI*. Ghent, Belgium, pp. 47–58.

Bertozzi, M., A. Broggi, A. Fascioli, and S. Nichele [2000]. “Stereo vision-based vehicle detection”. In: *Intelligent Vehicles Symposium, IEEE*, pp. 39–44.

Beymer, D., P. McLauchlan, B. Coifman, and J. Malik [1997]. “A real-time computer vision system for measuring traffic parameters”. In: *Computer Vision and Pattern Recognition, CVPR*. San Juan, Puerto Rico, pp. 495–501.

Bishop, C. M. [2006]. *Pattern Recognition and Machine Learning*. Springer. isbn: 978-0-387-31073-2.

Bouguet, J.-Y. [2007]. *Camera Calibration Toolbox for Matlab*.

Brauckmann, M., C. Goerick, J. Gross, and T. Zielke [Oct. 1994]. “Towards All Around Automatic Visual Obstacle Sensing for Cars”. In: *Intelligent Vehicles Symposium, IEEE*. Paris, France, pp. 79–84.

Brown, M., D. Burschka, and G. Hager [Aug. 2003]. “Advances in computational stereo”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 25.8, pp. 993–1008. issn: 0162-8828.

Brox, T., A. Bruhn, N. Papenberg, and J. Weickert [2004]. “High accuracy optical flow estimation based on a theory for warping”. In: *ECCV*, pp. 25–36.

Buch, N., F. Yin, J. Orwell, D. Makris, and S. A. Velastin [2009]. “Urban Vehicle Tracking using a Combined 3D Model Detector and Classifier”. In: *International Conference on Knowledge-Based and Intelligent Information & Engineering Systems*, pp. 169–176.

Bühren, M. and B. Yang [2006]. “Simulation of Automotive Radar Target Lists using a Novel Approach of Object Representation”. In: *Intelligent Vehicles Symposium, IEEE*, pp. 314 –319.

– [2007a]. “Initialization Procedure for Radar Target Tracking without Object Movement Constraints”. In: *7th International Conference on ITS Telecommunications*, pp. 112–117.

– [2007b]. “On Motion Models for Target Tracking in Automotive Applications”. In: *Workshop on Intelligent Transportation (WIT)*. Hamburg, Germany, pp. 1–6.

Catalin, G. and S. Nedevschi [Aug. 2008]. “Object tracking from stereo sequences using particle filter”. In: *Intelligent Computer Communication and Processing, ICCP. 4th International Conference on*, pp. 279 –282.

Chan, Y., A. Hu, and J. Plant [Mar. 1979]. “A Kalman Filter Based Tracking Scheme with Input Estimation”. In: *Aerospace and Electronic Systems, IEEE Transactions on* 15.2, pp. 237–244. issn: 0018-9251.

Chateau, T. and J. Lapreste [2004]. “Robust real time tracking of a vehicle by image processing”. In: *Intelligent Vehicles Symposium, IEEE*, pp. 315–318.

Chen, S.-C., M.-L. Shyu, and C. Zhang [Aug. 2001]. “An intelligent framework for spatio-temporal vehicle tracking”. In: *Intelligent Transportation Systems. Proceedings IEEE*, pp. 213–218.

Chen, Y., M. Das, and D. Bajpai [2007]. “Vehicle Tracking and Distance Estimation Based on Multiple Image Features”. In: *Computer and Robot Vision, Fourth Canadian Conference on*, pp. 371–378.

Choi, J.-H., K.-H. Lee, K.-C. Cha, J.-S. Kwon, D.-W. Kim, and H.-K. Song [Sept. 2006]. “Vehicle Tracking using Template Matching based on Feature Points”. In: *Information Reuse and Integration, IEEE Int. Conf. on*, pp. 573–577.

Chovan, J. D., L. Tijerina, J. H. Everson, J. A. Pierowicz, and D. L. Hendricks [1994]. *Examination of Intersection, Left Turn Across Path Crashes and Potential IVHS Countermeasures.* Tech. rep. U.S. Department of Transportation.

Cramer, H. [2004]. “Modelle zur multisensoriellen Erfassung des Fahrzeugumfeldes mit Hilfe von SchÃďtzverfahren”. PhD thesis. Technische UniversitÃďt Chemnitz.

