Freie Universität Berlin

# Multiple camera road perception and lane level localization in urban areas

**Dissertation**

zur Erlangung des Grades eines
Doktors der Naturwissenschaften

am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

vorgelegt von

Frank Philipp

Berlin 2020

|  |  |
|---|---|
| Erstgutachter: | Prof. Dr. Raúl Rojas<br>Freie Universität Berlin |
| Zweitgutachter: | Prof. Dr. Jonas Sjöberg<br>Chalmers University of Technology |
| Tag der Disputation: | 05.02.2021 |

## Selbstständikeitserklärung

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Dissertation selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.
Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.
Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keiner anderen Universität als Prüfungsleistung eingereicht.

Berlin, 12. February 2020

..............................

Frank Philipp

## Zusammenfassung

Die Lokalisierung des eigenen Fahrzeugs und die Assoziation der Pose in bestehendem Kartenmaterial ist eines der zentralen Aufgaben und Grundpfeiler der automatischen Fahrzeugführung. In städtischen Gebieten kommt es dabei zu besonderen Herausforderungen, die sich aus den unterschiedlichsten Verdeckungen und Reflexionen der Satellitensystemsignale ergeben. Die Störungen führen zum Teil zu stark verfälschten Lösungen für die Berechnung der eigenen Position. Diese Effekte treten abhängig vom Bebauungszustand der Umgebung auf, so dass lokal geltende Korrektursignale das Problem nicht lösen.
In dieser Arbeit wird ein Lokalisierungssystem für urbanes Umfeld vorgestellt und untersucht. Es basiert auf der alleinigen Verwendung heute üblicher Serienkameras. Kombiniert man diese zu einem vereinheitlichten Bild der Umgebung ist es durch Zuhilfenahme der Fahrzeugodomometriedaten möglich, Lokalisierungsaufgaben in urbanem Gebiet mit einer Präzision zu lösen, die eine spurgenaue Zuordnung des Fahrzeugs ermöglicht. Die Kameras werden genutzt, um die für die Positionsschätzung notwendigen Landmarken zu detektieren und klassifizieren. Dieser Prozessschritt wird im Detail in dieser Arbeit dargestellt und ausgewertet. Die Ergebnisse werden im Folgenden zur Positionsbestimmung des Fahrzeugs genutzt. Dazu wurden für diese Arbeit OpenStreetMap-Daten derart angereichert, dass sie zur Lokalisierung über die gewonnenen Landmarken genutzt werden können. Welche Form der Datenerweiterung dabei erfolgen muss, wird in dieser Arbeit vorgestellt. Das Nutzen der Initialposition, Positionsupdates über die Fahrzeugodometrie und schließlich die Nutzung der Landmarken für die spurgenaue Lokalisierung werden ebenfalls detailiert präsentiert und ausgewertet. Kern des Lokalisierungsmoduls ist ein Partikelfilter mit dem Bayessches Tracking umgesetzt wird. Es wird angestrebt, dass die durch die Partikel beschriebene Wahrscheinlichleitsverteilung ein guter Repräsentant für die tatsächliche Position ist. Das Modell der Schätzung in einem normalverteilten Prozess wird zu einem multimodalen Ansatz erweitert. Die Vorteile des Partikelfilters können somit bestmöglich genutzt werden.
Die Charakteristika des Landmarken-Messsystems werden dargestellt und es wird gezeigt, wie Ausreißer identifiziert werden können. Darüberhinaus wird dargestellt, wie die Anordnung mehrerer Kameras dabei helfen kann Verfügbarkeitslücken in der Wahrnehmung zu minimieren. Eine Methode zur Assoziation der aktuellen Messung zum Kartenmaterial über eine Kostenfunktion wird dargestellt. Es wird dargestellt, wie eine typische Resamplingmethode dahingehend erweitert werden kann, dass sie den multimodalen Ansatz bestmöglich stützt und die Verstärkung der Wahrscheinlichkeitsdichte im Zustandsraum optimiert. Es wird gezeigt, dass die typischen Mechanismen zur Bewertung eines Partikelfilter-Zustands in urbanem Umfeld nicht ausreichend sind, warum dies so ist und Lösungsmöglichkeiten vorgeschlagen. Die Performanz des Systems wird durch Testfahrten und deren Auswertung in urbaner Umgebung nachgewiesen. Dies schließt dichte Bebauung, Kreisverkehre und Tunnel ein.

## Abstract

The localization of the vehicle and the association of the estimated pose is one of the essential tasks in automated driving. Within urban environment, this task is a challenging one, due to the disturbances that interfere the satellite navigation system signals like reflections or multipath propagation. The disturbances result into an erroneous estimation of the ego pose. The effect and its impact depend on the city structure around the vehicle and therefore local correction signals are not useful.
In this thesis, a precise localization system is introduced and investigated. A main goal for the developed system is to combine all information the car could provide by its serial hardware and using this information for a stable and precise localization in challenging surroundings. Combining the signals of the cameras to a joined view on the surrounding and using cars odometry information, the localization problem within urban areas is solvable. With the cameras it is possible to detect and measure the relative position of lane segment markings, arrow markings, pedestrian crossings and stop lines. The detection process is presented and evaluated in detail. The landmark information is used with enhanced map data based on Open Street Map (OSM). Thereby, a landmark based estimation can be established. The GNSS information is used for an initial pose guess, the vehicle odometry for position updates and finally the detected landmarks for pose corrections. All information is aggregated within a particle filter for Bayes tracking. It is ensured, that the probability dense function from the particles is a good representation of the actual pose and its probability. The estimation from normal distributed processes is enhanced to a multimodal method. Thereby, the particle filter can demonstrate its benefits.
The characteristic of the landmark measurement system is presented and it is shown, how outliers can be identified. Furthermore, the advantage of using multiple cameras in order to improve system availability is presented. A method to associate current measurements with map data by cost function is shown. Additionally it is shown, how the typical resampling method is enhanced, that it supports the multimodal shape of the probability density in best possible way and to optimize it in the state space. It is presented in detail why typical mechanisms to evaluate the particles sets state are not sufficient for urban environment and how to solve this issue. The potential performance of the system is evaluated in test drives in real urban environment, including dense development, roundabouts and tunnels.

## Acknowledgement

This work would not be possible without help of some very important persons. I would like to thank my family for the immense support they provided me all the time. First of all my parents, who supported me for a long time with money and encouragement.
During this work I became a husband and earned a lot of responsibility. In 2016 my wife gave birth to my son Markus and later in 2018 to my daughter Josephine. These events completely changed my life and it was no longer easy to manage the things between family and this work. I am very proud and grateful for my wife, for keeping my back free and make it possible to concentrate on my studies. Jeanette, you have a great share on this work - I love you.
Next to the shoulders of my family, I could rely on a very strong partner for this dissertation: my employer IAV GmbH in Berlin, who supported my work with money, equipment and test cars. They made this work not only an academic one, but with practical use for real life and its evidence. Furthermore, I would like to thank my colleagues for their support and contribution to this work, especially my former students Sven Schumacher and Sven Adfeldt. Not to forget, the most important supporter at IAV for me, Dr. Hadj Hamma Tadjine, for making it possible to visit conferences and to publish the latest obtained findings in highly regarded publications. Next to this, he was a supervisor who kept me on the right way, if I was about to get lost in this work. Thank you, Hamma!
Finally, I would like to thank my academical supervisor Professor Dr. Raúl Rojas from Freie Universität Berlin and his staff. He provided a great support for this work and always had an open door for me.
According to the terms of use, I had to thank the following institutions and communities. Map data copyrighted OpenStreetMap contributors and available from https://www.openstreetmap.org.
Digital ortho-photos (DOP) from Geoportal Berlin, Senate Department for Urban Development and Housing used for this thesis with reference and in accordance to the terms of use in appendix A.8. When data is edited, a remark is provided beside the presented data.

# Contents

# Acronyms

| | |
|---|---|
| **CAN** | Controller Area Network |
| **DGPS** | Differential GPS |
| **DLR** | Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Center) |
| **DOP** | Digital Orthophotos |
| **EKF** | Extended Kalman Filter |
| **ESC** | Electronic Stability Control |
| **GNSS** | Global Navigation Satellite System |
| **GPS** | Global Positioning System |
| **GPU** | Graphics Processing Unit |
| **HOG** | Histogram of Oriented Gradients |
| **iid** | independent identically distributed |
| **IMU** | Inertial Measurement Unit |
| **IPM** | Inverse Perspective Mapping |
| **LiDAR** | Light Detection And Ranging |
| **MMSE** | minimum mean square error |
| **MSE** | mean squared error |
| **NMEA** | National Marine Electronics Association |
| **OSM** | Open Street Map |
| **pdf** | probability density function |
| **RBF** | Radial Base Function |
| **ROI** | Region Of Interest |
| **SIS** | sequential importance sampling |
| **SVM** | Support Vector Machine |
| **UDP** | User Datagram Protocol |
| **UTM** | Universal Transverse Mercator |
| **XML** | Extensible Markup Language |

# 1. Introduction and Motivation

## 1.1. Motivation

In recent years, many efforts were made to push the trend to highly automated driving from a highway piloting function into urban areas. Along with this, new challenges came into focus. One of these challenges, is the position estimation for vehicles to achieve lane accurate driving in cities. Due to multi-path signal propagation, reflections and even blackouts, a single Global Navigation Satellite System (GNSS) solution will not be sufficient to face the challenges for a high precise localization and driving. Differential GPS (DGPS) platforms combined with an Inertial Measurement Unit (IMU) are a step forward, but currently they are expensive and at longer blackouts the position estimation will fail [32] either. Furthermore, they are not a solution for the multi-path propagation problem, which impact is very fine granular in urban surroundings. For this reasons, further methods were proposed and are currently investigated. One research field is the landmark based visual localization and derivatives of it. With this method, special features - retrieved from the cars surrounding by optical sensors - are used to generate an estimation of where inside the map the car is or was located for a specific time. Such systems must rely on a very precise measurement of these landmarks. In this work, one of the best measurable landmark is used to get a hint were the car is: road markings. This thesis will show that road markings of any type are an excellent option to perform ego-position estimation. One challenge in this work, is to correctly classify road markings, even if their shape is distorted or classification must be performed under challenging environmental conditions. It can be shown, that the use of multi-camera sensor setup can give more reliable results for classification, than single sensors and thus help to interpret the vehicles driving situation. It is shown, that line segments - integrated to a higher interpretation level - lead to a meaning, that helps to relocate inside a street-map. The system should distinguish between

- lane border
- stop line
- pedestrian crossing
- arrow marking type

Further categories are feasible, as long as registered in the street-map.
The detections retrieved from cameras, must be aggregated and evaluated within

a structure, that represents the cars environment at a specific time. A special *marking map* is introduced to provide this functionality. After aggregation and fusion of all gathered information from road surface, it must be interpreted. An interpretation layer is presented in this work, to get reliable information about the currently driven lane and its direction type.
The information and measurements retrieved from urban streets could be immense, versatile and inconsistent. Furthermore, the measurement precision differs proportional to the distance and type of camera it was measured. To model this behavior as process with Bayes properties, a particle filter is presented. It is shown, how to model such filter for a well approximation of the cars position on map. The filter is furthermore supported through a vehicle motion-model based on wheel movement. To achieve lane level localization and allocation, the position estimation must meet sub-meter precision for most of the cases.
For the localization process a handcrafted highly detailed map (so called *HD-map*) is used. It is obvious, that for future applications such maps will be indispensable. HD-maps are a profitable business model for card supplier. Therefore, from the beginning of this work, there were no free access road-models available. This issue will certainly be a matter of time. In this work, it is shown how a map must be adjusted and extended to be associated with the position estimation process, exemplary with an Open Street Map (OSM) including parts of Berlin, Germany.
A further goal is to implement a combined detection and localization system, that is based on mass production sensors and hardware, already integrated in nowadays cars. Neither *exotic* nor *expensive* hardware should be used.

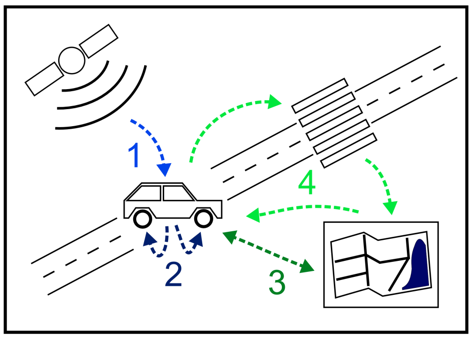


Figure 1.1.: *Diagram of the localization process. The system gets an initial position by GNSS (1). Ego-motion estimation is calculated from wheel movement (2). The system uses a highly detailed map to reference landmarks(3). The cameras detect landmarks like pedestrian crossings. The car relative position of the landmark is calculated and its global position is derived from the street map (4). All information sources are deduced to a final position estimation.*

## 1.2. Structure of this thesis

This work is separated into sequential chapters to give the reader a golden thread, that allows easy comprehension.
Chapter 3 is an introduction to theoretical principles which are helpful to understand this work. The chapter provides the basics for feature calculation of road markings, the different coordinate systems, the ego-motion estimation and finally an introduction to *particle filter* and therefore *Bayes tracking*. An introduction to the used classifiers Support Vector Machine (SVM) and 'random forest' is presented in appendix. In chapter 2 the related works are introduced. It points out, that there is already several academical work focused on road marking extraction and classification. And furthermore the use of extracted landmarks for positioning tasks. The section is followed by chapter 4, describing the methods for image processing to extract the landmark features. In chapter 4.2 the used classifiers and preparation of features for the classification step is presented. Moreover, general principals are introduced to evaluate classifier performances and the descriptiveness of features. Chapter 4.3 shows, how different sensor inputs (e.g. marking contours) can be synchronized and aggregated to a consistent view of the surrounding environment on a moving platform (vehicle). In chapter 4.4, the models are introduced to push environment information to a higher abstraction layer. It presents, how lanes could be modeled especially in urban surroundings with regards to the typical polynomial representation. The lane border model is extended appropriately. Alongside, it is shown how arrow marking information must be processed to get a reliable information about the lanes driving prescription and thus, being a base for lane level localization. Chapter 5 introduces the localization principles this work is based on. It is shown in detail, how the gathered information from the cars surrounding is used for the localization process. The following chapter 5.2, presents the implemented position estimation based on the detected landmarks. In chapter 6, an extensive evaluation of the systems detection and localization performance is given. The single layers *perception* and *localization* are assessed separately. The chapter points out system limitations, also. The last chapter 7 gives a summary about the implemented system at a glance. The results are discussed in detail. The contribution of this work is presented and open topics are shown.

## 1.3. Contributions

This work will show, that improvements for current localization methods are necessary, to make lane level localization possible. All evaluations will be performed on recorded scenes from the area of Berlin (Germany), a city with comparably lot of free space and relatively low buildings. Nevertheless, the effects of multi-path propagation and the permanent leaving and approaching of satellites in sight are remarkable. Furthermore, tunnels and large bridges cause signal blackouts. Issues

that make lane level localization a challenge. Regarding dense populated areas with higher buildings, large bridges and even streets on different levels, it becomes clear, that new localization approaches are necessary.
The following list points out the contribution of this thesis at a glance.

- Lane boundary detection and measurement with machine learning approach, based on lightweight contour classification.
- Two methods for marking arrow classification, including a detailed evaluation of both approaches. Good classification results are shown, even when using small sample sets for training.
- Methods for temporal and spacial fusion of asynchronous lane information from multiple cameras with different lens distortions.
- Therefore, a low cost road perception and interpretation module, usable at systems with small processing capacity.
- Detailed presentation of enhancements to estimate a particle filters effectiveness. And furthermore the detection of failed filter states.
- Presentation of approaches to evaluate and process lane information for particle filter updates. Furthermore, the evaluation of different strategies to trigger update and correction with motion and lane information.
- Introduction of methods for lane segment selection and its assessment at ambiguous lane information.
- A position estimation module with capability for lane level localization.
- A localization method suitable for dense lateral and sparse longitudinal lane information as well.

This thesis shows, that extracted road markings are well suited to reference street maps, and thus to estimate a precise vehicle pose. For streets with multiple lanes, this can be performed at lane level. Two approaches of marking extraction and classification will be introduced and compared. Afterwards, it will be demonstrated how to use the classifier results for localization. Different strategies to manage the set of pose hypotheses within a particle filter are introduced - considering an optimal representation for the actual vehicle pose. As result, a vehicle localization system with lane level precision ($\epsilon < 1m$) is presented. It is completely based on mass production and low cost sensors, already available in nowadays cars.

# 2. Related work

The detection of lane and arrow markings was investigated in several papers. Shigang Li und Y. Shimomura introduced a lane detection for laterally mounted fisheye cameras [46]. In their paper they used the fact, that lane markings on straight lanes have the same direction as the car drives. They use epipole calculation as initial position for the lane vanishing point. With the subsequent edge detection on markings, they presented a method to detect lane borders. A classifier for arrow markings or other line types was not presented.
To increase the detection robustness under challenging weather conditions, in [28], Kum et al. introduced the fisheye lane detection as an improvement for front-camera-systems performance. They used four orthogonal mounted cameras and transformed the images to a bird view perspective. Afterwards, they detect line candidates and tracked them with a linear Kalman filter. They did not present a detection of arrows or other road markings. A system with combined lane detection and arrow marking recognition was introduced by Vacek et al. in 2007 with the paper [51]. They used a typically mounted front camera to detect lanes by a scan line approach and the arrows by a template matching method. Therefore they remapped the arrow templates to ground plane and checked candidates and template for congruence. As validation data set, three scenes including a crossing were used. Thus, the overall number of validation samples was very small (46, with 36 arrows). The capability of their classifier for generalization and which matching method they used was not mentioned.
Another method to detect road markings was described by Danescu and Nedevschi in [34]. They used a scan line based approach combined with grey level segmentation to detect arrow hypotheses. In classification step, a decision tree and geometrical features like size-relations were used. In [30] a geometric approach for arrow marking detection that is based on class prototypes, modeled with arc splines, was presented. Afterwards contour extraction, the arrow candidates were classified by comparison to class-models. Typical classification result tables were not presented. Instead, one contour was evaluated exemplary. Nevertheless, the method reveals the effectiveness of model based geometrical approaches to detect (special) lane markings. N.Wang et al. presented a method to detect arrow markings from ground plane projected images in [52]. They showed, that shape classifying algorithms like Haar wavelets are suitable to detect markings in urban environment. For classification, a multi-class SVM was used. The introduced method showed robustness against occlusion and poor visibility. In [45] a landmark localization - based on lane markings from highly accurate maps - was introduced. The localization accuracy was evaluated on test track and rural roads. In both cases, accuracy

in centimeter-range was achieved by combining map and landmarks.
Most of the studies, focus only one goal out of processing speed, detection accuracy and robustness. Often a separation is made between lane-segments and road objects like arrows. Lane segments are extracted by the typical edge-based methods, but not by a machine learning process. What means, edge groups are classified as lane borders by passing plausibility and logic checks. The method reveals known problems like wrong classified lines, that are not lane borders. The presented algorithms did not learned the 'gestalt' of lane segments. Thus, they are less robust and not adaptable to new types (e.g. border of bike paths). Often, test- and validation sets are very small and no general conclusions about the performance can be made, because the different methods are not compared.
The largest group of input samples, when analyzing urban surroundings, is residual data. In the papers often called 'other objects' or 'negative samples'. The papers often show good recall rates. For real world applications it is not only important to have a high sensitivity on the target class, but a very low sensitivity on the negative class to avoid false positives. Therein, the presented results are often not clear or no information about this issue is provided.
A detailed analysis and comparison of geometrical and image feature based classification methods was presented in [37] and the consecutive publication [38].
The localization problem in this thesis can be abstracted to a sequential state estimation, where the state is the current position, estimated from a sequence of measurements. The measurement process can be modeled with a Bayes filter - investigated and evaluated in many publications. A detailed introduction to *Bayes filtering* and its use for localization problems can be found in [9]. In practice, often *Kalman filter* and *particle filter* are used. A detailed introduction to Kalman and *Extended Kalman filter* can be found in [53]. For the *sequential Monte Carlo sampling* methods, commonly known as particle filter, an introduction can be found in [13]. Furthermore, comparisons of these filters were published in [2] and [50].
There are a few papers that propose the sequential monte carlo sampling for localization, while merging different signal sources. The combination of street-map data, containing visual landmarks, and the Global Positioning System (GPS) signal was proposed by [32]. In their work, they used lane markings and stop lines together with a high precision map from a test track. For data fusion, a particle filter was used. With an initialization period of five minutes (GPS data from the test site) and a thirty minutes long test drive, they get a mean error of $11.6cm$ ($\sigma = 30.4\ cm$) and $2.1\ cm$ ($\sigma = 44.0\ cm$) for the easting and northing values. Within this time, the test track ($l = 2.5\ km$) was driven around multiple times. During another evaluation, they forced a GPS blackout for about thirty minutes and get a mean error of $1.91\ m$. Another work on urban localization problems was published by [21]. They merged GPS, the car odometry and the position of landmarks within a combination of Kalman- and particle filter. As landmarks, they used house corners detected by a laser scanner (Light Detection And Ranging (LiDAR)). The house corners were noticed inside an OSM. At their test run about $l = 860\ m$, they get a mean position error of $4.9\ m$ and a maximum error

of 14.95 $m$.
In [49] GPS, car odometry and lane markings were combined within an extended Kalman filter. For the map-look-up a high precision map was used, containing the marking positions. The map was created handcrafted from a previous drive. They do not used any special high-definition-maps and worked with car series sensors like the Electronic Stability Control (ESC). On their test drive about ten minutes, they get a mean position deviation about 1.21 $m$ ($\sigma = 0.75$ $m$) and a maximum error of 3.04 $m$.
In [39] a method for lane level positioning on downtown roads was proposed. Therefore, a particle filter was extended with a novel step, combining importance weight update and sampling. The method uses a mapping of the detected markings and measurements from multiple-lane scenarios to a high definition street map. Therefore, the probability distribution of the particles was modeled according to the measured distance to the markings with maximum at the closest lane. In the sampling step, the particles were sampled from the posterior belief. Thus, the particles converge to a probable position inside the lanes. It was shown, that this combination of weight update and sampling gives a more stable solution than traditional measurement updates. The evaluation was focused on system availability and correct lane assignment, no statements about a metrical positioning performance was made. For measurement inputs, only signals from nowadays available series sensors were used.
Using a set of multiple cameras to perform lane level localization and the advantages of this method was shown in conference paper [36]. Additionally, it was presented, how to deal with data fusion and the parallel processing of large sensor data.