Dahlkamp, H., A. Pece, A. Ottlik, and H. Nagel [2004]. “Differential Analysis of Two Model-Based Vehicle Tracking Approaches”. In: *Lecture Notes in Computer Science.* Vol. 3175. Springer, pp. 71–78.

Danescu, R., S. Nedevschi, M. Meinecke, and T. Graf [2007]. “Stereovision Based Vehicle Tracking in Urban Traffic Environments”. In: *Intelligent Transportation Systems, IEEE Conference on*, pp. 400–404.

Danescu, R., F. Oniga, S. Nedevschi, and M.-M. Meinecke [June 2009]. “Tracking multiple objects using particle filters and digital elevation maps”. In: *Intelligent Vehicles Symposium, IEEE*, pp. 88–93.

Dang, T., C. Hoffmann, and C. Stiller [2002]. “Fusing optical flow and stereo disparity for object tracking”. In: *Intelligent Transportation Systems, IEEE Conference on*, pp. 112–117.

Dellaert, F. and C. Thorpe [1997]. “Robust Car Tracking using Kalman filtering and Bayesian templates”. In: *Conference on Intelligent Transportation Systems*, pp. 72–83.

Dietsche, K.-H. and T. Jäger [2003]. *Kraftfahrtechnisches Taschenbuch.* Friedr. Vieweg & Sohn Verlag, Wiesbaden.

Ebbecke, M., M. Ali, and A. Dengel [1997]. “Real time object detection, tracking and classification in monocular image sequences of road traffic scenes”. In: *International Conference on Image Processing* 2, pp. 402–405.

Eberly, D. [Jan. 2001]. *Intersection of Convex Objects: The Method of Separating Axes.* Geometric Tools, LLC.

Effertz, J. [2008]. “Sensor Architecture and Data Fusion for Robotic Perception in Urban Environments at the 2007 DARPA Urban Challenge”. In: *RobVis*, pp. 275–290.

Einhaus, J. [2010]. *Tracking- und Prädiktionsverfahren für Fahrzeuge in komplexen Straßenverkehrsszenarien.* Bachelor Thesis. Bielefeld University. Faculty of Technology.

Färber, G. [2004]. *Automobile Zukunft.* Seminarband. Technische Universität München.

Faugeras, O. [1993]. *Three-Dimensional Computer Vision.* MIT Press.

Ferryman, J. M., S. J. Maybank, and A. D. Worrall [2000]. “Visual Surveillance for Moving Vehicles”. In: *International Journal of Computer Vision* 37, pp. 187–197.

Förstner, W. and E. Gülch [1987]. “A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centers of Circular Features.” In: *Proceed-*

ings of the ISPRS Intercommission Workshop on Fast Processing of Photogrammetric Data, pp. 281–305.

Forsyth, D. and J. Ponce [2003]. *Computer Vision: A Modern Approach.* Prentice Hall.

Franke, U. and S. Heinrich [Sept. 2002]. “Fast obstacle detection for urban traffic situations”. In: *Intelligent Transportation Systems, IEEE Transactions on* 3.3, pp. 173–181. issn: 1524-9050.

Franke, U. and A. Joos [2000]. “Real-time stereo vision for urban traffic scene understanding”. In: *Intelligent Vehicles Symposium, IEEE*, pp. 273–278.

Franke, U. and I. Kutzbach [Sept. 1996]. “Fast Stereo based Object Detection for Stop and Go Traffic”. In: *Intelligent Vehicles Symposium, IEEE.* Tokyo, Japan, pp. 339–344.

Franke, U., C. Rabe, H. Badino, and S. Gehrig [2005]. “6D-Vision: Fusion of Stereo and Motion for Robust Environment Perception”. In: *27th DAGM Symposium on Pattern Recognition*, pp. 216–223.

Franke, U., C. Rabe, S. Gehrig, H. Badino, and A. Barth [2008]. “Dynamic stereo vision for intersection assistance”. In: *FISITA World Automotive Congress, VDI-FVT.*

Fu, C., C. Huang, and Y. Chen [2006]. “Vision-Based Preceding Vehicle Detection and Tracking”. In: *Pattern Recognition, International Conference on*, II: 1070–1073.

Gahlot, A., S. Arya, and D. Ghosh [Oct. 2003]. “Object-based affine motion estimation”. In: *TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region.* Vol. 4, 1343–1347 Vol.4.

Gehrig, S., F. Eberli, and T. Meyer [2009]. “A Real-Time Low-Power Stereo Engine Using Semi-Global Matching”. In: *International Conference on Computer Vision Systems, ICVS*, pp. 134–143.

Gehrig, S. and F. Stein [June 2007]. “Collision Avoidance for Vehicle-Following Systems”. In: *Intelligent Transportation Systems, IEEE Transactions on* 8.2, pp. 233–244. issn: 1524-9050.

Gennery, D. B. [197]. “A Stereo Vision System for an Autonomous Vehicle”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 576–582.

Gillespie, T. D. [1992]. *Fundamentals of Vehicle Dynamics.* Society of Automotive Engineers (SAE).

Goldstein, B. E. [2001]. *Sensation and Perception.* 6th ed. Wadsworth Publishing Company.

Hahn, M., C. Wöhler, J. Einhaus, C. Hermes, and F. Kummert [2010]. “Tracking and Motion Prediction of Vehicles in Complex Urban Traffic Scenes”. In: *4. Tagung Sicherheit durch Fahrerassistenz.* München, Germany, pp. 1–8.

Hampel, F. R. [1973]. “Robust Estimation: A Condensed Partial Survey”. In: *Z. Wahrscheinlichkeitstheorie verw. Gebiete* 27, pp. 87–104.

Harris, C. and M. Stephens [1988]. “A combined corner and edge detector.” In: *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151.

Hartley, R. and A. Zisserman [2003]. *Multiple View Geometry in Computer Vision.* 2nd ed. Cambridge University Press.

Haselhoff, A., A. Kummert, and G. Schneider [Sept. 2007]. "Radar-vision fusion for vehicle detection by means of improved haar-like feature and adaboost approach". In: *15th European Signal Processing Conference*, pp. 2070–2074.

Heikkila, J. and O. Silven [1997]. "A Four-step Camera Calibration Procedure with Implicit Image Correction". In: *Computer Vision and Pattern Recognition, CVPR*. Vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, pp. 1106–1112.

Hemayed, E. [July 2003]. "A survey of camera self-calibration". In: *Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance*, pp. 351–357.

Hermes, C., C. Wöhler, K. Schenk, and F. Kummert [June 2009a]. "Long-term Vehicle Motion Prediction". In: *Intelligent Vehicles Symposium, IEEE*, pp. 652–657.

Hermes, C., A. Barth, C. Wöhler, and F. Kummert [2009b]. "Object Motion Analysis and Prediction in Stereo Image Sequences". In: *Proc. Oldenburger 3D-Tage*, pp. 172–182.

Hirschmüller, H. [June 2005]. "Accurate and efficient stereo processing by semi-global matching and mutual information". In: *Computer Vision and Pattern Recognition, CVPR*. Vol. 2, pp. 807–814.

– [Feb. 2008]. "Stereo Processing by Semiglobal Matching and Mutual Information". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 30.2, pp. 328–341. issn: 0162-8828.

Hirschmüller, H. and S. Gehrig [2009]. "Stereo Matching in the Presense of Sub-Pixel Calibration Errors". In: *Computer Vision and Pattern Recognition, CVPR*, pp. 437–444.

Hong, L. and G. Chen [June 2004]. "Segment-based stereo matching using graph cuts". In: *Computer Vision and Pattern Recognition, CVPR*. Vol. 1, pp. 74–81.

Horn, B. K. P., H. Hilden, and S. Negahdaripour [1988]. "Closed-Form Solution of Absolute Orientation using Orthonormal Matrices". In: *Journal of the Optical Society A* 5.7, pp. 1127–1135.

Horn, B. and B. Schunck [Aug. 1981]. "Determining Optical Flow". In: *AI* 17.1-3, pp. 185–203.

Huber, P. J. [1964]. "Robust Estimation of a Location Parameter". In: *The Annals of Mathematical Statistics* 35, pp. 73–101.

Huguet, F. and F. Devernay [Oct. 2007]. "A Variational Method for Scene Flow Estimation from Stereo Sequences". In: *Proc. International Conference on Computer Vision, ICCV*. IEEE. Rio de Janeiro, Brasil, pp. 1–7.