# 3. Introduction to methodology

## 3.1. Coordinate systems

In this thesis, various coordinate systems are used. They will be introduced in this chapter. For global position references, the Universal Transverse Mercator (UTM) system is used in this work. This is applicable, especially for coordinates in street maps and elements of it. For car-relative positions, the *DIN ISO 8855:2013-10* [12] is used. The standard defines terms regarding vehicle dynamics. The corresponding coordinate system is introduced in 3.1.2.

### 3.1.1. Universal Transverse Mercator

The UTM is a standardized global coordinate system. It is based on the projection of the earths surface on cylindrical and equator-parallel slices. This type of projection is called Transverse Mercator Projection. The system covers the earths surface from 80° south to 84° north, where each hemisphere is divided into 60 zones from east to west, with a zone width of 6°. Every zone has its own specific coordinate system with corresponding projection. The UTM coordinate center is the intersection between equator line and the zones meridian. To avoid negative coordinate values, the zones meridian coincides with 500.000 m east. Coordinates with an easting-value of 400.000 m would be 100 km west from the zones meridian and due to a small projection error nearly 100 km on the earths surface. To distinguish between northern and southern hemisphere, UTM coordinates include a hemisphere designator (N,S). Germany is almost completely covered by UTM zone 32 and 33. Berlin is completely covered by zone 33, Figure 3.1
The spatial division in the UTM system is fine enough to have only small projection errors [43]. The advantage of the system is, that global positions can be described with cartesian coordinates and therefore the euclidean distance and other geometry calculations can be done straight forward with high precision.
Remember, that every zone uses its own projection of the Cartesian grid and therefore every transformation in or from UTM is only valid for a specific zone. This circumstance must be considered when passing from one zone to another. There is no relation between inter-zone-coordinates. At the border between zones, it can be a good solution to determine which one is mainly used and let its grid overlap into the other. Generally, the error would be small. For long distance travels, a smooth handover between zones must be implemented.

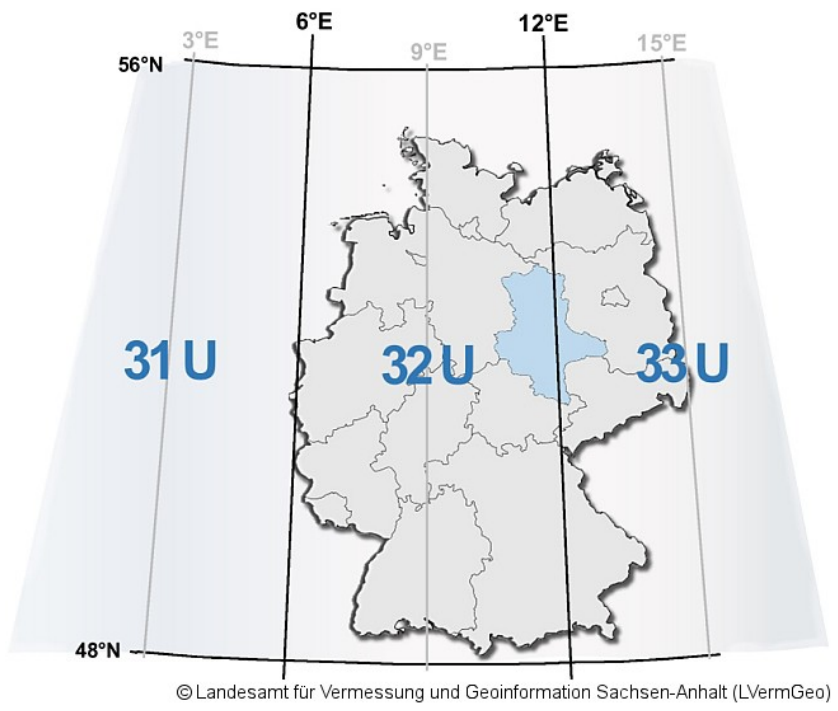

6°E
12°E
3°E
9°E
15°E
56°N
48°N
31 U
32 U
33 U
© Landesamt für Vermessung und Geoinformation Sachsen-Anhalt (LVermGeo)

Figure 3.1.: *Germany is almost completely covered by UTM zones 32 and 33, Berlin completely by zone 33. The U describes the position in the south to north slice, where U is a slice from northern hemisphere.*

### 3.1.2. Vehicle and sensor related coordinate systems

#### Car coordinates

The detected objects from the cars sensors are generally referenced in car relative coordinates. The system is fixed to the car and therefore it moves with the car. Moreover, detected objects can easily be shifted for a period, not only by their own movement, but additionally with motion of the car. The origin of the car coordinate system, is the middle of the rear axle at ground level. For a planar road, $Z = 0$ will correspond to the road surface. The $X$-axis points along the cars longitudinal axis to the front. The $Y$-axis is orthogonal to $X$ and points to the cars left side along the rear axle. Because it is a right-handed coordinate system, the $Z$-axis points upwards from the rear axle center (origin), 3.2. We use the index $_F$ for a point $P_F = (x_F, y_F, z_F)$, given in car relative coordinates.

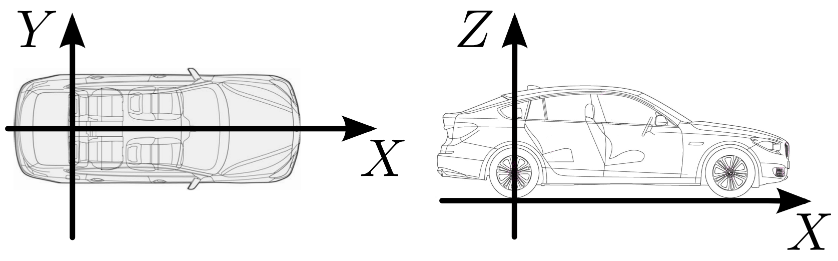


Figure 3.2.: *Views of the cars coordinate system from birdview ($XY$ plane/driving plane) and side view ($XZ$ plane/car profile)*

### Camera coordinates

The camera coordinate system is relevant for projections from the cars surrounding into camera image. It means that object positions are referenced relative to the camera. The systems origin is the camera center. The direction of the axes can depend on the used camera projection model (pinhole camera vs. fisheye lens), but it is - in our case - right-handed. The pinhole model uses an alignment where $Z$ points along the line of sight away from the camera and thus it is a scale for depth. According to this determination and with right-hand-criterion, the $X$-axis points to the right and the $Y$-axis downwards. The used fisheye model differs from that in a way, that it is rotated 180° around the $Y$-axis. $X$ then points to the right and $Z$ backwards from camera view. We use the index $_K$ for a point $P_K = (x_K, y_K, z_K)$, given in camera coordinates.

### Image coordinates

The image coordinate system is a two dimensional Cartesian system. It is used to reference the pixels of an image. For axes naming, $u$ and $v$ has been established. The origin in this system is the upper left image corner. The $u$-axis runs parallel to the image rows from left to right. The $v$-axis runs from top to bottom along parallel to the image columns. All pixels of an image are situated in the first quadrant of the image coordinate system. For an image coordinate $p = (u, v)$ we use lower case letters for distinction.

## 3.1.3. Transformations between coordinate systems

For localization systems, which associate image related objects (pixels) with a position on a map, it is necessary to define transformations between the coordinate systems for both directions. The illustration in Figure 3.3 shows the transformations that must be used. To transform a point $P_F$ from car coordinate system to camera coordinate system, we use equation 3.1. Where $R$ is a $3 \times 3$ rotation matrix and $\vec{t}$ a translation vector, describing the rotation and shift between the coordinate systems. In our case, it describes the mounting position and orientation of the camera to the car and therefore is called extrinsic camera parameters. They must be determined with a camera calibration procedure.

$$P_K = R \cdot P_F + \vec{t} \tag{3.1}$$

The equation for the opposite direction then is

$$P_F = R^{-1} \cdot (P_K - \vec{t}).$$

The transformation between camera and image coordinates is given by the models in 3.1.4. In this transformation, from three-dimensional space to a (image-)plane, the third component (depth or $z$) get lost. For the inverse mapping from image to camera coordinates, that component is missing. Therefore, a projection ray is assigned to every image coordinate. A specific solution can be calculated by intersect this ray with a given reference plane.

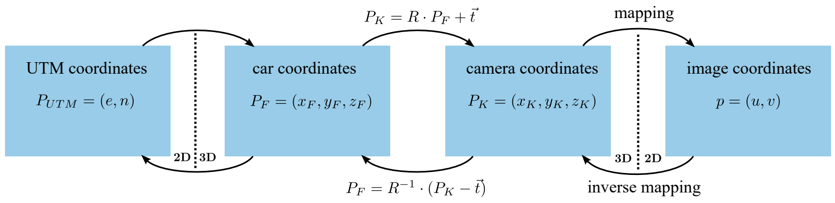


Figure 3.3.: *Transformation between coordinate systems from a global metric position to image pixels and vice versa.*

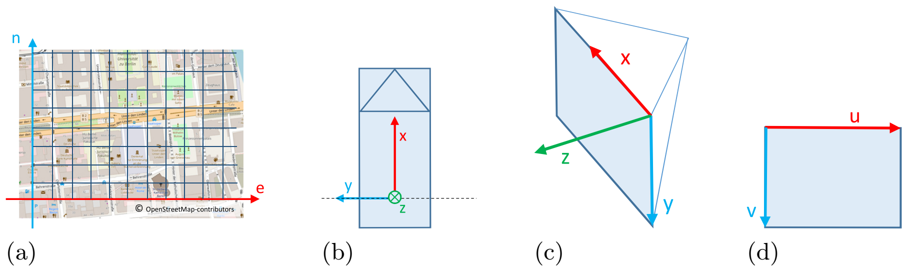


Figure 3.4.: *Illustrations of the relevant coordinate systems. (a) Universal Transversal Mercator. (b) Car coordinate system. (c) Camera coordinate system. (d) Image coordinate system.*

## 3.1.4. Lens models

### Pinhole model

For this thesis, I use the mathematical model from Heikkila and Silven [20]. The idealized geometrical relations of a pinhole model is illustrated in figure 3.5. The $Y$-coordinate of a point mapped to the image plane is given by the *theorem on*

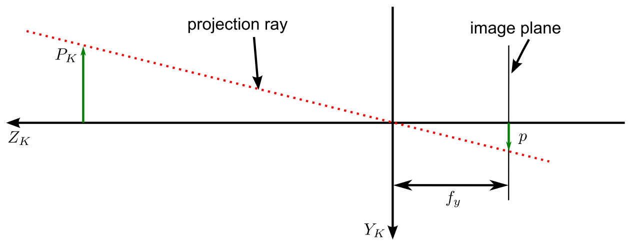


Figure 3.5.: *Geometrical relations in the ZY-plane. Point $P_K$ is projected to the point p on image plane. f is the focal length, depending on the used optics.*

*intersecting lines* and the focal length $f_y$ in $y'_K = f_y \cdot \frac{y_K}{z_K}$. The focal length is the distance between the center of projection and the image plane.
The optical center of the camera lens generally differs slightly from the image center. It is described with the values $c_y$ and $c_x$. Applying this shift and scaling the camera coordinate by pixel height $s_y$ and pixel width $s_x$, the image position is given by

$$\begin{aligned} u &= s_x \cdot (x'_K + c_x), \\ v &= s_y \cdot (y'_K + c_y) \end{aligned} \tag{3.2}$$

Because of the idealization of this model, one important aspect is missing. Real lenses generally come with distortion, especially when using enhanced field of view optics. The most common types of lens distortion are radial distortion $y''_{Krad}$ and tangential distortion $y''_{Ktan}$. They can be mathematically defined separately and their impact to the mapping function 3.2 is additive, $y''_K = y''_{Krad} + y''_{Ktan}$.
The radial part of the lens distortion is modeled by a polynomial function:

$$y''_{Krad} = y'_K \cdot \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \ , \tag{3.3}$$

where $r = x'^2_K + y'^2_K$ is the distance to camera center and $k_1$ to $k_3$ the radial distortion coefficients. For the $X$-component the calculation is in analog manner.
The tangential part of lens distortion for $Y$-component is

$$y''_{Ktan} = p_1 \left(r^2 + 2y'^2\right) + 2p_2 x' y' \tag{3.4}$$

and for the $X$-component

$$x''_{Ktan} = 2p_1 x' y' + p_2 \left(r^2 + 2x'^2\right) \ . \tag{3.5}$$

Where $p_1$ and $p_2$ are the tangential distortion coefficients. The transformation to image coordinates can then be done with substituting $y''_K$, $x''_K$ in equation 3.2.
The inverse transformation from image to camera coordinates is quite special because of the missing depth information and the distortion, that must be reversed. Considering the equations 3.3 to 3.5, it is obvious, that reversing the functions is non trivial. Therefore, in practical use the un-distortion is done by approximation

methods. This can be achieved by calculate mappings between two image planes, one for the distorted and one for the undistorted pixels, with the equations above. The un-distorted plane will have gaps, where no information about the origin from the distorted plane is available. These gaps generally are filled via linear combinations (*bilinear filtering*) from the adjacent pixels, where an information is available. As mentioned earlier in this chapter, the back-projection of an image point to its corresponding camera-coordinate is not unambiguous possible because of the missing depth information. Therefore, to every pixel a projection ray is assigned via the idealized pinhole model and after un-distortion. With a known reference plane, for this ray the intersection can be calculated to retrieve a corresponding world coordinate.

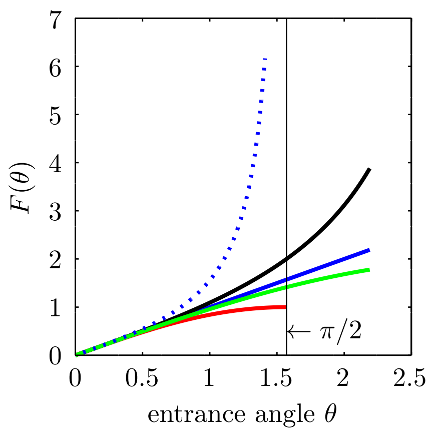


Figure 3.6.: *Illustration of different mappings $F(\theta)$ ($f = 1mm$) and their function at lens aperture angles $\theta$ with $0 \leq \theta \leq 180°$ and above.*

Figure 3.7.: *Mapping functions.*

| projection type | function |
|---|---|
| — equidistant | $F(\theta) = f \cdot \theta$ |
| — conformal | $F(\theta) = 2f \cdot \tan(\theta/2)$ |
| — orthographic | $F(\theta) = f \cdot \sin(\theta)$ |
| — equal-area | $F(\theta) = 2f \cdot \sin(\theta/2)$ |
| ··· pinhole | $F(\theta) = f \cdot \tan(\theta)$ |

## Fisheye lens model

The distance $r$ of an image point to the optical center for a pinhole model can also be described with the angle $\theta$ of the projection ray to the optical axis and the focal length $f$ with $r = f\tan(\theta)$. It is obvious, that for wide angle cameras, large image planes (respectively sensor planes) are needed and there is no mapping for a field of view larger than 180°. Modeling the projection ray through fisheye optics is a widely researched topic. Scaramuzza et al. [44], Kannala and Brandt [25], as well as Basu and Licardie [4] propose a camera model that bases on a polynomial function

$$F(\theta) = k_0 + k_1\theta + k_2\theta^2 + k_3\theta^3 + ... + k_N\theta^N \quad (N\text{...desired grade}). \tag{3.6}$$

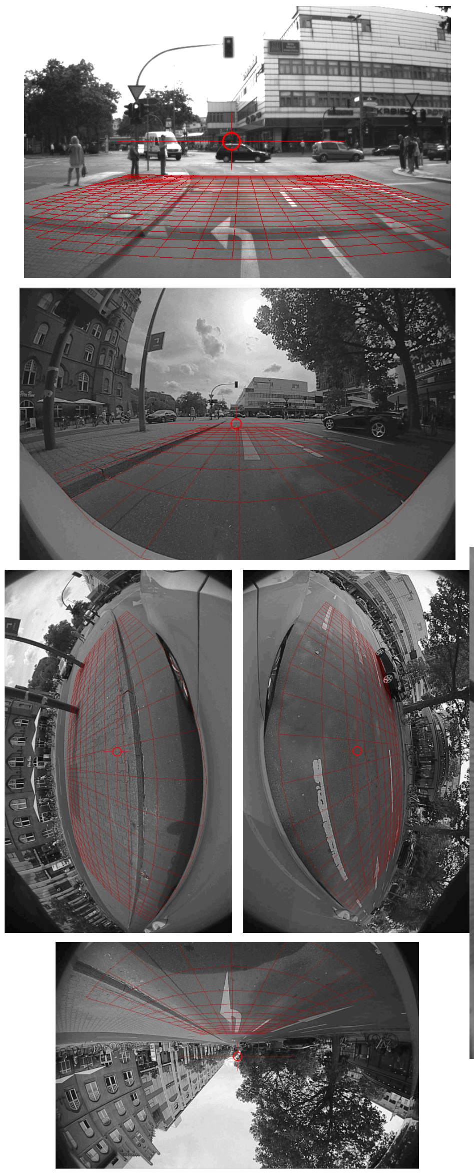
Thus, it is possible to describe all mapping relations as shown in figure 3.6 with only one polynomial function. Moreover, it is possible to use a unified model for all camera optics, independent from the field of view. In this thesis a polynomial model with fourth degree and $k_0 = 0$ is used.

### 3.1.5. Inverse perspective mapping

The view angle from a camera into a scene leads to a perspective mapping of the three dimensional surrounding to the $2D$ image plane. The Inverse Perspective Mapping (IPM) is inverse to this process. It is a special image to image transformation, where the values of image coordinates are remapped into their world representation, generally speaking, to car relative coordinates at height $z = 0$. Thus, it is a $2D \rightarrow 2D$ transformation, the $z$-information from the original scene get lost. Especially, when analyzing the road surface, the IPM could be very useful. The IPM includes a transformation from camera coordinates to sensor pixels, which depends on the chosen camera lens model (pinhole vs. fisheye) and describes the rays through the camera optics.
For the mapping process the following information of a camera must be known:

- cameras intrinsic and extrinsic calibration parameters
- the outer coordinates of the (world)area that should be mapped
- the desired resolution of the IPM-image

Figure 3.8 shows the mapping input and output for an inverse perspective mapping.


(a)

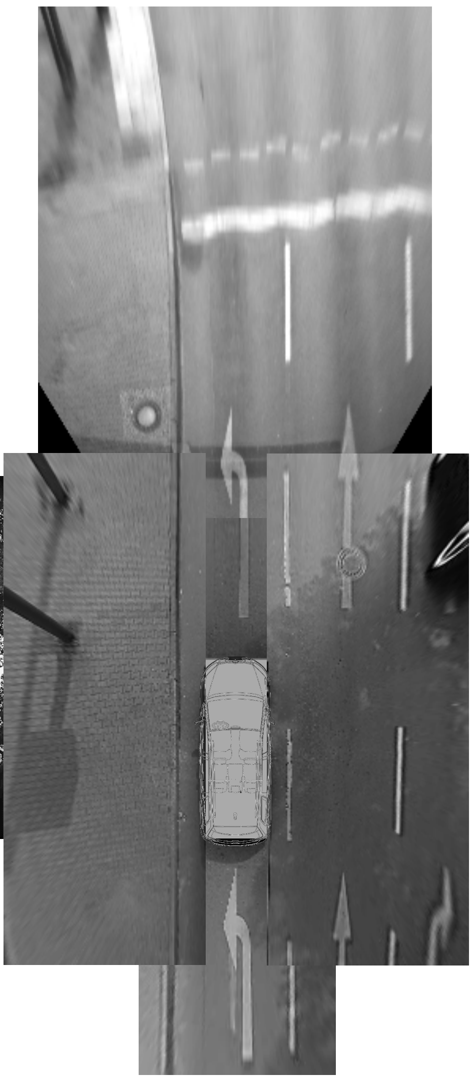

(b)

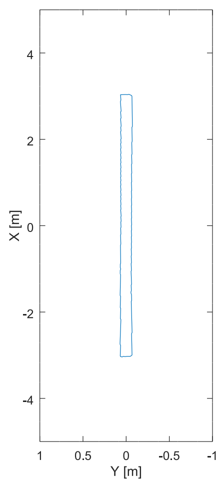
Figure 3.8.: *a) Origin images and b) the result after inverse perspective mapping of the red grid area. The IPM images are aligned to their corresponding outline coordinate.*

## 3.2. Descriptive features of road marking contours

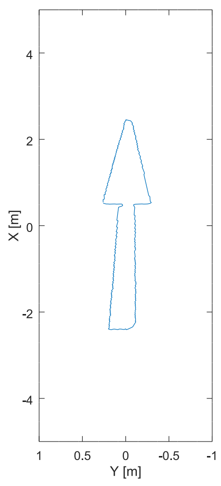
Contours are described as a sorted list of two-dimensional points

$$C = \{P_1, P_2, ..., P_N = (X_{F,N}, Y_{F,N})\}, \tag{3.7}$$

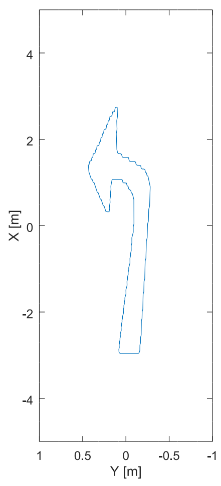
where $N$ is the number of contour points. The third dimension $Z_F = 0$ (height) can be neglected, assuming planar road surface. Figure 3.9 illustrates road markings, extracted by the implemented perception module.



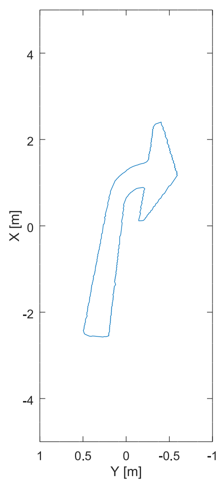
(a) *class 1*



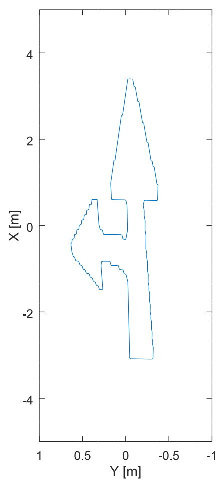
(b) *class 2*



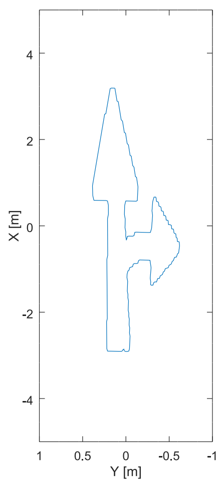
(c) *class 3*



(d) *class 4*



(e) *class 5*



(f) *class 6*

Figure 3.9.: *Illustration of several marking contours, extracted by the perception module.*

For contour classification, a set of features must be defined, that are characteristically for the contour classes. The proposed feature set consists of three groups: moments, geometrical relations and Fourier descriptors.

### 3.2.1. Moments

Moments are a description for orientation and shape of weighted distributions. The extracted contours can be taken as a two-dimensional distribution with value 1. Typically, moments are determined by integration over their distribution. For contours, obviously no distribution is given, but its outline. Following the *Green theorem* [18], the integration over an enclosed area can be transformed to an integration over the areas border. Hence, it is possible to calculate moments for contours.
From moments $m_{pq}$, where $p$ and $q$ describe the order of the moment, *centralized moments* $\mu_{pq}$ can be derived. Therefore, the center of the contour is calculated by $\overline{P} = (\overline{X_F}, \overline{Y_F})$ with

$$\overline{X_F} = \frac{m_{10}}{m_{00}}, \quad \overline{Y_F} = \frac{m_{01}}{m_{00}}.$$

The result is not the geometrical center, but the mass center of the contour. With the given point of the mass center, the centralized moments can be calculated with use of equation 3.8. In the next step, the centralized moments can be *normalized* using $\eta_{pq} = \mu_{pq}/\mu_{00}^{\gamma}$, where $\gamma = 0,5 \cdot (p+q) + 1$. [17]

$$\mu_{pq} = \sum_X \sum_Y (X - \overline{X_F})^p \cdot (Y - \overline{Y_F})^q \tag{3.8}$$

According to Hu [33], from *normalized, centralized moments* another group of moments - the so-called *Hu-moments* - can be determined. The advantage of Hu-moments is their in-variance against rotation, translation, and scaling. The *Hu-moments* $m_{H0}$ to $m_{H6}$ and the *normalized, centralized moments* $\eta_{20}, \eta_{11}, \eta_{02}, \eta_{30}, \eta_{21}, \eta_{12}, \eta_{03}$ forming the first part of the proposed feature set.

### 3.2.2. Additional geometrical features

Geometrical features, that can directly be calculated from contours like area $A$ or length $L$, depend on the contours size and hence are not suitable for a feature selection. A better choice is the use of relations between them. For example *roundness* $R = L^2/A$ is a more shape describing feature. Another one is the *enclosing rectangle to contour area-ratio* $R_{RA} = A_R/A$, where $A_R$ is the area of the minimum enclosing rectangle of the contour. In similar way, an *enclosing circle to contour area-ration* $R_{CA} = A_C/A$ can be determined. Finally, the area of the convex hull $A_H$ can be useful and the corresponding ratio (*convex hull to contour area-ration*) can be calculated with $R_{HA} = A_H/A$.

With centralized moments, as defined in 3.2.1, the axes $a$ and $b$ of an ellipse can be determined. The semi-major axis $a$ and the semi-minor axis $b$ forming the *inertia tensor* of the contour. The ratio $R_{ab} = a/b$ (*major to minor axis-ratio*) describes the contours spatial expansion.

$$a = \sqrt{2\left((\mu_{20} + \mu_{02}) + \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}\right)}$$

and

$$b = \sqrt{2\left((\mu_{20} + \mu_{02}) - \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}\right)}$$

Furthermore, mass center and geometrical center can be used to derive a new feature. Generally, these points do not have a similar position and thus a vector between them can be determined. The angle between this vector and the major axis is an indicator for the position of the contours mass center relatively to the contours orientation. The feature is called the contours *gravity balance* $\beta$.

### 3.2.3. Fourier descriptor

The two dimensional contours (closed sequence of 2D-points) can be described as a set of complex numbers with $z_j = X_{F,j} + iY_{F,j}$. From that sequence it is possible to determine the discrete Fourier transform with the use of equation 3.9. $\tilde{z}(u)$ is called Fourier descriptor of the contour. Before transformation can be performed, it is required to have the contour coordinates in an equidistant form. That can be achieved with sub-sampling, 4.1.1.

$$\tilde{z}(u) = \frac{1}{N}\sum_{n=0}^{N-1} z_n \exp\left(-\frac{2\pi iun}{N}\right) \quad N \text{ ...number of points} \tag{3.9}$$

The first descriptor component $\tilde{z}(0)$ is the contours offset and thus describes the distance to system origin. Removing this component, will make the descriptor translation invariant. The Fourier components can be described with magnitude and phase. A rotation of the contour will lead to a corresponding rotation in its Fourier descriptor. To get the descriptor rotation invariant, only magnitude is used and phase is neglected. Scaling in-variance can be achieved by scaling the descriptor with its first component. The descriptor is $N$-periodic and $N$ elements form the descriptor. A more thorough analysis shows, that the first Fourier components carry the most important information about the gestalt of the contour while higher components describe smaller details. This characteristic makes the Fourier descriptor a good choice for the classification of shapes [23, S. 551 ff.] [17, S. 655 ff.]. Figure 3.11 shows contours Fourier transformed and vice versa. Before determination of the inverse Fourier transform, different subsets of the $N$ components are selected to show the impact.
The third and final part of the feature vector for contour classification, is formed by the Fourier components $u = -20, -19, ..., -1, 2, 3, ..., 20$.

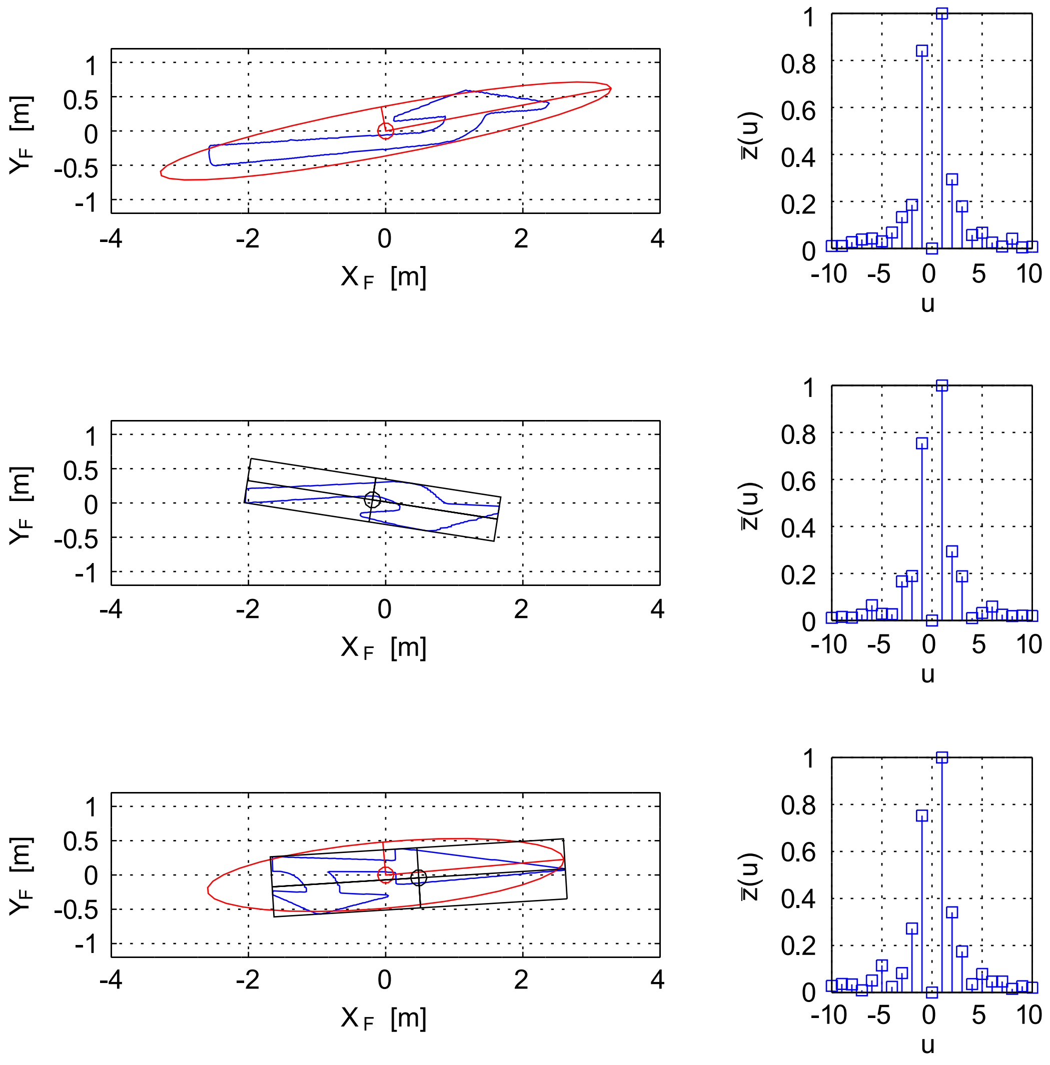
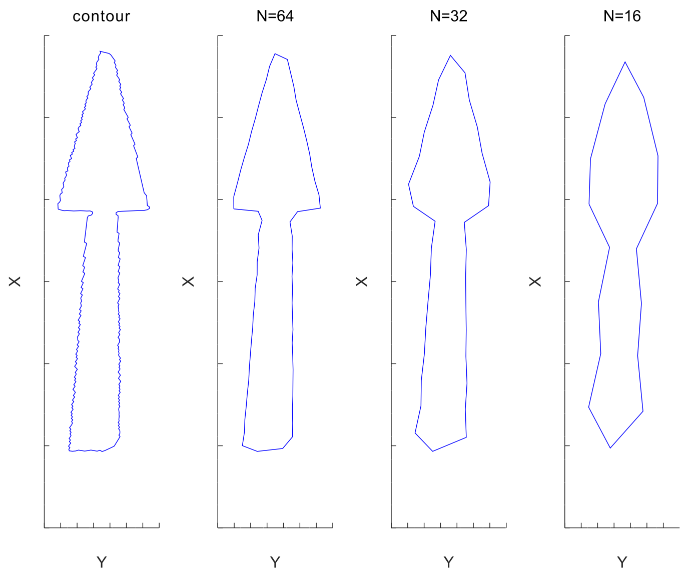
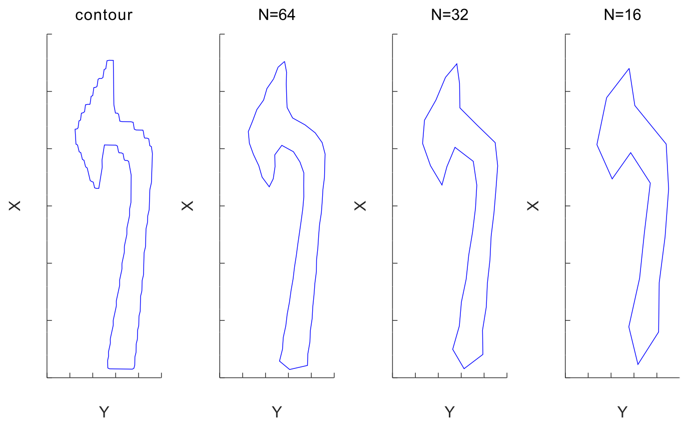


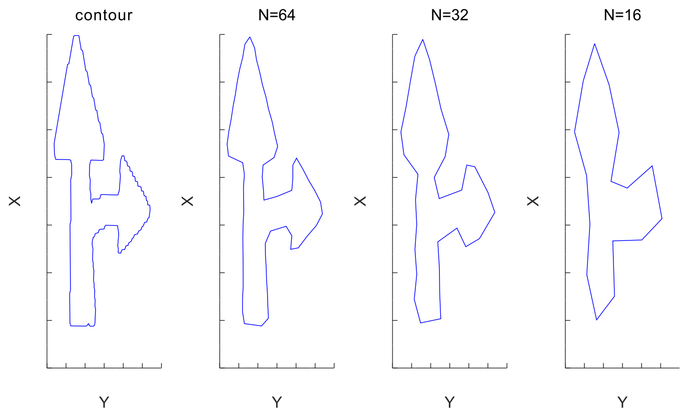
Figure 3.10.: *Illustrated feature subset for Contour examples on the left side. Diagrams on right side show the corresponding Fourier descriptors.*



(a) *contour from straight arrow*



(b) *contour from left arrow*



(c) *contour from straight-right marking*

Figure 3.11.: *Illustration of Fourier descriptors information for different marking classes, with decreasing descriptor size from left (N=64) to right (N=16).*

## 3.3. The HOG descriptor

The Histogram of Oriented Gradients (HOG) descriptor is a widely used method to describe patterns by its edge orientations. The feature describes the counted gradients for a specific image area (*window*). Inside this window, the edge orientations are assigned to a histogram. In practice, the window is divided into cells and the edge orientations are counted in histograms for each cell. The histogram itself is the HOG-feature. The sequence of all cell-histograms for a window, build the feature vector of this window and therefore the feature vector of the image patch to classify.
The basic idea to collect grouped edge orientations in histograms and use the histogram for categorical separation of inputs (classification) was mentioned in a patent from 1986 [31]. The inventor describes in detail, how the features are calculated and how the histograms could be used for classification of patterns. He explicitly mention the use of his invention with images and shows how edge grouping could be done. Furthermore, he shows classification result tables. Actually, the inventor did not named the feature *histogram of oriented gradients*.
The breakthrough of this idea and the brand "HOG" came with a widely respected article from 2005. In [10] the basics of HOG were described and its very powerful use for pedestrian classification in images. Based on the work of [10], many application possibilities for the feature were introduced in a large number of scientific works. It was shown, that the HOG-feature is suitable for a wide field of pattern classification tasks in images. Therefore, the original feature was often extended to meet the specific classification problem.
In this thesis, the HOG descriptor is used in its original implementation, as introduced in [10]. Thus, the following descriptions are orientated on this article. The biggest change for this thesis, is the adjustment of the HOG cell- and block-size to the specific classification problem and the use of a window size that meets better the proportions of markings on urban streets, especially arrows. Therefore, the feature vector has another size, than the one that was proposed in [10] for pedestrian classification. Another difference is the use of gray scale images to improve the processing time. With the use of IPM, the main content of the image will be road surface (gray) and road markings (white). Getting better classification results with the use of color images, thus is not expected. The HOG descriptor

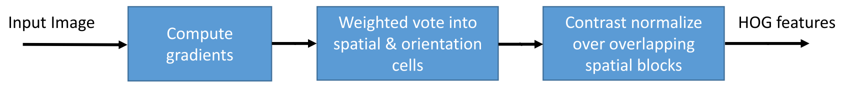

Input Image
Compute gradients
Weighted vote into spatial & orientation cells
Contrast normalize over overlapping spatial blocks
HOG features

Figure 3.12.: *Illustration of the processing chain to get HOG features from an input patch. The chain is similar to [10] but shortened to the essential feature calculation steps.*

bases on the idea to describe and recognize shapes by local gradients. Thereby, the occurrence of gradient orientations is more important, then their positions. [10] proposed a processing chain as shown in figure 3.12. For this thesis, no gamma correction is applied to the input images. To calculate a feature vector from the image, it is divided into cells. The implementation in [10] for example, uses a cell size of $8 \times 8$ pixels. The cells are grouped to blocks, where $2 \times 2$ cells or $16 \times 16$ pixels are forming a block. This is more a data organizing scheme but no processing is made up to that point. In the next step, the image is convoluted with gradient filter masks to expose vertical and horizontal gradients. Dalal and Triggs tried different types of gradient filters. Their conclusion was, that the mask $[-1, 0, 1]$ and its transposed give the best detection performance. The results after convolution are two gradient images $G_h$ and $G_v$. Afterwards, the magnitude and angle for an image coordinate $(x, y)$ can be calculated with

$$\begin{aligned}\gamma(x,y) &= \sqrt{G_h(x,y)^2 + G_v(x,y)^2} \\ \alpha(x,y) &= arctan\frac{G_v(x,y)}{G_h(x,y)}.\end{aligned} \tag{3.10}$$

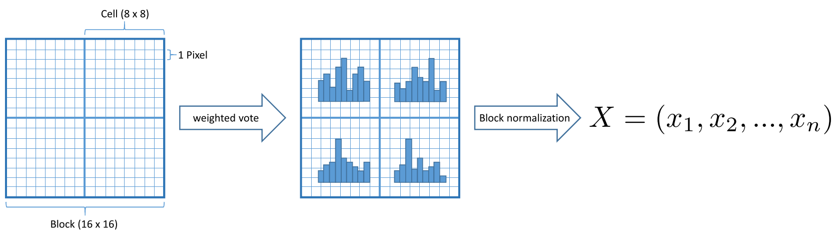


Figure 3.13.: *Calculation scheme for a HOG feature vector X from image patch.*

For each block, now the gradient space is divided into 9 bins between 0 and 180°. The value pairs (magnitude and angle) for each origin-pixel are collected into the histogram bins. Magnitude is used to weight the angle votes. Each histogram corresponds to one cell $(8 \times 8)$ and therefore, consists of 64 value pairs at standard-cell-size. One can expect, that illumination conditions have influence to the hardness of the resulting edge magnitudes. Thus the histograms will be different, even if image patches contain the same type of object. Thus, the histogram values must be normalized. Therefore the cells are grouped into larger blocks and every block is separately normalized. Dalal and Triggs showed, that detection performance increases by overlapping blocks for normalization in a way, that each scalar cell response, contributes several components to the final descriptor vector.

## 3.4. Vehicles motion model

To understand why an odometry model - often called 'ego-motion-model' - is important when dealing with sensor data of a moving vehicle, this section starts with a short introduction to the problem. In the second part of this section, the ego-motion model and its basic equations are shown. For this thesis, multiple camera sensors are used to model a scene understanding. Therefore, no sensor data is used as single snapshot, but fused. The output rate of a computer vision system may correspond to the cameras frame rate. Often, this could not be guaranteed. In this thesis, there are five cameras leading to five parallel processing chains. The computation time for each processing chain is individual and even varies from frame to frame for each sensor. The processing time for each chain depends on

- image size
- image type (color, gray, compressed)
- Region Of Interest (ROI) size
- image content (number of contours extracted)

For a moving vehicle, the task is to link the processed sensor data (contours) to its original position. During process time the car has moved and measured positions by camera projection must be combined with the car odometry for process time span. Thus, it is obvious, that sensor data from moving platforms always should be marked with a timestamp.
This section focuses on how the odometry is calculated for a fixed period of time. To calculate the model, a set of signals from the Controller Area Network (CAN) bus of the test vehicle is used. For example, the signals of the left and right rear wheel tick sensor. For simplification, *period of time* means the cycle time of the corresponding CAN signals. The odometry can be described as a sequence $X_i$ consisting of displacements in $x$ and $y$ direction and a yaw angle $\phi$.

$$X_i = \begin{pmatrix} \delta x \\ \delta y \\ \delta \phi \end{pmatrix} \tag{3.11}$$

The state $X$ can be calculated according to the following equations between consecutive timestamps and the given car geometry [26]

$$\delta\phi = \frac{\delta s_l - \delta s_r}{d_{track}} \tag{3.12}$$

$$\delta r = \frac{d_{track}(\delta s_l + \delta s_r)}{2(\delta s_l - \delta s_r)} \tag{3.13}$$

$$\delta s_a = 2\delta r \cdot sin\left(\frac{\delta\phi}{2}\right) \tag{3.14}$$

$$\begin{aligned}\delta x &= \delta s_a \cdot cos\left(\frac{\delta\phi}{2}\right) \\ &= \delta r \cdot sin(\delta\phi)\end{aligned} \tag{3.15}$$

$$\begin{aligned}\delta y &= \delta s_a \cdot sin\left(\frac{\delta\phi}{2}\right) \\ &= \delta r \cdot (cos(\delta\phi) - 1),\end{aligned} \tag{3.16}$$

with $\delta s_l$ and $\delta s_r$ as the driven distance on left and right rear wheels, $d_{track}$ as the cars track width, $\delta r$ as the resulting curve radius and $\delta s_a$ as the driven arc length.

## 3.5. Tracking with particle filter

The particle filter is a method for Bayes tracking. Bayes tracking is a recursive state estimation of $x_k$, describing the evolution of a sequence $\{x_k,\ k \in \mathbb{N}\}$. It is based on measurements

$$z_k = h_k(x_k, v_k), \tag{3.17}$$

where $h$ is a potentially nonlinear function with process noise $v_k$. A filtered estimation of $x_k$, based on all available measurements $z_{1:k} = \{z_i,\ i = 1, ..., k\}$ should be found. Therefore, the probability density function (pdf) $p(x_{k-1}|z_{1:k})$ must be determined. Assuming $p(x_0)$ is known, it is possible to determine the pdf recursively with two steps: prediction and correction.

The *particle filter* is a sequential Monte Carlo method, where the probability density function for Bayes tracking is defined by a set of samples (particles) and assigned weights. The particles and weights forming the quantized probability density function of the process. With a large number of particles, it is possible to have a sufficient representation for the actual pdf of the process. And therefore, beeing an optimal solution for the Bayes' estimation [2]. The advantage of this method is, that no assumptions have to be made regarding linearity, nor requiring normal distributed process and measurement model. Thus, particle filters and other Monte Carlo algorithms became more and more useful in Bayes' tracking problems in recent years [13].