Jähne, B. [1995]. *Digital Image Processing. Concepts, Algorithms, and Scientific Applications*. Springer-Verlag GmbH. isbn: 3540592989.

Jiang, G.-Y., M. Yu, S.-N. Wang, and R.-D. Wang [Aug. 2005]. "New approach to vehicle tracking based on region processing". In: *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*. Vol. 8, pp. 5028–5033.

Jianguang, L., T. Tieniu, H. Weiming, Y Hao, and S. Maybank [2005]. "3-D model-based vehicle tracking". In: *Image Processing, IEEE Transactions on* 14, pp. 1561–1569.

Julier, S. J. and J. K. Uhlmann [1997]. "A New Extension of the Kalman Filter to Nonlinear Systems". In: *Proc. of AeroSense: The 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls*, pp. 182–193.

Kaempchen, N., K. Weiss, M. Schaefer, and K. Dietmayer [2004]. “IMM object tracking for high dynamic driving maneuvers”. In: *Intelligent Vehicles Symposium, IEEE*, pp. 825–830.

Kaempchen, N., B. Schiele, and K. Dietmayer [Dec. 2009]. “Situation Assessment of an Autonomous Emergency Brake for Arbitrary Vehicle-to-Vehicle Collision Scenarios”. In: *Intelligent Transportation Systems, IEEE Transactions on* 10.4, pp. 678–687. issn: 1524-9050.

Kaempchen, N. [2007]. “Feature-Level Fusion of Laser Scanner and Video Data for Advanced Driver Assistance Systems”. PhD thesis. University Ulm.

Käfer, E., C. Hermes, C. Wöhler, H. Ritter, and F. Kummert [2010]. “Recognition of Situation Classes at Road Intersections”. In: *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1–6.

Kalman, R. [1960]. “A New Approach to Linear Filtering and Prediction Problems”. In: *Transaction of the ASME, Journal of Basic Engineering* 82, pp. 35–45.

Kamijo, S., Y. Matsushita, K. Ikeuchi, and M. Sakauchi [Oct. 2000]. “Occlusion robust vehicle tracking for behavior analysis utilizing spatio-temporal Markov random field model”. In: *Intelligent Transportation Systems, IEEE Conference on*, pp. 340–345.

Kammel, S., B. Pitzer, S. Vacek, J. Schroeder, C. Frese, M. Werling, and M. Goebl [2007]. *Team AnnieWAY.* DARPA Urban Challenge Technical Paper.

Kanhere, N. and S. Birchfield [Mar. 2008]. “Real-Time Incremental Segmentation and Tracking of Vehicles at Low Camera Angles Using Stable Features”. In: *Intelligent Transportation Systems, IEEE Transactions on* 9.1, pp. 148–160. issn: 1524-9050.

Karmann, K. and A. Brandt [1990]. *Moving object recognition using an adaptive background memory.* Elsevier, Amsterdam.

Kay, S. M. [1993]. *Fundamentals of Statistical Signal Processing.* Vol. 1. Prentice Hall.

Kim, Z. and J. Malik [2003]. “Fast vehicle detection with probabilistic feature grouping and its application to vehicle tracking”. In: *Proc. International Conference on Computer Vision, ICCV*, pp. 524–531.

Klappstein, J. [2008]. “Optical-Flow Based Detection of Moving Objects in Traffic Scenes”. PhD thesis. University of Heidelberg.

Koller, D., K. Daniilidis, and H. Nagel [June 1993]. “Model-Based Object Tracking in Monocular Image Sequences of Road Traffic Scenes”. In: *International Journal of Computer Vision* 10.3, pp. 257–281.

Koller, D., J. Weber, and J. Malik [1994]. “Robust Multiple Car Tracking with Occlusion Reasoning”. In: *ECCV*, pp. 189–196.

Labayrade, R., D. Aubert, and J.-P. Tarel [June 2002]. “Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation”. In: *Intelligent Vehicles Symposium, IEEE.* Vol. 2, pp. 646–651.

Lane, R. A. and N. A. Thacker [1996]. *Stereo Vision Research: An Algorithm Survey.*

Leibe, B., N. Cornelis, K. Cornelis, and L. Van Gool [2007]. “Dynamic 3D Scene Analysis from a Moving Vehicle”. In: *Computer Vision and Pattern Recognition, CVPR*, pp. 1–8.

Leotta, M. and J. Mundy [Sept. 2007]. “Epipolar Curve Tracking in 3-D”. In: *Image Processing, IEEE International Conference on.* Vol. 6, pp. 325–328.

Li, X., X. Yao, Y. Murphey, R. Karlsen, and G. Gerhart [2004]. "A real-time vehicle detection and tracking system in outdoor traffic scenes". In: *17th International Conference on Pattern Recognition*, II: 761–764.

Liu, F., J. Sparbert, and C. Stiller [June 2008]. "IMMPDA vehicle tracking system using asynchronous sensor fusion of radar and vision". In: *Intelligent Vehicles Symposium, IEEE*, pp. 168 –173.

Liu, H., F. Sun, and K. He [Apr. 2007a]. "Symmetry-Aided Particle Filter for Vehicle Tracking". In: *Robotics and Automation, IEEE International Conference on*, pp. 4633–4638.

Liu, H., F. Sun, L. Yu, and K. He [Oct. 2007b]. "Vehicle tracking using stochastic fusion-based particle filter". In: *Intelligent Robots and Systems, IEEE International Conference on*, pp. 2735–2740.

Liu, M., C. Wu, and Y. Zhang [Dec. 2007c]. "Motion vehicle tracking based on multiresolution optical flow and multi-scale Harris corner detection". In: *Robotics and Biomimetics, IEEE International Conference on*, pp. 2032–2036.

Loose, H. and U. Franke [2010]. "B-Spline-Based Road Model for 3D Lane Recognition". In: *Intelligent Transportation Systems, IEEE Conference on.*

Lucas, B. D. and T. Kanade [1981]. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *International Joint Conference on Artificial Intelligence*, pp. 674–679.

M. A. Fischler, R. C. B. [1981]. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Comm. of the ACM* 24, pp. 381–395.

MacKay, D. J. C. [2003]. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press.

Maehlisch, M., R. Schweiger, W. Ritter, and K. Dietmayer [July 2006]. "Multisensor Vehicle Tracking with the Probability Hypothesis Density Filter". In: *Information Fusion, 2006 9th International Conference on*, pp. 1–8.

Maehlisch, M., W. Ritter, and K. Dietmayer [June 2007]. "De-cluttering with Integrated Probabilistic Data Association for Multisensor Multitarget ACC Vehicle Tracking". In: *Intelligent Vehicles Symposium, IEEE*, pp. 178–183.

Mahalanobis, P. C. [1936]. "On the generalised distance in statistics". In: *Proceedings of the National Institute of Sciences of India.* Vol. 2. 1, pp. 49–55.

Mark, W. van der and D. M. Gavrila [2006]. "Real-time dense stereo for intelligent vehicles". In: *Intelligent Transportation Systems, IEEE Transactions on* 7.1, pp. 38–50.

Marr, D. [1982]. *Vision.* New York: Freeman.

Martin, L. A. [1907]. *Text-book of mechanics.* J.Wiley & Sons, Inc.

Maybank, S. J., A. D. Worrall, and G. D. Sullivan [1996]. "A Filter for Visual Tracking Based on a Stochastic Model for Driver Behaviour". In: *Proc. 4th European Conf. Computer Vision.* Springer-Verlag, pp. 540–549.

Maybeck, P. S. [1979]. *Stochastic models, estimation, and control.* Vol. 141. Mathematics in Science and Engineering. Academic Press.

McGlone, C., ed. [2004]. *Manual of Photogrammetry.* 5th ed. Amer. Soc. Photogrammetry.

Mehrotra, K. and P. Mahapatra [Oct. 1997]. “A Jerk Model for Tracking Highly Maneuvering Targets”. In: *Aerospace and Electronic Systems, IEEE Transactions on* 33.4, pp. 1094–1105.

Mémin, E. and P Pérez [1998]. “Dense estimation and object-based segmentation of the optical flow with robust techniques”. In: *Image Processing, IEEE Transactions on* 7, pp. 703–719.

Min, D. and K. Sohn [2006]. “Edge-preserving simultaneous joint motion-disparity estimation”. In: *Computer Vision and Pattern Recognition, CVPR*.

Mitschke, M. [1990]. *Dynamik der Kraftfahrzeuge - Band C: Fahrverhalten*. 2nd ed. Springer.