Having a random measurement
$\{x^i_{0:k}, \omega^i_k\}^N_{i=1}$, that describe the posterior pdf $p(x_{0:k}|z_{1:k})$, with $\{x^i_{0:k}, i = 0, ..., N\}$

as particles set and $\{\omega_k^i, i = 1, ...N\}$ as their normalized weights. Then the posterior pdf of the measurement can be approximated with

$$p(x_{0:k}|z_{1:k}) \approx \sum_{i=1}^{N} \omega_k^i \delta(x_{0:k} - x_{0:k}^i). \tag{3.18}$$

$x_{0:k} = \{x_i, i = 0, ...k\}$ is the set of all states from zero to $k$ and $\delta$ the Dirac-delta-function. To determine the weights for this discrete approximation of the actual pdf, often *importance sampling* is used [2].

The importance sampling uses a *proposal distribution* $q(x)$, from which it is easier to sample, than from the actual probability distribution $p(x)$. If the samples $x_{0:k}^i$ are generated from a proposal distribution $q(x_{0:k}|z_{1:k})$, than the weights can be assigned according to the rate between actual and proposed density:

$$\omega_k^i = \frac{p(x_{0:k}^i|z_{1:k})}{q(x_{0:k}^i|z_{1:k})} \tag{3.19}$$

The proposal distribution can be factorized as:

$$q(x_{0:k}|z_{1:k}) = q(x_k|x_{0:k-1}, z_{1:k})\ q(x_{0:k-1}|z_{1:k-1}). \tag{3.20}$$

Applying this factorization and the Bayes' theorem on $p(x_{0:k}^i|z_{1:k})$ from (3.19), yields to a recursive equation for the weights:

$$\omega_k^i \propto \omega_{k-1}^i \frac{p(z_k|x_k^i)\ p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{0:k-1}^i, z_{1:k})}. \tag{3.21}$$

Since Markov property is given, and with taking the proposal as prior

$$q(x_k|x_{k-1}^i, z_k) = p(x_k|x_{k-1}^i), \tag{3.22}$$

the recursive update term can be reduced to

$$\omega_k^i \propto \omega_{k-1}^i\ p(z_k|x_k^i). \tag{3.23}$$

Note, that this is the most commonly used proposal density, because it can be implemented very intuitively. Furthermore, there is a variety of densities that might be used, also [2]. The filtered posterior probability density can now be approximated with

$$p(x_k|z_{1-k}) \approx \sum_{i=1}^{N} \omega_k^i \delta(x_k - x_k^i), \tag{3.24}$$

while the weights are assigned according to (3.23). It can be shown, that for $N \to \infty$ (increasing number of particles) the equation (3.24) converges to the actual probability density $p(x_k|z_{1:k})$ [2].

The sequential importance sampling (SIS) method, as one possible implementation of a particle filter, that calculates continuously the weights and particles according to (3.23), with measurements arriving sequentially [2]. This implementation often comes with a *degeneration problem.* The weights of all particles but one, converge to null after only few iterations. Thus, much processing time is used for particles that do not contribute substantially to the estimation. It can be shown, that the variance of the weights, derived from a proposal distribution like (3.20), only increases over time. Thus, it is impossible to prevent the degeneration problem. [13].

One method to deal with the degeneration problem is *resampling.* The idea is, that whenever the grade of degeneration passes a threshold, a new set $\{x_k^{i*}\}_{i*=1}^N$ of particles is generated. Therefore, the approximated pdf $p(x_k|z_{1:k})$ given by (3.24) is used, so that $Pr(x_k^{i*} = x_k^j) = \omega_k^j$ is given. Thus, the set of particles is concentrated in regions with high probability, while reducing the number of particles in regions with less probability. Subsequently, the weights are reset to $\omega_k^i = 1/N$ [13]. Indeed, the degeneration can be reduced this way, but there is a trade-off with diversity in the particle set. This is because the finite number of particles and the concentration of them in regions with high probability. The phenomenon is called *sample impoverishment* and is particularly immanent for processes with low noise. In this case, the particle set collapses to a single point during a few updates [2].

If resampling is used, it must be performed periodically. One option is, to determine the grade of degeneration and perform a resampling if necessary. An appropriate grade for the degeneration is the*effective sample size* $N_{eff}$, introduced by [27]. It can not be determined exactly, but there is an efficient way for estimation:

$$N_{eff} \approx \frac{1}{\sum_{i=1}^{N}(\omega_k^i)^2} \tag{3.25}$$

If $N_{eff}$ is below a specific threshold, resampling is necessary. The idea for $N_{eff}$ is, that all weights would be identical, if the samples are derived from the actual distribution. The worse the estimation for the distribution, the bigger the variance in weights. So the effective sample size determines the variation of the weights and thus how precise the particles approximate the actual pdf [19].

As a good threshold, in [19] $N_{eff} = N/2$ is proposed. They evaluated this threshold experimentally and showed that resampling was performed if necessary, while minimizing the risk of replacing good particles.

# 4. Landmark extraction and classification

## 4.1. Landmark extraction

### 4.1.1. Contour extraction

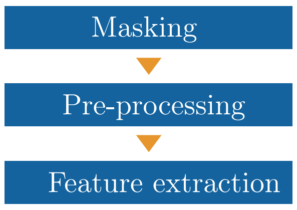


Figure 4.1.: *The computer vision process for feature extraction, condensed to the essential steps.*

The process of feature extraction is divided into three parts 4.1. At first, a ROI is used to mask the relevant parts of the image. This leads to faster processing speed (less pixels to process) and a more robust signal processing, because a large part of the image is kept out and thus interference from that part is prevented. For example a camera mounted to the front of the vehicle will not only capture the road surface, but in urban areas houses and other traffic participants. To avoid extracted contours from this area, that might be wrong classified as road markings, the image is masked.
In the pre-processing step, the image is optimized for feature extraction. Furthermore, distortions and disturbances are removed or weakened. Often, the image color format is converted to an appropriate type for the subsequent extraction algorithms - e.g. to gray-scale. To make a color-based classification of road markings in construction sites, it is useful to keep the color information for the extracted contours in a separate structure. For this thesis, color has no influence to the implemented algorithm.

#### Image masking

A typical way to mask an image is to define a rectangular shape and place it on the appropriate image region. All pixels inside the rectangle will be used in the image processing chain. This way of masking will not be sufficient to map the road surface correctly - especially when using fisheye cameras. Instead, the surface is masked in real-world coordinates or more precise: car coordinates. Hence, every camera of the sensor set has its own defined ROI depending on the view of the camera to the road. The outline of this area is then projected to the corresponding image. As result, there is a unique mask for each camera containing ones for every pixel which projection ray intersects with the defined rectangular area on road surface and null else.

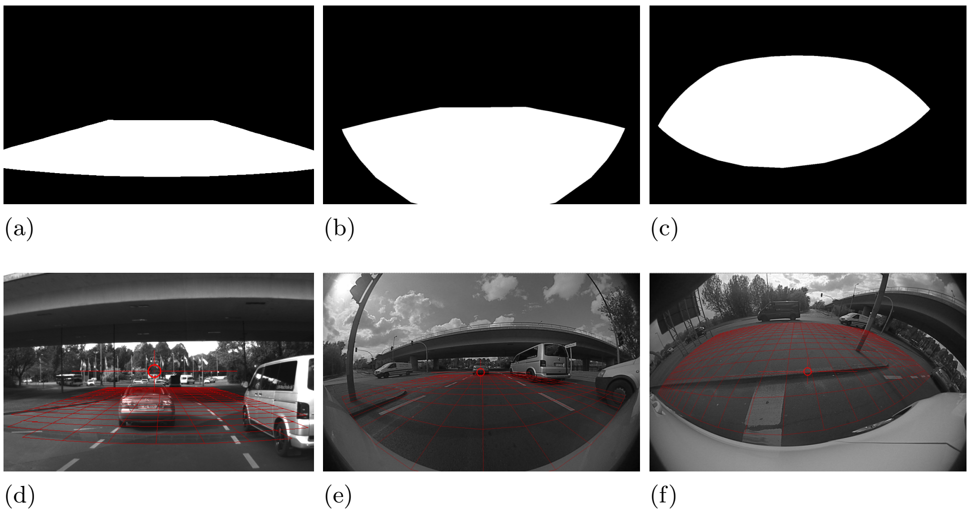
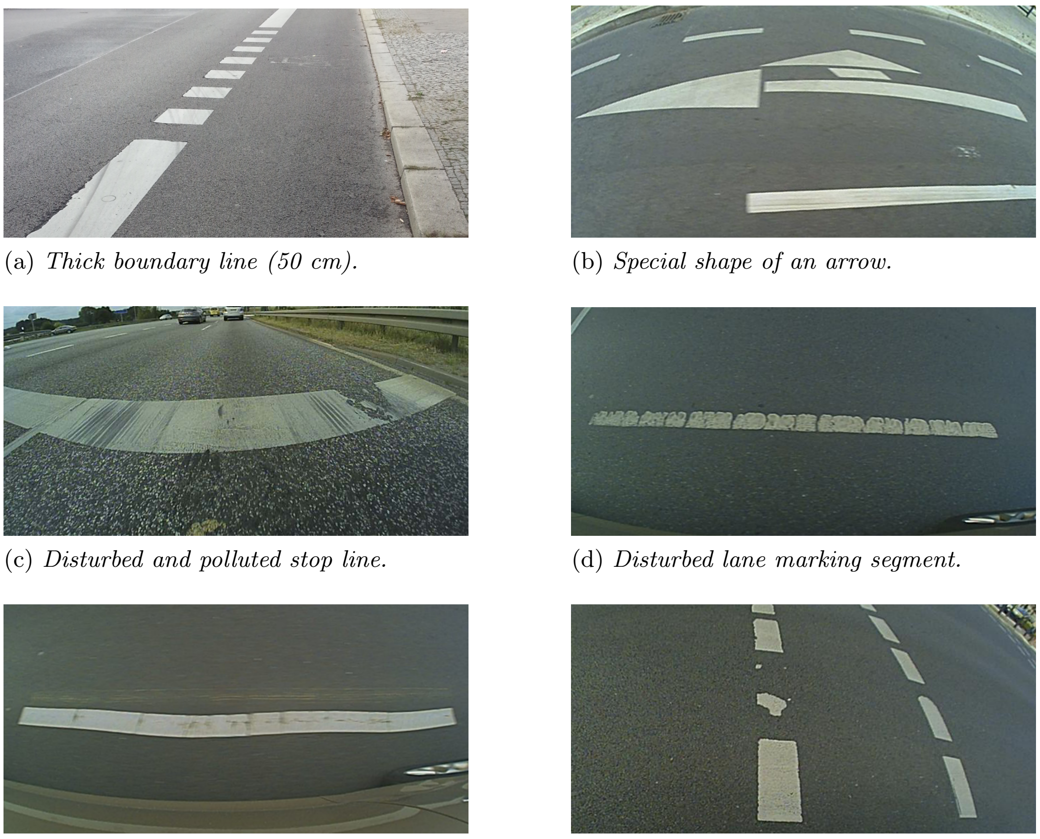


Figure 4.2.: *Examples of image masking. Upper row (a) to (c) shows the image masks in binary illustration and lower row with (d) to (f), the resulting masked area (red grid) for the input images.*

### Image processing and contour extraction

Subject of the image processing chain is the extraction of contours belonging to road markings. The markings are colored white and give a good contrast to the road surface. This is desired for practice, but actually one has to deal with disturbances on real markings, figure 4.3, and image artifacts or blurring.
The fisheye cameras provide images per User Datagram Protocol (UDP) over automotive ethernet. Therefore, the images are JPEG-compressed. Generally, the resulting image quality is sufficient to perform image vision tasks. But with a more detailed view, there are artifacts like blockiness and ringing on the image parts with higher frequency (sharp intensity transitions). For a contour extraction process this will lead to errors and non-closed contour fragments. Figure 4.4 (left image) shows this effect.
The computer vision platform in this thesis is a moving car and thus one has to deal with motion blur. This effect occurs especially on fisheye cameras, because of the mounting position and the special lens characteristic. The impact of motion blur is depending on the cars speed and exposure time of the cameras. For this work, the exposure time is adapted automatically by the camera system and is not provided to the user.
For an optimal preparation of the images for the following feature extraction and to reduce the mentioned effects like compression artifacts and motion blur, the



(a) *Thick boundary line (50 cm).*

(b) *Special shape of an arrow.*

(c) *Disturbed and polluted stop line.*

(d) *Disturbed lane marking segment.*

(e) *Specially curved lane marking segment.*

(f) *Missing line segment.*

Figure 4.3.: *Road markings in practice.*

images are applied with a median filter at first. The median filter calculates the intensity-median out of a desired number of adjacency pixels. That means, the pixels must be sorted. See figure 4.4, the median filter removes the compression artifacts, but keeps the shape and intensity-transition on road markings.
For feature extraction in *lane detection* systems, often a method based on an edge-detection is used. Thus, the use of an edge-based method is examined for this work. The edge extraction is implemented according to Canny [7]. The algorithm from Canny is known as an optimal edge extraction method [23, S. 366]. One disadvantage of this method is the complex parameterization of the two thresholds. Especially, for a continuously changing environment. For this thesis, an adaptive method was implemented to find the best fit for the threshold. It is obvious, that some statistical values, representing the middle intensity of an image area, might help for this problem and that the thresholds should be derived from that value. The threshold finding for $T_1$ and $T_2$, can then be based on mean-intensity

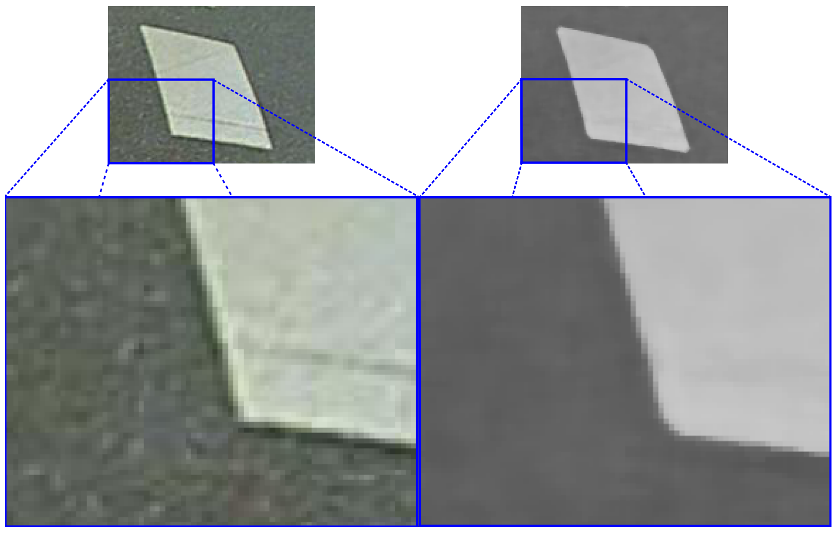


Figure 4.4.: *Comparison of input and output of a median filtered image. At left, the input image before filter. Clearly visible are the compression artifacts like blockiness and ringing at the contour border (light to dark transition). At right, the result after median filter. The dark and light areas are flattened and can be well separated from each other.*

or median-intensity of the investigated pixel area $I(u, v)$ with

$$\begin{aligned} &T_1 = a \cdot Mean(I(u,v)), \quad T_2 = b \cdot Mean(I(u,v)) \\ &\quad \text{or} \\ &T_1 = a \cdot Median(I(u,v)), \quad T_2 = b \cdot Median(I(u,v)) \qquad\qquad a, b \in \mathbb{R} \end{aligned}$$

The values $a$ and $b$ - represent an intensity scale to find the upper and lower Canny-threshold - must be chosen by hand. In practical use, this method has weaknesses using it for the fisheye cameras. With faster vehicle speed, the motion blur leads to softened edges and hence to an insufficient edge extraction with the Canny method. Moreover, the cameras have a very close view on the road surface and thus are prone to disturbed and polluted markings. That leads to extracted edges that do not belong to the actual marking. The result is a non-closed, fragmented set of contour-parts, that must be joined within an additional plausibility step.
To avoid this circumstance, a method was implemented that is not based on edges but on image regions. The idea is, to divide the image into light and dark regions, representing markings and road surface. A marking then is considered as a coherent area of light pixels. Its boundary might be disturbed or polluted, but without heavy impact to the contour extraction. In figure 4.5 the results of both methods, edge-based and region-based, are compared. It is clear, that with region-based method the boundaries of the markings can be extracted more reliably.
To separate light from dark regions, a threshold must be found again. The result is

a binary-image $I_{bin}(u,v)$ calculated from the original image $I(u,v)$, with threshold $T$ and the equation

$$I_{bin}(u,v) = \begin{cases} 0 & \text{for } I(u,v) < T \\ 1 & \text{for } I(u,v) \geq T \end{cases} \tag{4.1}$$

The problem of a best-fitting threshold $T$ must be solved. Since there is no constant illumination and the varying color of road surface, a fix threshold will not be sufficient. One way to find an appropriate threshold, is to derive it from an intensity histogram of the gray-scale image. The result can be ambiguous, if there are no markings inside the considered image area or there are differently colored road surfaces inside the investigated area. Furthermore, lightning effects like shadows can lead to ambiguous thresholds. A way to solve this, is the use of *adaptive thresholding*. In this case not a global threshold is found, but local threshold $T_{u,v}$ for a adjacency region around a pixel $p(u,v)$. Hence, $T$ is no longer constant, but depends on the actual pixel and the region around it. With adding an additional constant to scale the threshold, the separation can be enforced.
The resulting binary image can furthermore be improved with the use of *morphological* operations [23, S. 529-531]. Small fringed structures can be removed with *erosion* and gaps like in figures 4.3b and 4.3d can be closed via *dilatation*. The last step, is to extract the contours from the binary image. This is implemented with border finding algorithm according to Suzuki [48].

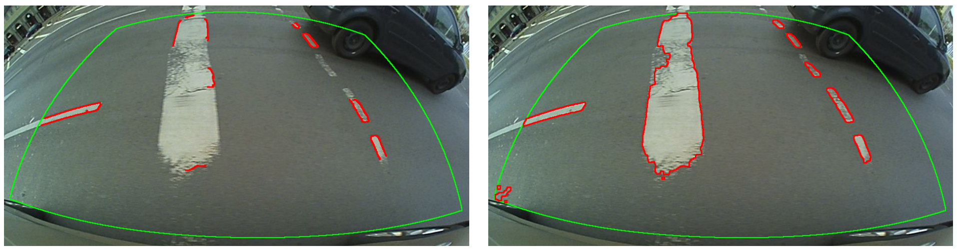

(a) *region based* (b) *edge based (Canny)*

Figure 4.5.: *Comparison of the contour extraction methods a) edge based and b) region based. The finally extracted contours are colored red. Weakness of the edge based method is visible: Not all markings are covered. The disturbed stop line is not detected and only surrounded by some contour fragments. One disadvantage of the region based method can be seen at the bottom left corner of (b): Shiny asphalt (light region) can lead to false extractions.*

## Contour subsampling

The extracted contours, defined as sequences of pixels $c = (\{u_1, v_1\}, ..., \{u_n, v_n\})$ describe the boundaries of the markings at the image plane. Because every camera has special orientation to the road surface and own lens distortion, the contour sequences vary in shape and size, even if they have the same origin in real world. To overcome this effect, the pixel sequences are transformed into the car-coordinate-system 3.1.2. Contours with the same origin will now have the same shape, height and width, but not necessary the same number of points and same point density along the outline. Figure 4.7 illustrates this effect. The shown contour is mapped from adjacent pixels. After transformation into car-coordinate-system, there is a varying distance in $X$- and $Y$-direction between the contour points, figure 4.7b. To make the contour an equidistant sequence of points, the algorithm 1 is used. A resulting contour after normalization is exemplary shown in 4.7c and has the following characteristic:

- All points have the same (constant) distance $d$ to predecessor and successor (except distance between last and first point, that is $\leq d$) - thus, the sequence is equidistant.
- The number of points in a contour can be smaller, larger or the same as on its origin-contour from image, depending on the pixel density on ground plane for each camera (figure 4.6).
- All points are from a segment of the origin-contour.

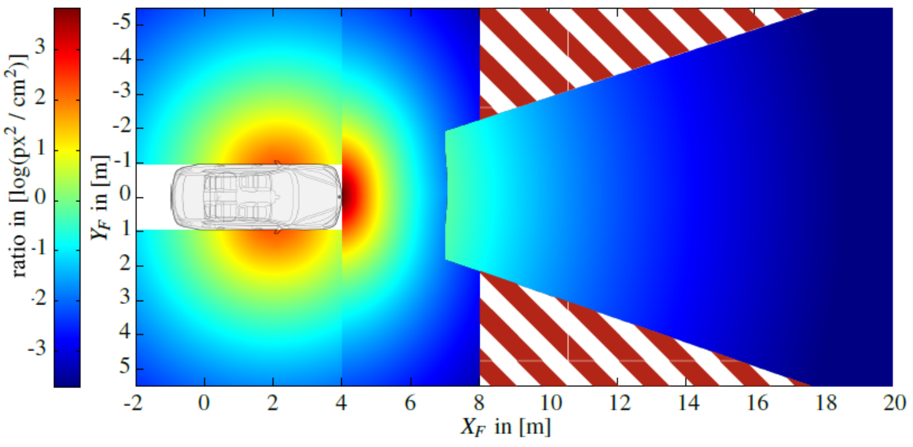
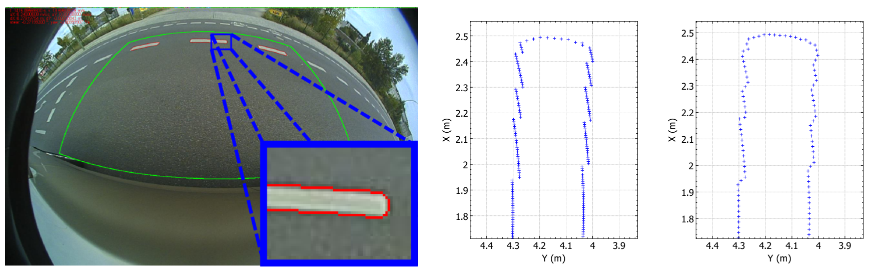


Figure 4.6.: *Pixel to area ratio for the camera sensor setup. Red color means* $e^{+3} \approx 20$ *camera square pixels representing one square centimeter on the ground plane. Blue color means* $e^{-3} \approx 0.5$ *pixels representing one square centimeter on ground.*

**Algorithm 1** Unify the inter-point-distance of a contour

```
contour: $C = \{P_1, P_2, ..., P_N\}$, number of points: $N$
normalized contour: $C_{norm} = \{P_1\}$
current point: $X = P_1$
$i = 2$, $d$ unified distance
while $i \leq N$ do
  if $\overline{XP_i} > d$ then
    add points from $\overrightarrow{XP_i}$ with distance $d$ to $C_{norm}$
    $X$ = last point ($\overline{XP_i}$ is now smaller then $d$)
  else if $\overline{XP_i} = d$ then
    $C_{norm} = \{C_{norm}, P_i\}$ , $i = i + 1$
  else
    $j = i + 1$
    while $j \leq N$ do
      if $\exists P_{new}(\overline{XP_{neu}} = d \wedge P_{new}$is from$\overrightarrow{P_{j-1}P_j})$ then
        $C_{norm} = \{C_{norm}, P_n ew\}, X = P_{new}, i = j$
        leave loop
      else
        $j = j + 1$
      end if
    end while
    $i = j$
  end if
end while
if $\overline{XP_1} > d$ then
  add points from $\overrightarrow{XP_1}$ with distance $d$ to $C_{norm}$
end if
```



(a) (b) (c)

Figure 4.7.: *Contours subsampling. (a) Contours image representation. (b) Transformed contour pixels into ground plane. The varying resolution in distance leads to different distances between the contour points in u- and v-direction. (c) Contour from (b) after subsampling as a sequence of equidistant contour points.*

### 4.1.2. HOG feature calculation

In this section - as second extraction method - the HOG feature extraction is shown. In section before it was shown, that the contour extraction follows a procedure, that starts with retrieving contours from masked images. Thus, some effort is required to prepare the images for a well working contour extraction. Those contours are mapped by camera projection functions into the driving plane and hence into the car coordinate system. A list of equidistant points describing the shape, is finally the *contour* and ready for classification.

With the use of HOG features it is necessary to rearrange this procedure. The incoming images from the cameras are now used without any pre-processing, like tone balancing or blur-filter. With use of the cameras projection model, an IPM image is generated according to the cameras mounting positions. The process of the IPM is explained in section 3.1.5. In short words: It is an image to image transformation that maps the image from origin view to a plane. Typically, the driving plane of the vehicle is used as target. For road marking detection, this plane is adjusted in a way that the current lane and at least the neighbor lanes are covered. With the boundaries of that mapping plane and a chosen resolution, one can generate an image, where each pixel represents a region on road surface. The actual mapping is done by selecting the corresponding camera pixel for every ground pixel. The result is an image that gives the bird-view perspective onto road surface. If the calibration of the camera extrinsic is precise enough, it would be possible to arrange the five images (the sensor set includes five cameras) to one single image. This might be one method for low level data fusion. However, in practice an extrinsic calibration with such a precision is not granted and the images would not fit together without artifacts at the stitching regions. Therefore, the fusion is made in the consecutive marking map module uses the extracted markings.

The task is, to detect and classify the road markings out of the IPM-images. This is done by a scanning algorithm - so called *sliding window*. The parameters of the detection window depend on the object size and IPM image resolution. The classification by HOG features bases on the idea, that the shape of an object could be derived and classified with the use of its orientated edges. This implies a resolution that is fine enough to display a detailed outline of the object. The detection window size is adjusted, that it covers a complete marking with additional space in height and width. The image is scanned by shifting the detection window along the image rows and columns. At every step, the HOG feature vector is calculated for the image area covered by the detection window. The feature vector length is depending on window size, the selected block size and the number of histogram channels:

$$n = n_c \cdot n_h \cdot \left(\frac{w}{b} - 1\right) \cdot \left(\frac{h}{b} - 1\right), \tag{4.2}$$

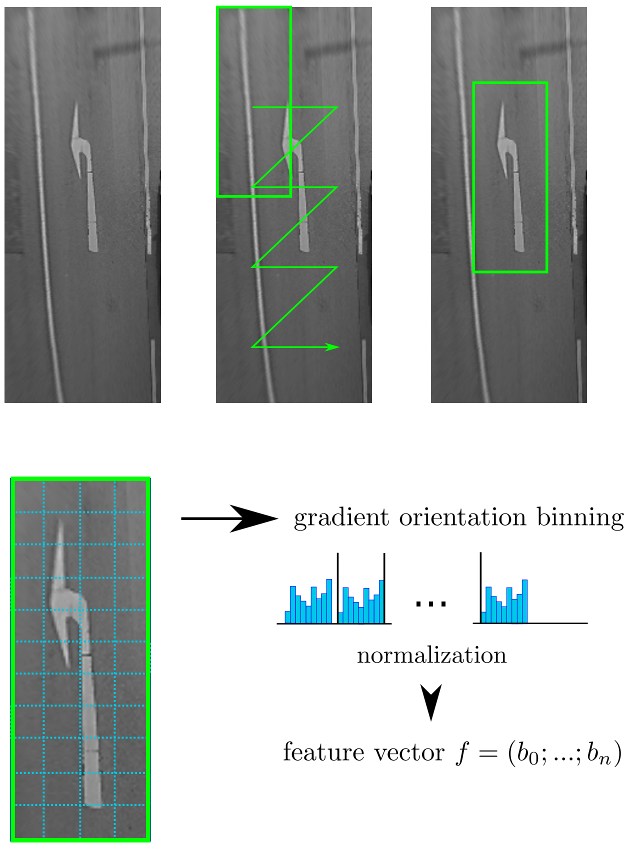

gradient orientation binning
...
normalization
feature vector $f = (b_0; ...; b_n)$

Figure 4.8.: *Illustration of the sliding window and the HOG feature vector calculation of an image patch.*

with $n_c$ as number of histogram channels, $n_h$ the number of histograms per normalization block (standard: $n_c = 9$, $n_h = 4$ ), $w, h$ as window width and height and finally $b$ the selected block size. It is essential, that the block size matches the spatial extent of the descriptive object parts.

## 4.2. Landmark Classification

Referring section 4.1.1, for the first case, the output of the image processing chain is a closed contour as sequence of equidistant $2D$-points in driving plane. Using second method, there is the output of the HOG processing chain, already as a set of feature vectors from the IPM-images. Where each vector represents the descriptor for an image patch from the sliding window. That means, that for the HOG-based classification no further processing to generate features is needed and the descriptor can be used as it is for classifier input. For geometrical analysis, the task is now to find appropriate, optimal descriptive features out of the contours. The contours input is exemplary shown in figure 4.9.

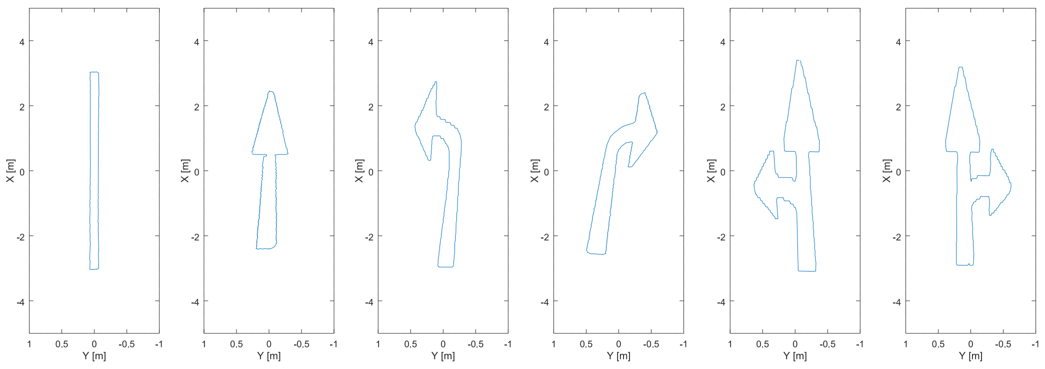


Figure 4.9.: *Contour examples, showing all relevant arrow marking classes, as extracted and classified by the image processing module.*

For a proper interpretation of the vehicles environment, the classification step must separate all incoming contours into classes. The first class is *line segments*. The width and height of these segments should be insignificant for classification. As well as the orientation of the line segments. This will help to classify stop lines as *line* and move the understanding of them into the scene interpretation layer. The arrow markings

- *straight*,
- *left*,
- *right*,
- *straight-left* and
- *straight-right*

will be distinguished in separate classes. Furthermore, there are a lot of contours, that do not fit into one of the defined classes. This occurs, when having false-extracted contours (e.g. other traffic participants) and signs on road-surface like bus-lanes, writings and so on. These contours should be classified as *residual*

*data.* In the following section the used features will be introduced. The features have a varying share on the classification result, hence a feature selection must be performed, to find the best feature set for the classification problem. The procedure is presented in section 4.2.1.
With the feature vectors from the geometrical analysis on one hand and the HOG-feature vectors on the other side, it is possible to use a classifier for separation into the different classes. In section 4.2.2 the parameterization of the classifier is presented.

### 4.2.1. Feature calculation and selection

In 3.2 the contour is introduced as a sorted list of two-dimensional points from ground plane $K = \{P_1, P_2, ..., P_N = (X_{F,N}, Y_{F,N})\}$. Furthermore, different groups of features were introduced, from which a descriptive feature set for contours can be build. The three groups of features are: *moments*, *geometrical features* and *Fourier descriptors*. Table 4.1 summarizes the features from 3.2.

| Group | Description |
|---|---|
| Moments: | normalized, centralized moments $\eta_{20}, \eta_{11}, \eta_{02}, \eta_{30}, \eta_{21}, \eta_{12}, \eta_{03}$ |
| | Hu-moments $m_{H0}$ to $m_{H6}$ |
| Geometrical features: | Roundness $R$ |
| | enclosig rectancle to contour-area ratio $R_{RA}$ |
| | convex hull to contour area ratio $R_{HA}$ |
| | circle to contour area ratio $R_{KA}$ |
| | major to minor axis ratio $R_{AB}$ |
| | gravity balance $\beta$ |
| Fourier descriptor: | $\tilde{z}(u)$ |

Table 4.1.: *Summary of all features (69 in total).*

From the collection of features which should subjectively build a descriptive feature set, now a method is introduced to set the decision to an objective base. Moreover, the metrics will help to reduce the set from features with only small impact to classification. The goal is to have a small number but highly descriptive features for optimal calculation performance and sufficient classification results.

#### Normalization of the feature vector

Using a classifier, where every feature has the same unscaled impact on the class result (e.g. for SVM), the classification result will be unbalanced and biased to features with greater values.
To have all features having the same contribution to the regression problem of the SVM, they need to be normalized. The normalized feature $x^*_{ik}$ can be calculated

with

$$x^*_{ik} = \frac{x_{ik} - \overline{x}_k}{\hat{\sigma}_k} \tag{4.3}$$

with the mean value of a feature over all measurements $i$

$$\overline{x}_k = \frac{1}{N}\sum_{i=1}^{N} x_{ik} \tag{4.4}$$

and the standard deviation

$$\hat{\sigma}_k = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_{ik} - \overline{x}_k)^2}. \tag{4.5}$$

The normalization is applied to every of the $k$ feature variables while preparing the training set. To have the normalization factors for each variable in a representative manner, the training set size should not be too small. The vector of normalization factors, including the mean $\overline{x}_k$ and the standard deviation $\hat{\sigma}_k$ can be used (recycled) to normalize feature vectors of incoming contours in validation phase.
When using decision trees for classification, a normalization of the feature vector is not necessary. The tree does not try to find a separation function with optimal distance like SVM. Instead, at every node of the (random) tree(s) a binary decision over the corresponding variable is made. A scaling of the feature-variable would not have effect to the node-decision at all.

### Feature selection

The goal of the feature selection is to find features with a high separation quality. One evaluation method, is to identify the Fisher discriminant $f$ for features.

$$f = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{4.6}$$

A degree for the class overlapping can be determined by maximizing 4.6. With the mean value of the measurement $\mu_1$ for class 1, respectively $\mu_2$ for class 2 and the corresponding standard deviations $\sigma_1$ and $\sigma_2$, a value for separation or overlapping can be calculated. It is clear, that the metric is suitable for evaluation of two class problems with only one feature. But, from that base, more metrics could be derived for multivariate classes and features.
Therefore, the distribution of the classes itself, as well as the distribution of features in classes must be considered. Because features can correlate, additionally the co-variances must be take into account. To evaluate the features, two matrices are essential: the *intra-class-co-variance-matrix* $\underline{W}$ and the *inter-class-co-variance-matrix* $\underline{B}$.

$$\underline{W} = \frac{1}{K-1}\sum_{l=1}^{K}\left(E\{(\underline{x}_i - \underline{\mu}_i)\cdot(\underline{x}_i - \underline{\mu}_i)^T\}\right) \tag{4.7}$$

$$\underline{B} = E\{(\underline{\mu} - \frac{1}{K}\sum_{l=1}^{K}\underline{\mu}_i) \cdot (\underline{\mu} - \frac{1}{K}\sum_{l=1}^{K}\underline{\mu}_i)^T\} \tag{4.8}$$

$\underline{W}$ is the co-variance matrix over all features and their combinations. The variances for the features are on the matrix diagonal and the other elements are filled with their corresponding co-variances. $\underline{B}$ holds the relation of the classed to each other, by their co-variance on the class-mean.
To derive a numeric value for the separation importance, the *trace-criteria* can be used. The *trace* of a matrix is the sum of its diagonal elements.

$$tr(X) = \sum_{i=1}^{n} x_{ii} \tag{4.9}$$

One can distinguish between two types of trace criteria. For the first trace criterion $J_1$, the traces of $\underline{B}$ and $\underline{W}$, and hence the summed diagonals of the matrices (summed variances) are used to derive an evaluation metric. For the second criterion $J_2$, first the two matrices are multiplied and the trace-operation is applied afterwards. Thus, not only the variance, but co-variances will affect the final result for evaluation.

$$J_1 = \frac{trace(\underline{B})}{trace(\underline{W})} \tag{4.10}$$

and

$$J_2 = trace(\underline{W}^{-1} \cdot \underline{B}) \tag{4.11}$$

For feature selection, $J_2$ is used in this thesis. The trace criterion helps to evaluate the separation importance of features. But it is not a method to minimize the number of features in the feature set. For an optimal feature selection, one has to assess all feature permutations. With an increasing number of features this can be an enormous task and not feasible for practical use. Instead, a non-optimal procedures is used to reduce the number of features, for example the *knock-out-method* or *add-on-method*.
Within the *knock-out-method*, at first the separation importance of all features is aggregated. After that, the lowest performing feature is removed. This procedure is repeated until the desired number of features is reached. The *add-on-method* works in complementary direction. Thus, the feature with the highest importance is added to the feature vector until the desired number of features is reached. The result of the feature selection for this work can be found in 6.1

### 4.2.2. Classification of road markings

The typical way to build a classifier, is to determine descriptive features, for the object that needs to be categorized. This can be achieved, by manually calculated and selected sets as performed for contour classification or with use of an image

related descriptor like HOG. Of course, many more descriptors exist. Another way, can be the use of neural network and derivatives of it. Actually, the hardware in the test car did not meet the requirements that come with greater neural networks. Furthermore, the parametric adjustment and arrangement of neural network layers is an own wide field of research and not the target of this work.
With the found descriptive features, the classifier-creation is divided into training and validation phase. Therefore two separated sets of samples are used, but all have in common, that they are already categorized. During the learning phase, the *training set* is used to feed the classification-machine with input samples. The procedure is like practicing a small child with sentences like: "This is a flower. This is a car. This is a house." and so on, where "this" means the feature vector representing the object. The classification machine finds an optimal relation for separation, according to its principles and its parameterization, see appendix A.4.1 and A.4.2 . To evaluate the classifier performance, the second set of samples, called *validation set* is used. As mentioned, the validation set is already categorized. To evaluate the performance, one usually use a tabular metric where all predictions-and-truth-combinations are registered. Prediction is the value that the classifier predicts for a sample, truth the a priori known category for it. Out of the tables it is possible to derive classification performance indicators like true positives or false negatives and several ratios like the detection-rate.
The focus used for parameterizing a classifier depends on its application. For example, a high false positive rate might be acceptable, since it comes with high detection rates. False positives might be removed in later steps or their influence to a system is low due to its architecture. For this work, an optimal balance between the performance indicators *precision* and *recall* should be found. This can be done, by using the *F-score* of a classifier as a weighted value of the performance indicators,

$$\begin{aligned} recall: &\qquad r = \frac{c_i}{c_i + w_n} \\ precision: &\qquad p = \frac{c_i}{c_i + w_p} \\ F - score: &\qquad F = 2 \cdot \frac{r \cdot p}{r + p}, \end{aligned} \tag{4.12}$$

where $c_i$ for number of correct classification (prediction $i$ meets sample category $i$), $w_n$ for number of wrong missed samples (prediction $\neq i$ does not meet the sample category $i$) and $w_p$ for false positive (prediction is $i$, but sample category $\neq i$).

The data-set content for contour classification is presented in 4.2 and for HOG-based approach in 4.3. It is essential for meaningful results in evaluation, to have no relations between the samples from training and validation. The data-sets must be separated considering that issue.

Table 4.2.: *Data set content for contour based classification.*

| class type | number of samples |
|---|---|
| residual data | 11233 |
| line segment | 9079 |
| straight arrow | 1601 |
| left arrow | 864 |
| right arrow | 864 |
| straight left | 839 |
| straight right | 839 |

Table 4.3.: *Data set content for HOG based classification.*

| class type | number of samples |
|---|---|
| residual data | 3594 |
| straight arrow | 160 |
| left arrow | 52 |
| right arrow | 54 |
| straight left | 42 |
| straight right | 42 |

The two data sets, created for this thesis bases on a large amount of measurement files, captured during drives in urban area at diverse environmental conditions. For learning procedure, all contours were extracted from several independent video frames and in a second step they were categorized manually. The output of this procedure was a set of contours, that build a representative data-set for the classification task. The HOG data-set was created by cutting several samples for all classes out of the measurement data base. The samples were taken to teach a base classifier. This classifier was used to run over further measurement files. The output contained a lot of correct classified samples, but of course a much greater set of wrong classified samples. The wrong class was corrected manually and the - now extended - training set was used to re-learn. This procedure was repeated several times, until the output was precise enough to use the classifier model for road marking categorization in the context of this work. For HOG based classification, the training was focused on arrow markings. Thus, no class for line segments was reserved. There is an obvious reason for this. With the use of the sliding window method, one cannot directly determine the line segments position. The only information is: the line is elsewhere inside the window. This is not sufficient for a precise lane border estimation and thus for lane level localization. In contrast, the arrow marking position needs to be assigned to lanes to determine their category. The center point of the sliding window is precise enough for the lane assignment of arrows.
With the introduced metrics, it is possible to get optimal classification results for the desired use case. The next step, is to evaluate the different configuration parameters of the classifier, for example $C$ for a C-SVM or the number of trees in a random forest. Changing the parameters, often affects the loss minimization problem inside the classifier algorithms in a way, that is not transparent to the user. The validation results may vary a lot, even on small parameter changes. Especially, when heaving multi-class categorization problems. A solution to overcome this problem, is the *grid search* on model parameters or at least the promising parameters with reasonable values. Therefore, one can derive a set of parameter permutations and validate them with randomly selected subsets from validation data. The classification result must then be evaluated. For example with the av-

erage F-score over all classes. More complex cross-validation methods are feasible.

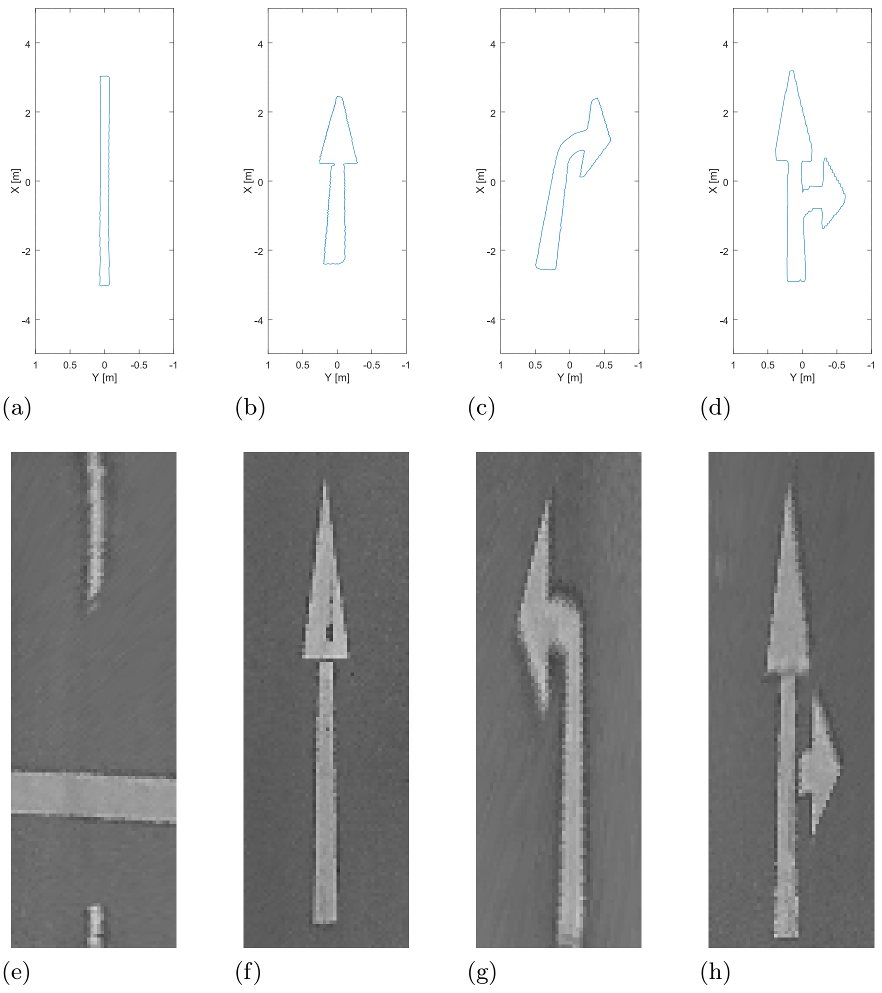


Figure 4.10.: *Exemplary extractions from contour and HOG data sets. The upper row - (a) to (d) - shows examples from the contours training set. The lower row shows image patches for training with HOG features. A class $c = 1$ for line segments as shown in (a), is only available in the contours data set. Patch (e) shows data for residual class $c = 0$ (non-relevant patch).*

## 4.3. Sensor data synchonization and fusion

This chapter strongly refers to the explanations in section 3.4 about the vehicles ego-motion-model and why synchronization is essentially important for moving computer vision platforms like the test car used for this thesis.
As the system consists of un-synchronized sensors, there is a need to compensate the vehicles motion between the signal occurrences. Furthermore, it is not granted, that every processing chain, from input image to the list of extracted markings has similar duration. Neither for the different cameras, nor for a single camera on consecutive images.
The task is, to assign the classified contours to its original position in car coordinate system. To solve this task, an environment map is used that keeps all collected data (classified contours) and therefore represents the current surrounding of the car. Within this map, mechanisms are implemented to store, fuse, track and manage the incoming markings. These will be introduced in the following sections.