Möbus, R. and U. Kolbe [2004]. “Multi-target multi-object tracking, sensor fusion of radar and infrared”. In: *Intelligent Vehicles Symposium, IEEE*, pp. 732–737.

Morales, S. and R. Klette [2009]. *Prediction Error Evaluation of Various Stereo Matching Algorithms on Long Stereo Sequences*. Tech. rep. University of Auckland, MItech-TR-38.

Morris, B. and M. Trivedi [Sept. 2008]. “Learning, Modeling, and Classification of Vehicle Track Patterns from Live Video”. In: *Intelligent Transportation Systems, IEEE Transactions on* 9.3, pp. 425–437. issn: 1524-9050.

Mosabbeb, E., M. Sadeghi, M. Fathy, and M. Bahekmat [Dec. 2007]. “A low-cost strong shadow-based segmentation approach for vehicle tracking in congested traffic scenes”. In: *Machine Vision, International Conference on*, pp. 147–152.

Narayana, M. and D. Haverkamp [2007]. “A Bayesian algorithm for tracking multiple moving objects in outdoor surveillance video”. In: *The 4th IEEE CVPR Workshop OTCBVS*, pp. 1–8.

Ohta, Y. and T. Kanade [1985]. “Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 7, pp. 139–154.

Ottlik, A. and H. H. Nagel [2008]. “Initialization of Model-Based Vehicle Tracking in Video Sequences of Inner-City Intersections”. In: *International Journal of Computer Vision* 80.2, pp. 211–225. issn: 0920-5691.

Patras, I., E. Hendricks, and G. Tziritas [1997]. “A joint motion/disparity estimation method for the construction of stereo interpolated images in stereoscopic image sequences”. In: *Annual Conference of the Advanced School for Computing and Imaging, Heijen, Netherlands*, pp. 1–9.

Petrovskaya, A. and S. Thrun [2009]. “Model Based Vehicle Detection and Tracking for Autonomous Urban Driving”. In: *Autonomous Robots Journal* 26, pp. 123–129.

Pfeiffer, D., A. Barth, and U. Franke [2009]. “Robust and Precise 3D-Modeling of Traffic Scenes based on Dense Stereo Vision”. In: *6th Workshop Fahrerassistenzsysteme*. Löwenstein/ Hößlinsülz, Germany, pp. 11–20.

Polychronopoulos, A., M. Tsogas, A. Amditis, and L. Andreone [Sept. 2007]. “Sensor Fusion for Predicting Vehicles’ Path for Collision Avoidance Systems”. In: *Intelligent Transportation Systems, IEEE Transactions on* 8.3, pp. 549–562. issn: 1524-9050.

Rabe, C., U. Franke, and S. Gehrig [2007]. “Fast detection of moving objects in complex scenarios”. In: *Intelligent Vehicles Symposium, IEEE*, pp. 398–403. issn: 1931-0587.

Rabe, C., T. Müller, and U. Franke [2010]. "Dense, robust, and accurate 3D motion field estimation from stereo image sequences in real-time". In: *ECCV*.

Rabiner, L. R. [1989]. "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE*. Vol. 77. 2, pp. 257–286.

Rajagopalan, A. and R. Chellappa [2000]. "Vehicle Detection and Tracking in Video". In: *Image Processing, International Conference on*, Vol I: 351–354.

Rannacher, J. [2009]. *Realtime 3D motion estimation on graphics hardware.* Bachelor thesis, Heidelberg University.

Richter, E., R. Schubert, and G. Wanielik [2008]. "Advanced Filtering Techniques for Multisensor Vehicle Tracking". In: *World Congress on Intelligent Transportation Systems*, pp. 1–12.

Rousseeuw, P. J. [1987]. *Robust Regression and Outlier Detection.* John Wiley & Sons, Inc.

Saneyoshi, K. [Oct. 1994]. "3-D image recognition system by means of stereoscopy combined with ordinary image processing". In: *Intelligent Vehicles Symposium, IEEE*, pp. 13–18.

Scharstein, D. and R. Szeliski [2002]. "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms". In: *International Journal of Computer Vision* 47, pp. 7–42.

Scharstein, D. and R. Szeliski [1998]. "Stereo Matching with Nonlinear Diffusion". In: *International Journal of Computer Vision* 28.2, pp. 1573–1405.