### 4.3.1. Synchronized data registration and tracking

The incoming map data is five streams of classified contours from the five cameras. Additionally, the map has access to an ego motion module, from where the motion vector for a specific time-span can be requested. The ego motion module calculates the vehicle motion states according to the signal period of the ESC-CAN-messages and with the equations given in 3.4. Within the motion-module, the motion states

$$X_i = \begin{pmatrix} \delta x \\ \delta y \\ \delta \phi \end{pmatrix} \tag{4.13}$$

are buffered to realize interpolated output states with the same form, but including the fractional parts of the requests. This is needed, because of the varying occurrence and period of CAN- and camera-signals.
Next to the category (line segment, straight arrow etc.), every contour holds its values from the feature calculation before. Some of the features are useful for the fusion task in the environment map:

**mass center** $\overline{P} = (\overline{X_F}, \overline{Y_F})$

**major and minor axis of the enclosing ellipse** $A$ und $B$

**angle of the major axis** $\theta$

**enclosing (rotated) rectangle** center $c = (X_{Fc}, Y_{Fc})$, width $w$, height $h$, angle $\theta_{rect}$

The contour processing and the communication between map and ego motion server is illustrated in figure 4.11.

The following explanations consider one incoming list of classified contours. This represents the classification results from one input image. This is for exemplary reasons and an easier reading. The procedure is the same for every input from a set of $n$ cameras.
Incoming contours are tagged with the time-stamp of the origin image. When a contour list arrives the map, the first task is determine, whether the maps data or the incoming list must be motion compensated. The compensation process always follows a positive time line, and data is always moved to the newer time-stamp. The compensation itself is done by applying the motion state to every contour:

$$f : C_{t_0} \rightarrow C_{t_1 > t_0} \tag{4.14}$$

where $f$ is a function that updates the contour : $C_{t_0}$ at time $t_0$ with $\delta X(t_0, t_1)$ to a new position and orientation $C_{t_1}$. The function is a combined *rotation* and

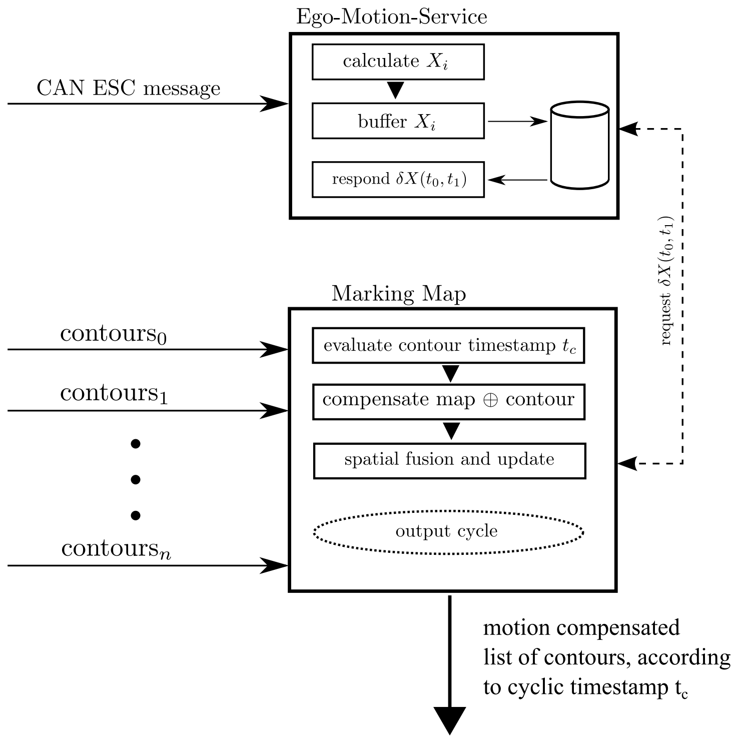

Ego-Motion-Service
CAN ESC message
calculate $X_i$
buffer $X_i$
respond $\delta X(t_0, t_1)$
request $\delta X(t_0, t_1)$
Marking Map
contours$_0$
contours$_1$
contours$_n$
evaluate contour timestamp $t_c$
compensate map ⊕ contour
spatial fusion and update
output cycle
motion compensated list of contours, according to cyclic timestamp $t_c$

Figure 4.11.: *Scheme of the marking map processes and its interacting with ego-motion service. Incoming contours positions will be motion compensated. Afterwards, they are spatially and logically fused with the map content. For motion compensation, the map interacts with an ego-motion service which is able to retrieve the interpolated rotation and translation of the vehicle between two timestamps (out of CAN ESC signals). The interaction is arranged as a client server architecture.*

*translation* on a plane (road surface). It can be described with

$$\begin{aligned} C_{t_1} &= C_{t_0} \cdot R + \vec{t} \\ &= C_{t_0}(x, y, \phi) \cdot \begin{pmatrix} cos\ (-\delta\phi) & -sin\ (-\delta\phi) & 0 \\ sin\ (-\delta\phi) & cos\ (-\delta\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} \delta x \\ \delta y \\ \delta\phi \end{pmatrix} \end{aligned} \tag{4.15}$$

As mentioned, a contour $C$ consists of the contour points and a collection of features. Hence, the compensation may affect a feature completely, but often the $2D$-coordinates only. Since the vehicle motion should be compensated, the equation is used inverse.
After motion compensation was applied (incoming $\oplus$ map), all the data correspond to a common time-stamp. With spatial data fusion, the two lists must now be joined. Therefore, the mechanism explained in the next section 4.3.2 is used. The join will result in three different states a contour from the map can have.

- overlap with new contour $\rightarrow$ confirmed
- not touched $\rightarrow$ not confirmed
- new contour

To avoid the transfer of uncertain data to consecutive modules, the map contours are extended with statistical values to indicate the contours track status. The combination of hit counting with occurrence-age and a plausibility threshold, results in an optimal solution between stability and confidence of the map output. A combination of separated hit-miss-counting was tested, but provided too flickery content at the map output. For map contours, that can be confirmed with the current input, the hit counting increases and the occurrence-age is updated to the current common time-stamp. If there is a new detected marking in input list, it is registered in the map with a unique ID.
For the output cycle, the map content is motion compensated to the current system time. The map is furthermore cleaned from old unconfirmed data. All contours that succeed a plausibility threshold, consisting of a weighted score of their hits and age, are collected for output.
The combination of motion compensation, updating and supervising makes the map to the central object tracking instance of the system. It holds all relevant marking data of the cars surrounding with additional statistics describing the confidence and likelihood for contours existence. Output of the map, is a fused list of contours - out of all tracked contours - with desired confidence.

### 4.3.2. Spatial data fusion

For the merging process, the new arrived contours, must be collated with the existing contours. This is achieved by evaluating the incoming contours for causal and logic conditions in a binary decision tree, with its leaves *'match'* and *'no*

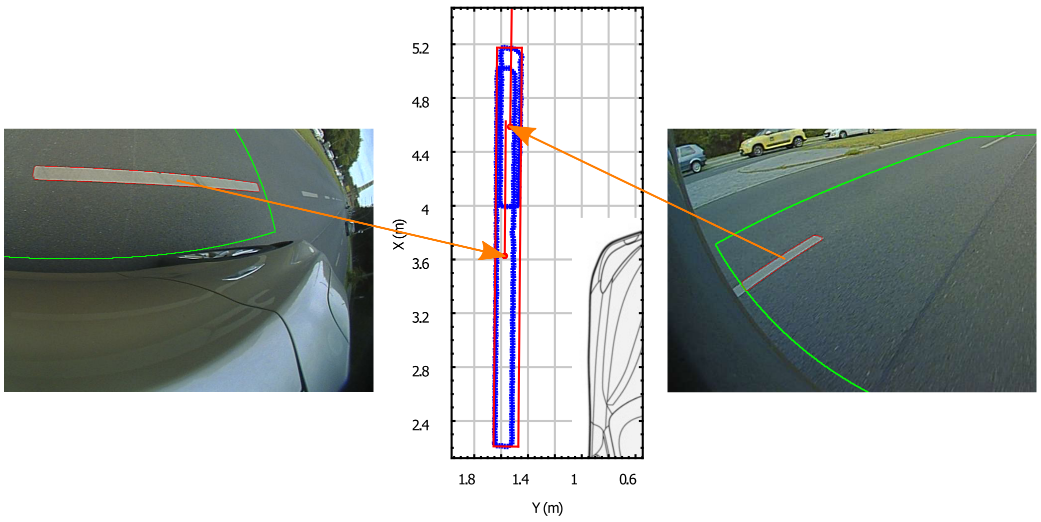


Figure 4.12.: *Fusion of two partial marking detections. The left image shows the camera view of the left fisheye camera. The right image shows the camera view from the front fisheye camera. Both contours (center image, blue color) are merged into one contour in the marking-map with a new enclosing rectangle (red). The green marking shows the masked area on ground for each camera, where the algorithm is active.*

*match'*. The first condition for merging is the class match. In the next step, the angle of the major axis $A$ is checked. Due to the fact, that this angle might be in opposite direction, the criterion is defined as

$$|\cos\alpha| > T, \tag{4.16}$$

where $\alpha$ is the angle between the major axes and $T \in [0, 1]$ the threshold for coherence. If this is given for two objects, the final step is to check, whether the enclosing rectangles of both objects overlap or not. The enclosing rectangles can be rotated and thus, the values for height and width combined with a rotation can have a different meaning. Especially, when comparing contours of lane border segments with contours from crossing lines. To solve this issue, a more universal strategy for the overlapping-check must be found. In this thesis, the *line clipping algorithm* from Cohen and Sutherland [14, S. 113] is used. The idea of the algorithm, is to check, if a line - given by two points - crosses an upright standing rectangle. To use this method for any rectangle both must be rotated, up to one of them is in upright position. Next step is to check, if one of the bounding lines from the other rectangle crosses the upright rectangle. If this is true, the rectangles overlap. Thus, the last condition of the decision tree is passed. If merging is indicated, the objects can be merged to a single object. The enclosing rectangle of the merged object is adjusted in a way, that it represents the common outer border of the two original objects.

## 4.4. Scene Interpretation

From the object level of contours in the marking map, the markings must be interpreted to get an understanding of the cars surrounding and current driving state. The first task, is to find a model that describes the lane borders sufficiently. Therefore, the single line segments (contours) must be grouped to a lane border (line). The used method is described in 4.4.1. Over the grouped contours, a model must be established to interpret and track the lines. The model based lane border estimation and tracking is orientated on nowadays lane detection systems and shown in section 4.4.1.
The next task for scene interpretation, is the determination of the driving prescription from of the detected arrow-markings. When achieved, the lanes - especially at multi-lane-crossings - can be categorized and thus the lane-level assignment to street map is less ambiguous. The lane categorization is introduced in 4.4.2.
Finally, in section 4.4.3, the landmark interpretation and classification is described. Furthermore, it is presented how landmarks can be filtered to consider the measurement noise depending on distance to camera and to furthermore stabilize the measured position.
The results of the scene interpretation are used afterwards for position estimation described in chapter 5.2.

### 4.4.1. Lane border estimation

#### Line segment grouping

Figure 4.13 illustrates the objective of the segment grouping. The output from the marking-map is a list of objects (the classified contours). All objects are treated independently. To determine the lane border, all segments from the contour list must be extracted, that relate to it. All segments, but those from the stop line, have a length to width ratio of 2:1. Thus, the longer side of the segment describes its orientation. This is assured by the marking map.
From the orientation-angle and the width and height of the marking, a direction vector for each marking is determined. With orientation vector and center point of the markings, start- and end-points can be calculated. The calculated values for each marking are used in the following line finding algorithm, as illustrated in figure 4.14.
It starts with a randomly selected object (marking) from the list. From the first object a search range is defined, according to the markings orientation. Subsequently, by iterating over the entire set of markings, the start and end points of all markings in the list are examined. The start and end points within a defined distance to the first object are collected and sorted. In the next step, the marking orientation is checked with the criterion from equation 4.16 for coherent direction. If the examined marking passed the orientation check, it joins the group and the search is resumed from this marking in a similar way. The previous steps are repeated until no further marking can be added to the group. Now, a new search

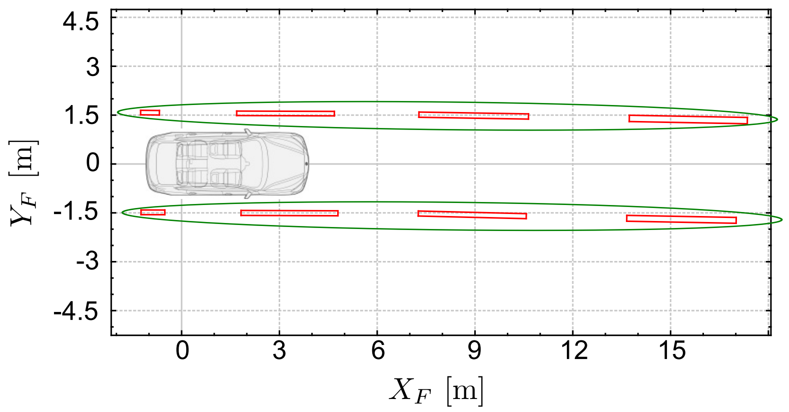


Figure 4.13.: *Tracked marking segments (red) in the marking map. For lane border estimation the line segments must be grouped (green).*

range from the first object of the group (the random one) into opposite direction must be defined and the steps must be repeated. If a resulting set contains at least two markings, it represents a group and thus a line candidate. The process is repeated until every marking was either the start point of a line search or belongs to a line candidate. Markings can be assigned to multiple candidates.

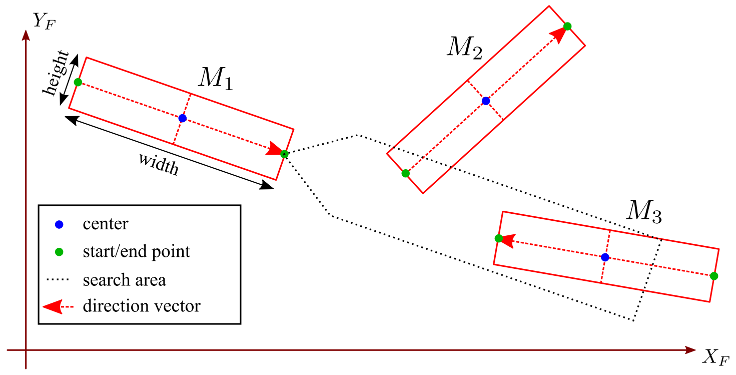


Figure 4.14.: *Line finding: Starting point is the randomly selected marking $M_1$. According to its orientation, a search range is defined from $M_1$. All other marking start- and end-points inside this area are examined with increasing distance. $M_2$ will be neglected, because of the difference in orientation. While $M_3$ is grouped with $M_1$.*

### Model based lane interpretation and tracking

In this thesis, lane borders are interpreted separately. Since lane borders have a continuous path, at least within a limited section, the lane constraint can be estimated between consecutive timestamps with a specific model. The model parameters are estimated with the use of a Kalman filter [24]. A model derived from the lane model introduced in [11] is used. It is defined as

$$y(l) = y_0 + \psi_F \cdot l + \frac{C_0\, l^2}{2} + \frac{C_1\, l^3}{6}. \tag{4.17}$$

The curvature is modeled with a polynomial function with the two parameters $c_0$ (curvature) and $c_1$ (curvature change) as

$$c(l) = \frac{c_0 l^2}{2} + \frac{c_1 l^3}{6}. \tag{4.18}$$

With $c_0 = c_1 = 0$ the border is a straight line. It is important to know, that the function 4.17 is an approximation for a clothoid. Clothoids are generally used for the planning and realization of (well constructed) roads. The approximation is based on the relation $sin(x) \approx x$. Hence, it is only valid for small angles between lane border and vehicle. For driving situations in urban areas, this approximation premises cannot be obtained. Therefore, the function $c(l)$ with $\psi$ must be rotated around the coordinate systems origin. The lateral offset to border is determined by $y_0$. In figure 4.15 the model is shown. A point $P$ on the lane border can be calculated using equation

$$P = \begin{pmatrix} 0 \\ y_0 \end{pmatrix} + \begin{pmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{pmatrix} \cdot \begin{pmatrix} l \\ \frac{c_0 l^2}{2} + \frac{c_1 l^3}{6} \end{pmatrix}. \tag{4.19}$$

To model the state vector $x$ of the Kalman filter, the distance to the border $y_0$ is needed and furthermore the rotation angle $\psi$ and the parameters for curvature $c_0$ and $c_1$. The equation for the Kalman state transition (4.20) describes the process. With $A$ being an identity matrix the model is constant. The state is straightly innovated with data $u = (\delta y, \delta\psi)^T$ from the ego-motion-service 3.4, ($B$ is $4 \times 2$ matrix, with $B_{1,1} = B_{2,2} = -1$, remaining elements $= 0$) and the process noise $w = (w_1, w_2, w_3, w_4)^T$.

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{4.20}$$

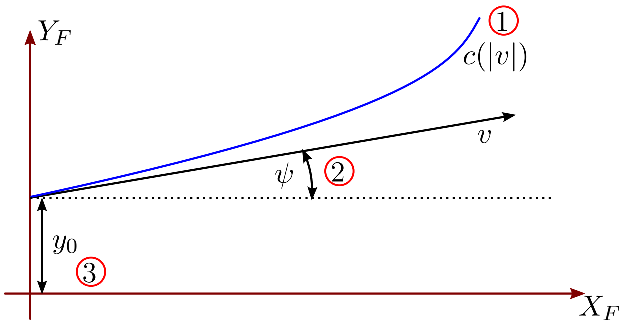


Figure 4.15.: *The path oft the lane model is described with a polynomial function (1). The function values are rotated around the coordinate systems origin (2) and shifted along the $Y_F$ axis (3).*

The measurement step is realized with points along the (curved) path, calculated from 4.19. Since the relation is non-linear, an Extended Kalman Filter must be used. The measurement relation is linearized via a Taylor polynomial of second degree. Therefore, the Jacobian matrix $H$ of the measurement equation is needed. It contains the partial derivatives:

$$H = \begin{pmatrix} 0 & -l \cdot \sin\psi - \cos\psi \cdot \left(\frac{c_0 l^2}{2} + \frac{c_1 l^3}{6}\right) & \frac{-sin\psi \cdot l^2}{2} & \frac{-sin\psi \cdot l^3}{6} \\ 1 & l \cdot \cos\psi - \sin\psi \cdot \left(\frac{c_0 l^2}{2} + \frac{c_1 l^3}{6}\right) & \frac{\cos\psi \cdot l^2}{2} & \frac{\cos\psi \cdot l^3}{6} \end{pmatrix}. \tag{4.21}$$

The Kalman filter steps for one cycle in the scene interpretation module, is explained in the following paragraph. The initial model parameters must be defined previously and should represent a common state. For example, one can suppose that the car is centered on a straight lane ($\rightarrow$ all parameters $= 0$, but $y_0$). Even if this is incorrect, the elements of state vector will converge to the correct values. Constraining the parameters can help to detect false convergence and to initiate a new initialization of the Kalman filter. At first, the state transition is applied with equation 4.20, where $w$ is unknown and neglected. With the new state, one can predict several new measurement points $\tilde{z}_k$ ( where $k = 1, 2, ..., N$ and $N$ as number of measurement points) for a lane border. The generated measurement points are used to select an appropriate line candidate (see section 4.4.1). Therefore, the average orthogonal distance between predicted measurement points and candidate is calculated and the nearest one is selected. The candidate will be dropped, if the distance exceeds a limit. The measurement process is shown in figure 4.16. The found corresponding points from the candidate are finally used for the measurement update of the Kalman filter.