Schneider, S., M. Himmelsbach, T. Luettel, and H.-J. Wuensche [June 2010]. "Vision and LIDAR – Synchronization, Correction and Occlusion Reasoning". In: *Intelligent Vehicles Symposium, IEEE.* San Diego, California.

Schubert, R., E. Richter, and G. Wanielik [2008]. "Comparison and evaluation of advanced motion models for vehicle tracking". In: *Information Fusion, International Conference on*, pp. 1–6.

She, K., G. Bebis, H. Gu, and R. Miller [Oct. 2004]. "Vehicle tracking using on-line fusion of color and shape features". In: *Intelligent Transportation Systems, IEEE Conference on*, pp. 731–736.

Shi, J. and C. Tomasi [June 1994]. "Good Features to Track". In: *Computer Vision and Pattern Recognition, CVPR.* Seattle, pp. 593–600.

Siegemund, J. [2008]. "Trajektorienrekonstruktion von bewegten Objekten aus Stereobildfolgen". Diploma thesis. University of Bonn.

Siegemund, J., D. Pfeiffer, U. Franke, and W. Förstner [June 2010]. "Curb Reconstruction using Conditional Random Fields". In: *Intelligent Vehicles Symposium, IEEE*, pp. 203–210.

Simon, D. [2006]. *Optimal State Estimation.* John Wiley & Sons, Inc.

Singer, R. A. [1970]. "Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets". In: *Aerospace and Electronic Systems, IEEE Transactions on* 6, pp. 473–483.

Smith, S. and J. Brady [1995]. "ASSET-2: Real-time motion segmentation and shape tracking". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 17.8, pp. 814–820.

SOATTO, S., R. FREZZA, and P. PERONA [1994]. “Motion estimation via dynamic vision”. In: *Automatic Control, IEEE Transactions on* 41.3. Registered as Technical Report CIT-CDS-94-004, California Institute of Technology, pp. 393–413.

STATISTISCHES BUNDESAMT WIESBADEN [July 2009]. *Unfallgeschehen im Straßenverkehr 2008*. German road traffic accident statistics.

STAUFFER, C. and W. E. L. GRIMSON [1999]. “Adaptive background mixture models for real-time tracking”. In: *Computer Vision and Pattern Recognition, CVPR*. Vol. 2, pp. 246–252.

STEIN, F. [2004]. “Efficient computation of optical flow using the Census Transform”. In: *DAGM Symposium on Pattern Recognition*, pp. 79–86.

SUN, Z., G. BEBIS, and R. MILLER [2004]. “On-road vehicle detection using optical sensors: a review”. In: *Intelligent Transportation Systems, IEEE Conference on*, pp. 585–590.

TEAM MIT [2007]. *Team MIT*. DARPA Urban Challenge Technical Paper.

THRUN, S., W. BURGARD, and D. FOX [2005]. *Probabilistic Robotics*. MIT Press.

TOMASI, C. and T. KANADE [Apr. 1991]. *Detection and Tracking of Point Features*. Tech. rep. CMU-CS-91-132. Carnegie Mellon University.

TOULMINET, G., M. BERTOZZI, S. MOUSSET, A. BENSRHAIR, and A. BROGGI [2006]. “Vehicle Detection by Means of Stereo Vision-Based Obstacles Features Extraction and Monocular Pattern Analysis”. In: *Image Processing, IEEE Transactions on* 15, pp. 2364–2375.

TRUCCO, E. and A. VERRI [1998]. *Introductionary Techniques for 3-D Computer Vision*. Ed. by M. HORTON. Prentice-Hall, Inc.

TSAI, R. Y. [1986]. “An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision”. In: *Computer Vision and Pattern Recognition, CVPR*, pp. 364–374.

– [1987]. “Metrology Using Off-the-Shelf TV Cameras and Lenses”. In: *IEEE Journal of Robotics and Automation* 3.4, pp. 323–344.

VAUDREY, T., C. RABE, R. KLETTE, and J. MILBURN [2008]. “Differences Between Stereo and Motion Behaviour on Synthetic and Real-World Stereo Sequences”. In: *IEEE 23th Int. Conf. on Image and Vision Computing New Zealand (IVCNZ)*, pp. 1–6.

VEDULA, S., S. BAKER, P. RANDER, R. COLLINS, and T. KANADE [2005]. “Three-Dimensional Scene Flow”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 27.3, pp. 475–480. ISSN: 0162-8828.

VEERARAGHAVAN, H. and N. PAPANIKOLOPOULOS [2004]. “Combining multiple tracking modalities for vehicle tracking at traffic intersections”. In: *IEEE Conf. on Robotics and Automation*.

WEDEL, A., U. FRANKE, H. BADINO, and D. CREMERS [2008a]. “B-Spline Modeling of Road Surfaces for Freespace Estimation”. In: *Intelligent Vehicles Symposium, IEEE*, pp. 828–833.

WEDEL, A., T. POCK, and D. CREMERS [2009a]. “Structure- and motion-adaptive regularization for high accuracy optical flow”. In: *Proc. International Conference on Computer Vision, ICCV*, pp. 1663–1668.

WEDEL, A. [2009]. “3D Motion Analysis via Energy Minimization”. PhD thesis. University of Bonn.

Wedel, A., C. Rabe, T. Vaudrey, T. Brox, U. Franke, and D. Cremers [Oct. 2008b]. “Efficient Dense Scene Flow from Sparse or Dense Stereo Data”. In: *Proceedings of the 10th European Conference on Computer Vision, ECCV.* Marseille, France: Springer, pp. 739–751. isbn: 978-3-540-88681-5.

Wedel, A., A. Meissner, C. Rabe, U. Franke, and D. Cremers [2009b]. “Detection and Segmentation of Independently Moving Objects from Dense Scene Flow”. In: *7th International Conference EMMCVPR.* Bonn, pp. 14–27.

Weiss, K., N. Kaempchen, and A. Kirchner [2004]. “Multiple-model tracking for the detection of lane change maneuvers”. In: *Intelligent Vehicles Symposium, IEEE*, pp. 937–942.

Welch, G. and G. Bishop [2006]. *An Introduction to the Kalman Filter.* Tech. rep. Department of Computer Science University of North Carolina at Chapel Hill.

Wender, S. [2008]. “Multisensorsystem zur erweiterten Fahrzeugumfelderfassung”. PhD thesis. University of Ulm.

Xie, L., G. Zhu, M. Tang, H. Xu, and Z. Zhang [June 2006]. “Vehicles Tracking Based on Corner Feature in Video-based ITS”. In: *ITS Telecommunications Proceedings, 6th International Conference on*, pp. 163–166.

Yu, T., R.-S. Lin, and B. T. Boaz J. Super [2007]. “Efficient Message Representations for Belief Propagation”. In: *Proc. International Conference on Computer Vision, ICCV*, pp. 1–8.

Zach, C., T. Pock, and H. Bischof [2007]. “A duality based approach for realtime TV-L^1 optical flow”. In: *DAGM Symposium on Pattern Recognition*, pp. 214–223.

Zeng, Z. and S. Ma [2002]. “An efficient vision system for multiple car tracking”. In: *Pattern Recognition, IEEE International Conference on.* Vol. 2, pp. 609–612.

Zhang, Z. [2000]. “A Flexible New Technique for Camera Calibration”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.11, pp. 1330–1334.

Zhang, Z. and O. Faugeras [1992]. *3D Dynamic Scene Analysis: A Stereo Based Approach.* Springer.

Zhang, Z., Z. Zhang, P. Robotique, and P. Robotvis [1997]. “Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting”. In: *Image and Vision Computing* 15, pp. 59–76.

Zhao, L. and C. Thorpe [1998]. “Qualitative and Quantitative Car Tracking from a Range Image Sequence”. In: *Computer Vision and Pattern Recognition, CVPR*, pp. 496–501.

Zhi-fang, L. and Y. Zhisheng [July 2007]. “A Real-time Vision-based Vehicle Tracking and Traffic Surveillance”. In: *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, Eighth ACIS International Conference on.* Vol. 1, pp. 174–179.

Zinner, C., M. Humenberger, K. Ambrosch, and W. Kubinger [2008]. “An Optimized Software-Based Implementation of a Census-Based Stereo Matching Algorithm”. In: *Proc. of the 4th International Symposium on Advances in Visual Computing, ISVC*, I: 216–227.

Zomotor, A. [1987]. *Fahrwerktechnik/Fahrverhalten.* 1st ed. Vogel.