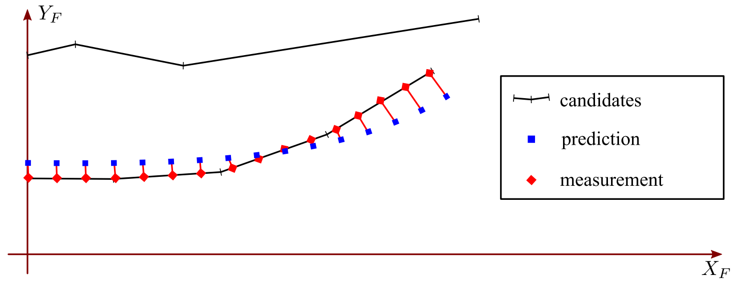
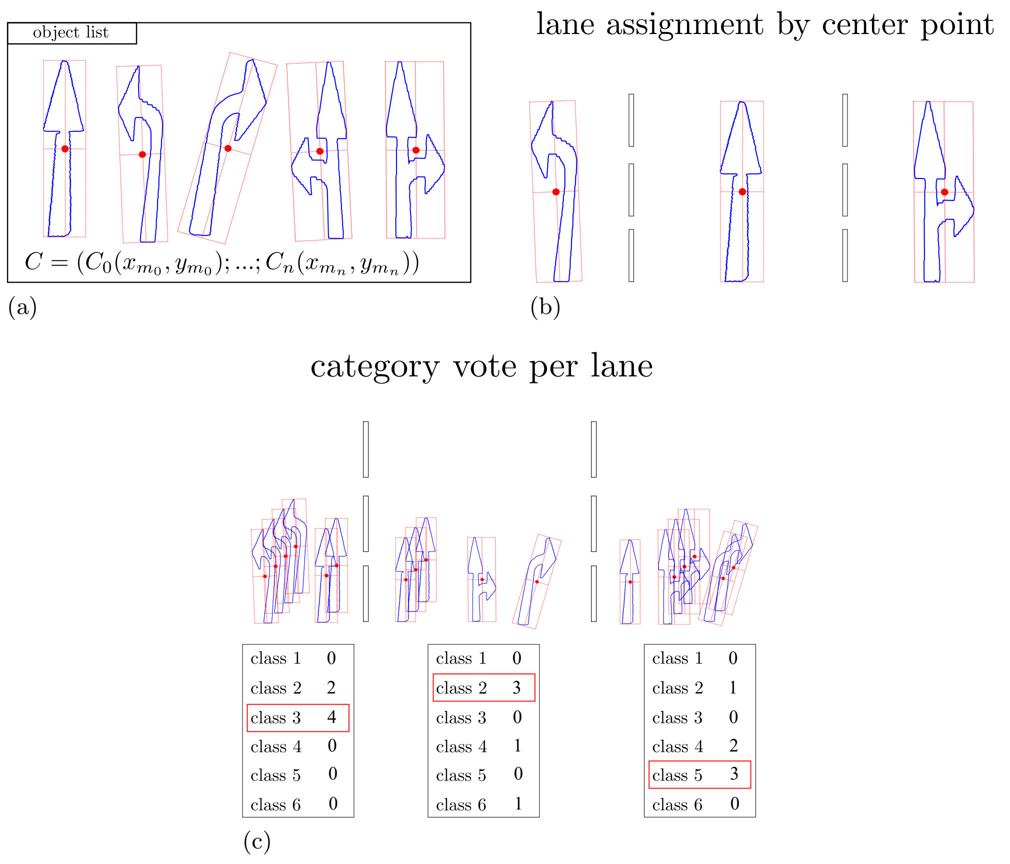


Figure 4.16.: *Kalman measurement for lane model estimation. The nearest candidate to prediction is selected. The measurement is the orthogonal projected points from prediction to candidate.*

### 4.4.2. Lane direction prescriptions

To determine the driving prescription of lanes with arrow markings, at least one estimated lane border must be present. The second can be projected parallel to it, assuming a typical lane width. The classification is implemented, by assigning the detected arrows and their classes to the detected lanes. Therefore, the center coordinate of the marking object (arrow) is used. For every detected lane, the class assignments are counted. This is done for all possible classes. If there was no assignment to a lane for an update cycle, all class counters from that lane decrease. It is obvious, that the number of detections and thus, the level of the class counters, depend on the current vehicles speed. While waiting at a crossing, the counters might be updated permanently with detections around the car. For the class decision, this behavior is acceptable, because of the majority vote. There is the issue, that the decision will last as long as the counters have fully decreased. To avoid such behavior, a miss-counter is implemented, that increases when no detection is registered. After a defined period or number of miss-counts, the lane category is resetted to unknown.


object list
$C = (C_0(x_{m_0}, y_{m_0}); ...; C_n(x_{m_n}, y_{m_n}))$
(a)
lane assignment by center point
(b)
category vote per lane
(c)

| | |
|---|---|
| class 1 | 0 |
| class 2 | 2 |
| class 3 | 4 |
| class 4 | 0 |
| class 5 | 0 |
| class 6 | 0 |

| | |
|---|---|
| class 1 | 0 |
| class 2 | 3 |
| class 3 | 0 |
| class 4 | 1 |
| class 5 | 0 |
| class 6 | 1 |

| | |
|---|---|
| class 1 | 0 |
| class 2 | 1 |
| class 3 | 0 |
| class 4 | 2 |
| class 5 | 3 |
| class 6 | 0 |

Figure 4.17.: *Lane direction prescription estimation. (a) shows the incoming set of all classified arrow markings. (b) The lane assignment is realized with the object center points. If the center point of the marking is inside a lane, it belongs to this lane. (c) Category vote over all assignments for each lane.*

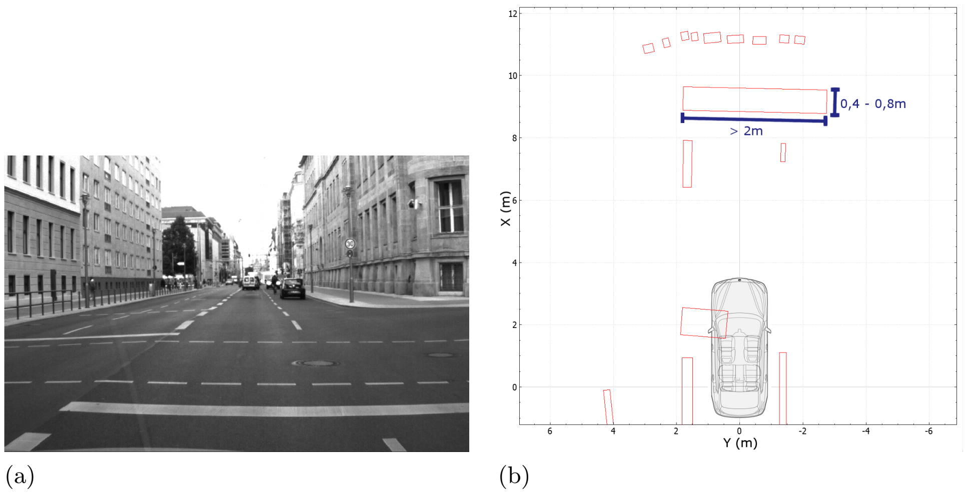
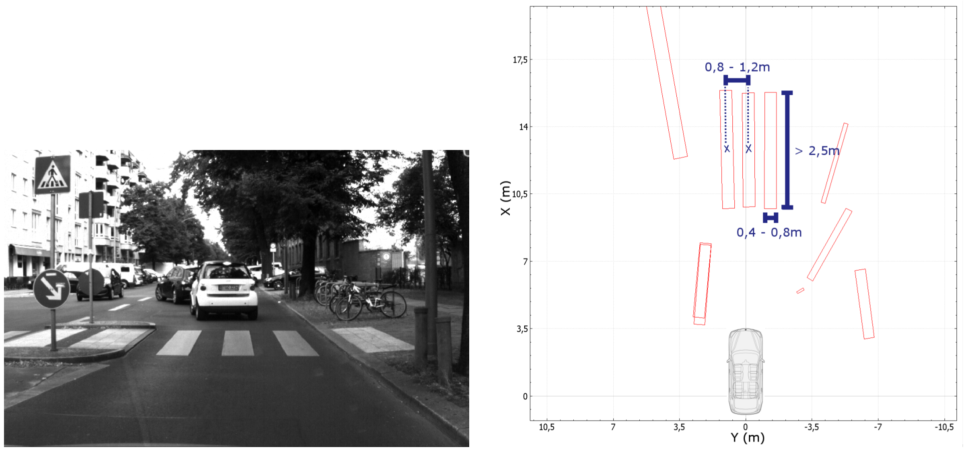


Figure 4.18.: *Illustrated example for stop line detection. a) View from the windshield camera with the stop line in sight. b) Plot of the detected markings in this scene in car coordinate system. The conditions for stop line categorization are shown.*

### 4.4.3. Pedestrian crossings and stop lines extraction

The extraction of stop lines and pedestrian crossings, is based on the contour list provided by the marking-map 4.3. At first, the stop line extraction and verification is explained. Stop lines are extracted from the object list using spatial analysis. Therefore, each marking is evaluated by its two major axes for a minimum length of $2,0m$ and a width between $0,4$ and $0,8m$ (figure 4.18). These conditions are orientated to the specifications in [41] (German guide for road markings) for stop lines, with a required width of $0,5m$ and a minimum length of $2,7m$. In practice, it is useful to apply tolerances to the spatial constraints for extraction. From the detected stop lines, the center points are used for further processing. Because no angle is used for classification, it is possible to extract stop lines not only from the currently driven lane, but from other lanes.

The extraction of pedestrian crossing is performed in similar way, but with adapted constraints. In [41] the specifications for pedestrian crossings are defined as a width of $0,5m$ and a length of at least $3,0m$. Therefore, the spatial analysis uses constraints for width between $0,4$ and $0,8$ meters, and a length minimum of $2,5m$ (figure 4.19) for selection. The output from this selection, is a subset of the incoming markings that meet the requirements. This subset is investigated for parallel markings, by comparing the major axis for a tolerance of two degrees and determining the relative distance between the elements. Therefore, the distance between center points along the two principle axes is evaluated (because of parallelism, the orientation is similar). If the distance for the first component is less than one meter, respectively between $0,8m$ and $1,2m$ along the second component, the sub-



(a) (b)

Figure 4.19.: *Example for pedestrian crossing extraction. a) View from the windshield camera with the pedestrian crossing in sight. b) Plot of the detected markings in the cars coordinate system. The conditions for pedestrian crossing categorization are visualized.*

set is classified as pedestrian crossing. The selected thresholds require parallelism and allow slightly shifted arrangement of lines and their distance and width [42]. Similar to stop line extraction, it is not assured, that the markings always comply specification and thus, additional tolerance is given to the constraints. The result of the extraction, is the center point of the validated line-subset in car coordinate system.

# 5. Landmark based position estimation

## 5.1. Landmark based localization in urban environment

The landmark based localization is a recent field of research. With automatic driving functions, its importance increases especially for complex driving maneuvers in urban areas. For example driving in a desired lane, driving through multi-lane roundabouts, as well as turning maneuvers at multi-lane crossings are some of the probable situations the systems must face.
Several methods were presented to deal with those complex situations. Often, the sensor set of the test car pre-defines which kind of objects can be a useful landmarks. *3D*-measurement as given by LiDAR or stereo cameras, for example, provide the opportunity to detect object outlines. Many research papers deal with the problem of detecting poles from lanterns, signs or trees and use them to estimate a position with use of high density maps. Another approach, is to use walls and corners from houses. An advantage is, that these objects are numerously present when driving through urban areas. A disadvantage of using raised objects as landmarks, is generally the danger of confusion, if objects are too dense and not sufficiently unique. Moreover, there is a danger for ambiguity, especially if reference objects / house corners are occluded by parking vehicles or new infrastructure.

### 5.1.1. Using road markings for localization

Using road markings for localization has one big advantage against other methods. There is a direct link between the source signals and the final position on road. While using trees or house corners for localization, this link does not exist, because landmarks have no logic connection to the currently driven lane. The localization result is generally applied to the reference-map without further relations as a pure geometry problem. A final assignment to the correct lane, can only be performed afterwards with additional logics.
In this thesis, the position estimation is performed with the use of detected road markings and vehicle motion estimation. Thus, the direct link between the map data and the observed markings is given. For position estimation, one can distinguish between lateral and longitudinal correction. The correction is not related to the *easting* or *northing* component of the current position, but to the current driving direction. Lateral correction is the position correction inside a lane and the correct assignment to a lane in a multi-lane situation. The longitudinal correction

is the correction along driving direction. It is performed when passing stop lines or pedestrian crossings. Figure 5.1 shows the position estimation as a schematic illustration with the position variances from longitudinal and lateral correction and the final estimation result marked with $X$.

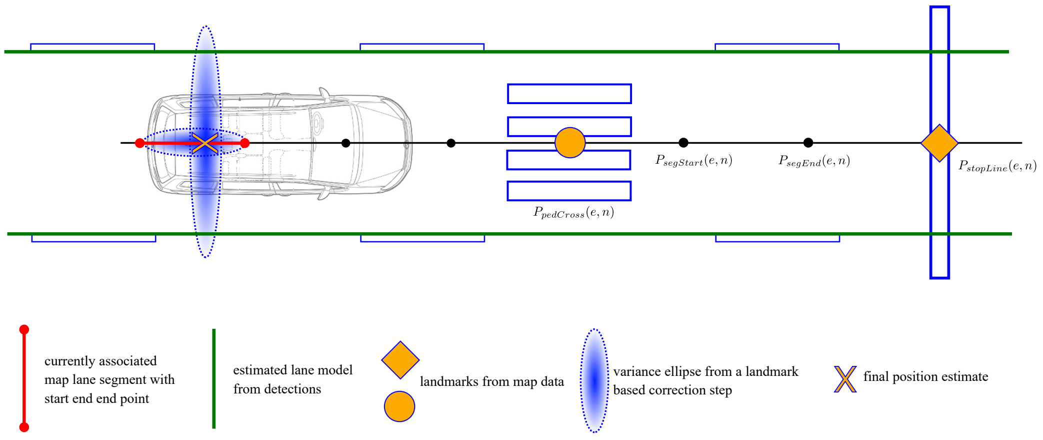

$P_{segStart}(e,n)$
$P_{segEnd}(e,n)$
$P_{stopLine}(e,n)$
$P_{pedCross}(e,n)$
currently associated map lane segment with start end end point
estimated lane model from detections
landmarks from map data
variance ellipse from a landmark based correction step
final position estimate

Figure 5.1.: *Scheme of the localization process with street map interaction. The superimposed variance ellipses for the probable position from landmarks and lane segments are shown in blue. The final (most probable) position estimate is marked with X.*

## 5.1.2. The reference map

To perform the localization, a street map is necessary, where the relevant landmarks are registered with an appropriate precision. To perform lane level positioning and correction the street map must provide data for each lane, driving prescription information and the precise position of landmarks (pedestrian crossings and stop lines).

When starting this thesis, a street map that fulfills these requirements, was not available and must be created by hand. Therefore, an OSM of Berlin, Germany was used as a start and enhanced with the necessary additional information. To make clear, that extended data is meant, the prefix $e$ is defined for $eOSM$ in this thesis. The enhancements consider the map modeling guidelines of the Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Center) (DLR), that have been proposed parallel to this work 2016 in [40]. The work was done in cooperation with major German car manufactures. The guidelines provide a format for detailed street maps, with separately modeled lanes and several landmarks like road markings, traffic signs and -light. The reference points for street nodes and landmarks are given in a global $3D$-coordinate. Thus, it is possible to model three-dimensional road-courses and raised landmarks. Figure 5.2 illustrates how the urban surrounding is modeled in the proposal of the DLR.

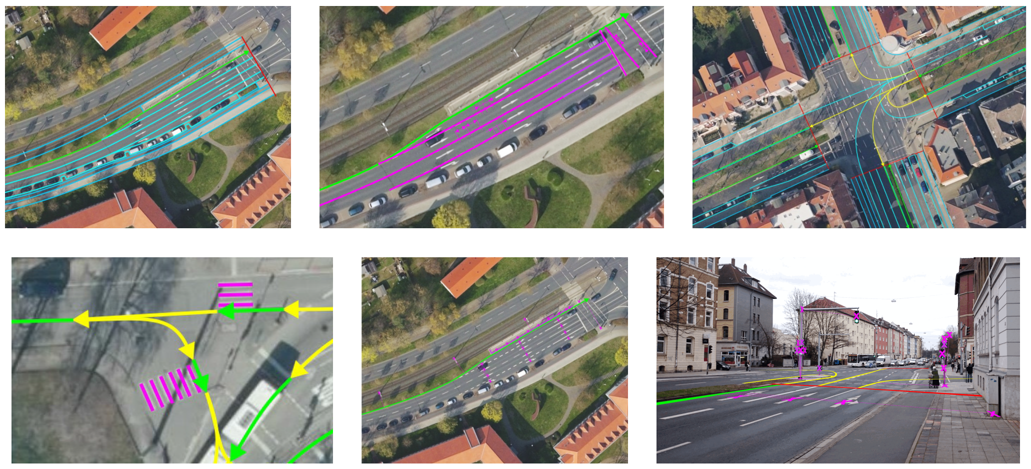


Figure 5.2.: *Illustrations from the Road2Simulation Guidelines. Each lane and lane border is modeled separately. Intersections are modeled with connection-lines. Additional information from pedestrian crossings, stop lines, traffic lights and arrows is included. All images taken from [40].*

### Lane border markings

Lanes are modeled as a sequence of points forming connected segments. Thus, an orientation of the segments is given. A segment is an approximation of the lane course in a constraint area. Each lane is modeled with a sequence of segment points (see Figure 5.1, $P_{segStart}(e, n)$) containing the corresponding easting and northing components. It is important, to have a sufficient segment point density when describing curves.

### Arrow markings

The driving prescription is an attribute of the way definitions in OSM. It is coded as a sequence of direction-notes, separated by a vertical bar character in the *turn:lanes* attribute. For example, the attribute-value for *turn:lanes* of a street with three lanes and the driving prescription left, through, right - marked with arrows in real world - is defined in OSM with the value $left \mid through \mid right$
Because there is a modeled lane for every lane in real world, the *turn:lanes* attribute must be adjusted to have exactly one direction prescription per lane in the $eOSM$ data.

### Pedestrian crossings and stop lines

Pedestrian crossings and stop lines for longitudinal correction must be noticed in the map, also. For pedestrian crossings, this is already provided with the attribute-value-pair $highway : crossing$. A street in OSM generally has a key $highway$ and its rank ($secondary, motorway, etc.$) is given by the corresponding attribute. The crossing position is a data point assigned to a position on the highway with the corresponding attribute-value $crossing$. Distinguishing different lanes, a pedestrian crossing on multi-lanes has multiple intersections (one with each lane). Every intersection is modeled separately. The definition of stop lines is quite similar. The attribute value instead, is not unique in OSM. Typically, for stop lines one of the values $give\_way \mid stop \mid traffic\_signals$ is assigned. To give stop lines a unique key-value-pair the value $marked$ is assigned to the key $highway$ for describe a stop line marking. Analogical to the pedestrian crossing, each intersection of the stop line with a lane is modeled separately. Figure 5.3 shows the part of a map at a multi-lane intersection with stop lines and all extensions done for this work.

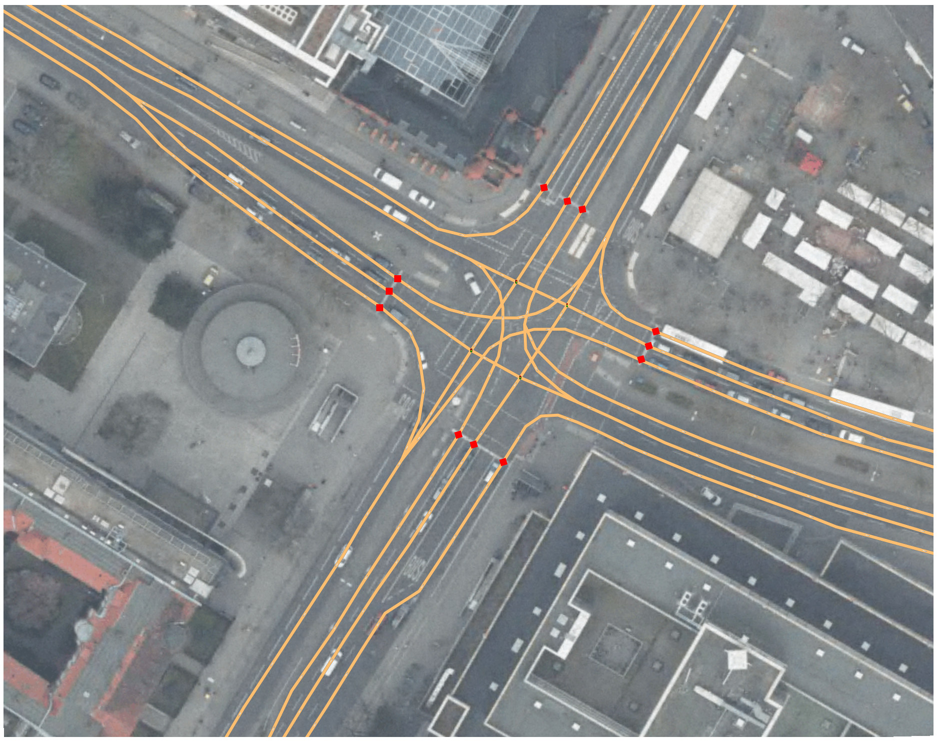


Figure 5.3.: *Illustration of the extended OSM-data at a multi-lane-crossing. All lanes are modeled separately with the use of the lane-middle-path. The stop lines intersections with lanes are shown as red diamonds. Connections on (multiple)turning lanes are modeled in a way, that they describe a natural path and according to guidance-markings, if available.*

### 5.1.3. Map server and local street model

In practice, the localization-module and the map-lookup is realized by a client server architecture. According to the current localization estimate, the localization-model request the server for the current probable lane segments and landmarks. Because of the special requirements, there is no relation between this map server and current available OSM map servers. It was fully designed and implemented according to the requirements of this thesis.

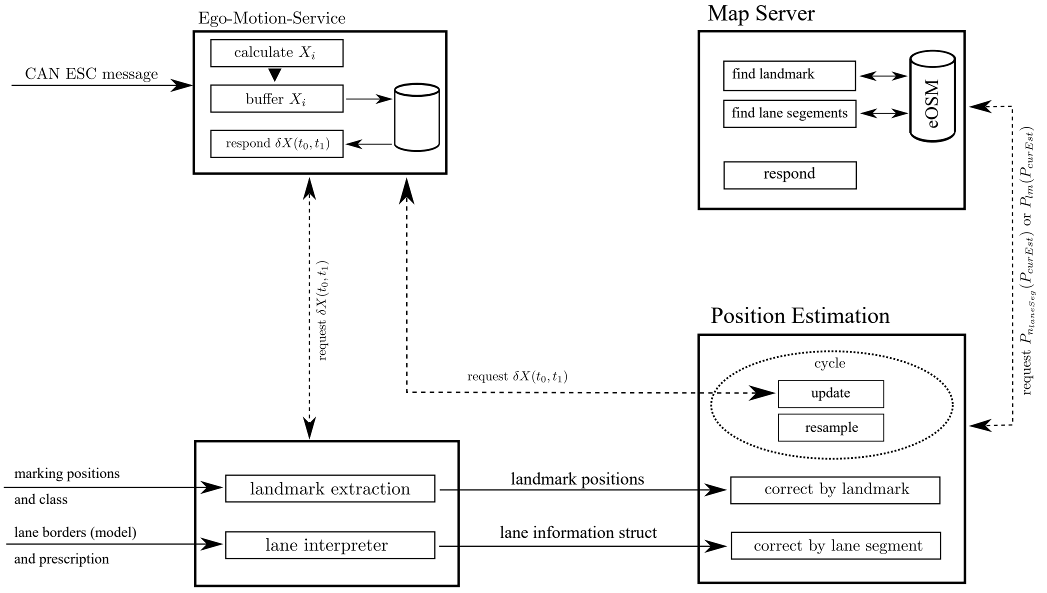


Figure 5.4.: *Scheme of the visual localization module. Further explanations, how interaction works between the components can be found in this section.*

The architecture of the visual localization module is shown in 5.4. The upper left part of the image (*Ego-Motion-Service*) is explained in 3.4. The lower left part contains the extraction and tracking of the landmarks and a re-arrangement of the polynomial model data (e.g. coefficients). The landmark extraction is explained in 4.4.3, the lane model and determination of driving prescriptions in 4.4.1 and 4.4.2. The essential tasks of the map server are

- load and keep updated current map data
- find landmarks
- find probable lane segments.

To prevent long search procedures, the look-up is performed in a small sub-map, that is cyclically updated according to the current position. It is essential, to keep look-up time low. Otherwise, responds from map server might be out of date. All events influencing the position estimation must be queued, including interim

requests to the map server. Afterwards, if correction is performed with use of the responded landmarks, the buffered events can be processed. This yields to the vehicle position (real) drifts away from the estimated system position. To recognize out of date responses from the server, requests and responses must be time stamped. If the response is too late, it should be rejected.
In practice, for a position $P$ the request($\mathcal{R}$)-respond($\mathfrak{R}$) relation for landmarks and lane segments maps to

$$\begin{aligned} \mathcal{R}\left(P(e,n,\sigma^2)\right) &\mapsto \mathfrak{R}\left(\mathfrak{L},\mathfrak{S}\right) \\ &\mathit{with} \\ &\quad \mathfrak{L} = \{L! \mid min(d_{PL})\} \\ &\quad \mathfrak{S} = \{S \mid d_{P\perp S} \leq t_S\} \\ &\mathit{with} \\ &d_{PL} = |P-L| \\ &\quad t_S = n \times \sigma_a + r \end{aligned} \tag{5.1}$$

where $\sigma_a$ is the standard deviation of the first principal component of position estimate $P$ and $r$ a free but fix radius for map look-ups. The respond provides (if exist) the nearest landmark $\mathfrak{L}$ and a sequence of $i$ lane segments
$\mathfrak{S} = \{S_1(P_{start}\ \ P_{end}\ \ c)^T, ..., S_i(P_{start}\ \ P_{end}\ \ c)^T\}$ matching the $n \times \sigma$ -distance threshold $t_S$, along with meta information $c$. The distance evaluation for each lane segment is performed with orthogonal projection from the current position estimate to each segment. Since the segments are constrained by end and start point, the projection can yield to different cases, whether the projected point falls between start and endpoint or outside the segment. If it falls outside, the shortest distance between estimated position and start or end point is provided. The landmark and segment selection is the input for the position estimation, described in chapter 5.2.

## 5.2. Position Estimation

### 5.2.1. Particle filter

The objective of the Bayes position estimator is to provide a distribution that represents the probability dense function for the current vehicles position. In this thesis the dense function can be multi-modal and thus can express multiple peaks. Input for this module, is the vehicle motion and the landmark positions from map, provided by the map-server. To build the error co-variance matrices for the system inputs, the model uses pre-defined measurement variances as described in the corresponding sections 5.2.2 and 5.2.3. The measurement noise for every input is modeled with a normal distribution $p(z_k|x_k^i) \sim N(x_k^i, \Sigma)$. The co-variance matrices $\Sigma$ must be adapted to the characteristics of the measurement system (sensors and modules measurement behavior) and the expected precision of map data.
The representation of the probability dense function is realized with SIS, as particle filter. It contains a set of particles and its weights. Each particle stands for a probable state of the process to approximate. Consequently, it represents the realization of an actual state

$$x_k = \begin{bmatrix} e(k) \\ n(k) \\ h(k) \end{bmatrix}. \tag{5.2}$$

It describes a position in UTM coordinates (*easeasting* and *nasnorthing*) with a heading component $h$. The probability for each realization is defined by the corresponding weight $w_k$. Usually, the particles are spread over a certain state-space, to have particles near all possible realizations of the process. It is obvious, that the number of particles in the filter correspond to the quantization of the state-space and furthermore of the probability dense function. Therefore, the number of particles should have an appropriate size. Otherwise, an increasing number of particles leads to proportional increasing processing effort. This is typically the trade-off for particle filter. The advantage, and therefore the reason to chose this kind of filter is, that it approximates a probability dense function for the complete state space and furthermore, this function do not need to be Gaussian, but can be multi-modal.
Output of the module is the current estimate for position, as the weighted state of all particles. This is equivalent to the minimum mean square error (MMSE) estimate of the system state.

$$\hat{x}_k = \sum_{i=1}^{N} \omega_k^i \; x_k^i \tag{5.3}$$

The corresponding variance for this estimate is the weighted co-variance matrix $\hat{\Sigma}_k$ of the particle set. It is equivalent to the mean squared error (MSE) matrix.

$$\hat{\Sigma}_k = \sum_{i=1}^{N} \omega_k^i \; (x_k^i - \hat{x}_k)^T \; (x_k^i - \hat{x}_k). \tag{5.4}$$

The variance of the heading component is the third entry on the diagonal of $\hat{\Sigma}_k$. For position variance in easting and northing, the eigenvalues of the upper left $2 \times 2$ sub-matrix from $\hat{\Sigma}_k$ have to be determined. The two eigenvalues correspond to the two major components of an ellipse. The ellipse outline describes the $\sigma^2$ distance to the MMSE estimate.

### 5.2.2. Position prediction with vehicle motion

The prediction step of the particle filter is realized with the vehicles motion information. As described in 4.3.1, the vehicle motion for a specific period can be requested from the ego motion service. The cycle time for the particle filters update is adjustable and motion vectors can be interpolated. With the responded vehicle motion $(\delta x \; \delta y \; \delta\phi)^T$ every particle is updated in its position and direction. Therefore, the latest estimate for direction $(h)$ is used to transform the motion vector from car coordinates to UTM coordinates.

$$x_k = x_{k-1} + \begin{bmatrix} \sin h_{k-1} & -\cos h_{k-1} & 0 \\ \cos h_{k-1} & \sin h_{k-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \\ -\delta\phi \end{bmatrix} + e \tag{5.5}$$

The yaw-part $\delta\phi$ is negatively signed, because *'heading / yaw'* in the UTM meaning is defined in opposite direction to the vehicle coordination system.
Additionally, an error $e$ is added, according to the measurement noise of the ego motion estimation process. The noise is modeled as multivariate normal distribution $p(e) \sim N(0, \Sigma)$ with co-variance $\Sigma$.

$$\Sigma = \begin{bmatrix} \sigma_e^2 & 0 & 0 \\ 0 & \sigma_n^2 & 0 \\ 0 & 0 & \sigma_\phi^2 \end{bmatrix} \tag{5.6}$$

To generate discrete samples from distribution $N(0, \Sigma)$, a *Cholesky factorization* is used. The sampling process is explained in appendix A.5.

### 5.2.3. Position correction

The correction step of the particle filter is realized with the use of GNSS, the position of landmarks and lane segments. Therefore, the particle weights - representing the probability dense function - must be adapted with the recursive update rule from (3.23). The result is, that particles gain an increasing weight, if it is more probable, that with the current measurement they represent the actual state.

Each single measurement and its variance is modeled with a specific normal distribution. The variance modeling for the different types of measurements will be described in 5.2.3 to 5.2.3. The probability $p(z_k|x_k^i) \sim N(x_k^i, \Sigma)$ of the particles $x_k^i$ belonging to a measurement $z_k$ is calculated from the dense function of multivariate normal distributions

$$p(z_k|x_k^i) = \frac{1}{\sqrt{(2\pi)^{dim}\ |\Sigma|}}\ \exp\left(-\frac{1}{2}\ (z_k - x_k^i)^T\ \Sigma^{-1}\ (z_k - x_k^i)\right). \tag{5.7}$$

Where $|\Sigma|$ is the determinant of $\Sigma$ and $dim$ the dimension of the measurement. For landmarks $dim = 2$, because the detection system can not extract an orientation from landmark measurements. For GNSS and lane segment measurements an orientation is present and therefore $dim = 3$ in this case. To determine the difference between measurement and particle for the state component $h$ it is recommended to use vector geometry. Because, when $h$ describes the circular orientation with $0° \leq h < 360°$ there exists a discontinuity at 0. For example a measurement $z_k$ next to state $x_k$ with orientations $z_k(h) = 358$ and $x_k(h) = 2$ should cause a weight increase from (5.7). Instead, the straight way to determine the difference by angle subtraction, will overestimate the difference and thus, leads to wrong weight for the particle $x_k$. After updating all particle weights with (5.7), the weights in the particle set must be normalized to meet the requirement

$$\sum_{i=1}^{N} \omega_k^i = 1, \tag{5.8}$$

with $N$ as number of particles.
Afterwards, the particle set is examined for an indicated resampling, 5.2.5.

### The initial position estimation

The use of GNSS in this thesis is separated into two operation modes. On the one hand, the GNSS can be used to perform a periodical correction update on the particle set, like other periodical or event-triggered inputs. On the other hand, the input can be used to only perform an initial position guess and without any further updates.
The initial guess, is the most important reason to use GNSS information in this thesis and consequently the normal operation mode of the positioning module is the second without periodical GNSS updates.
When system starts, no information about position exists. This is called 'the lost robot problem'. For a constrained area, it would be possible to retrieve this information solely from the particle filter with a sufficient large number of correction cycles. For a vehicle based system, with its large possible state space, another option should be chosen. In this thesis, the initialization for the first state (position and heading) is performed with GNSS information. Hence, the error variances for this input is modeled separately and with knowledge of the typical deviations of

the signal, all existing internal filters and motion models inside the GNSS hard-/software should be switched off. The information, received from the GNSS hardware should contain the triangulated localization solution without any additional filters. The messages, received from the hardware follows the National Marine Electronics Association (NMEA) 0183 standard, [3]. The essential information for a measurement update is given with

$$z_{GNSS} = \begin{bmatrix} e \\ n \\ \phi \end{bmatrix}, \tag{5.9}$$

referring to the state space with easting, northing and heading. Subsequently, the initial position measurement is used to sample the first set of particles and set the local map radius of the street map for the following look-ups.

The expected noise of an GNSS measurement is modeled with

$$\Sigma_{GNSS} = \begin{bmatrix} \sigma_e^2 & 0 & 0 \\ 0 & \sigma_n^2 & 0 \\ 0 & 0 & \sigma_\phi^2 \end{bmatrix}, \tag{5.10}$$

where each component should use a typical deviation for the solutions received from the GNSS hardware, eg. $\sigma_e = \sigma_n = 20.0\ m$ and $\sigma_\phi = \pi/18 \equiv 10°$.

### Static landmarks - longitudinal correction

For longitudinal correction of the vehicles position the landmarks *stop line* and *pedestrian crossing* are used. At first, it must be defined which position on these landmarks should be the reference point for corrections. As shown in figure 5.5 and 5.1, the reference point of a landmark is defined as its center. In multi-lane situations, the reference point is the intersection of the landmark with the lanes center.
The error modeling for landmarks must consider different varieties of errors, constant and dynamic. Thus, for every landmark based correction update, a separate co-variance matrix $\Sigma_{lm}$ must be determined. The constant part $\sigma_c$, to model the measurement noise is the maps precision. The high density parts of the street map were modeled handcrafted, using Digital Orthophotos (DOP)-20 areal photos. It means, that the width of one pixel on the photo equals 20 centimeters in real world. Assuming an error for the hand crafted part of one pixel, the constant part can be taken as $\sigma_c = 0.2\ m$.
The dynamic part of noise modeling is the distance-dependent precision of the optical measurement system, $\sigma_d$. It is obvious, that the precision decreases with increasing distance to the landmark. The relation between image row $v_i$ and distance is inverse proportional, $d_i \propto 1/v_i$. Considering a flat world, with camera parallel to the surface and without lens distortion, the row to row deviance mapped

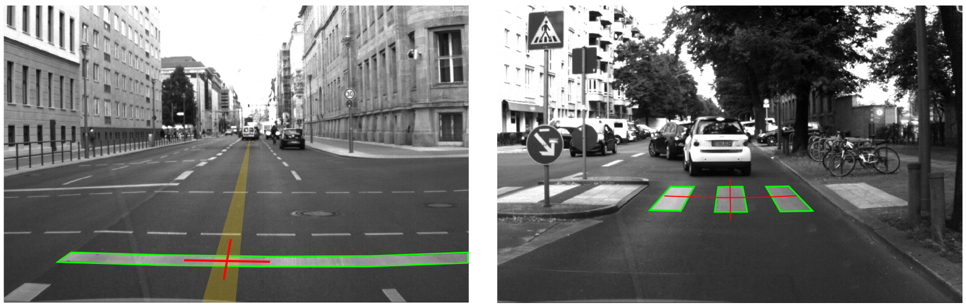

(a) (b)

Figure 5.5.: *Reference points of landmarks. a) Shows a stop line as detected (green) and the related reference point (red cross). b) Shows detection (green) and reference point (red) of a pedestrian crossing. Additionally, in (a) the measured lane center is shown (orange).*

to ground plane can be determined with

$$\begin{aligned} tan\ \alpha_i &= \frac{d_i}{h_i} = \frac{f_x}{v_i - c} \\ d_i &= \frac{f_x\ h_i}{v_i - c} \\ \sigma_d &= d_i - d_{i+1} \\ &= \frac{f_x\ h_i}{v_i - c} - \frac{f_x\ h_{i+1}}{v_{i+1} - c} \qquad \text{with } v_i > c, \end{aligned} \tag{5.11}$$

with $f_x$ $[px]$ as focal length, $v_i$ as image row greater than $c$, the v-coordinate of the principal point and $h_i$ the height of an image row over surface. With the scales of an automotive camera system, one can say $h \approx h_i \approx h_{i+1}$ without making noticeable error. See figures in A.1, to get a detailed view to the relations between image plane and ground plane. In sum, the error in driving direction or $x$ in car coordinates, can be described as co-linear error $\sigma_{col}$ with

$$\sigma_{col} = \sigma_d + \sigma_c. \tag{5.12}$$

Additionally to the co-linear error, an orthogonal error $\sigma_{orth}$ exists. This error component depends on the standard deviation of the current heading estimate and the measured distance to the landmark.

The reason is: within the longitudinal correction step with landmarks, the current estimated heading component is used. A deviation $\sigma_h$ on this value will lead to a distance ($d$) depending error in orthogonal direction. The error $\sigma_{orth}$ can be determined with the trigonometric relation

$$\sigma_{orth} = \sigma_{col} + \tan(\sigma_h) \cdot d. \tag{5.13}$$

The two error components $e_{col}$ and $e_{orth}$ can be used to model the variance ellipsoid for a landmark measurement. As the position estimation is performed in the UTM system, the matrix must be rotated from car coordinates to UTM coordinates with rotation matrix $R$

$$R = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}, \tag{5.14}$$

where $\phi$ is the angle between the major component of the ellipsoid and the easting axis in the UTM system. Finally, the co-variance matrix of the landmark measurement, describing a rotated variance ellipsoid, is given with

$$\Sigma_{lm} = R \cdot \begin{bmatrix} \sigma^2_{orth} & 0 \\ 0 & \sigma^2_{col} \end{bmatrix} \cdot R^T. \tag{5.15}$$

### Lane borders - lateral correction

Consider the request to the map server for current probable lane segments. The response is a list of map-lane-segments around the current estimate with position and meta information. There are two open tasks. Model the variance ellipsoid

$$\mathcal{E} \sim N(\epsilon, \Sigma_{lane}),\ \epsilon \sim z_k \tag{5.16}$$

with the ellipsoids position and orientation $\epsilon$ ($z_k$) and c-ovariance matrix $\Sigma_{lane}$ for each lane segment.
And as a second task, validate and score the ellipsoids before applying them to the particle set.

To model the variance ellipsoids, the currently measured deviation from lane center is determined, using the lane model provided by scene interpreter module. The current estimate is used to perform an orthogonal projection to the map segment. That point on projection, which distance to segment equals the determined lateral deviation, defines the center of the variance ellipse. The measurement error $\sigma_{lon}$ in longitudinal direction, is modeled using the minimum from map-segment-length and $n \times \sigma_e$ of the current position estimate. It describes the first component $a$ of the variance ellipse and thus the first dimension of the probability density function. The lateral error $\sigma_{lat}$ is the second component $b$ of the ellipsoid. It is determined, using the current lateral process noise, or the given lane width $w_{map}$ from map as maximum value. Finally, the orientation angle inside the lane, as given by the lane model, defines the ellipsoids rotation. Figure 5.6 illustrates this procedure and values.
To consider a map defined road model error, that describes the precision of the map data, the three error dimensions $\sigma_{lon}$, $\sigma_{lat}$ and $\sigma_\Phi$ can be extended by adding a corresponding constant $\sigma_c$. According to the given propositions and equations, the error co-variance matrix for lateral correction updates is given with

$$\Sigma = \begin{bmatrix} \sigma^2_{lon} & 0 & 0 \\ 0 & \sigma^2_{lat} & 0 \\ 0 & 0 & \sigma^2_\Phi \end{bmatrix}. \tag{5.17}$$

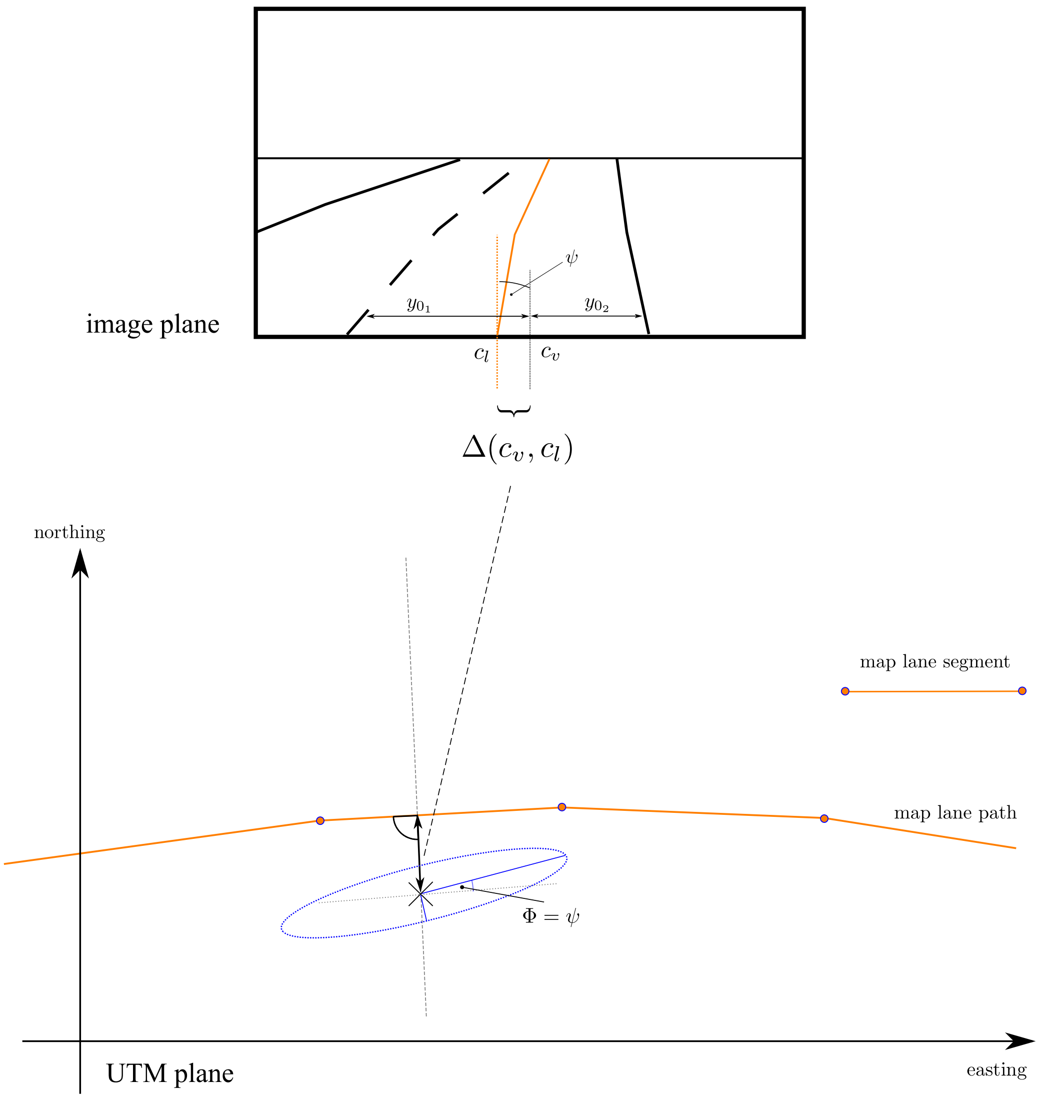


Figure 5.6.: *Position error modeling for lane measurements. The upper image shows the image plane. The map reference is colored orange. $y_{0_i}$ are the deviations to the left and right lane border as measured by the system. $c_v$ is the vehicles center position and $c_l$ the center of the map lane. The difference $\delta(c_v, c_l)$ is used for the orthogonal projection onto the map lane segment in UTM plane (lower image). The resulting point (X) is the center of the probable positions variance ellipsoid. The length of the ellipsoid components depends on the current estimates uncertainty and the predefined map error. The ellipsoids rotation $\Phi$ is given by measured vehicles angle to lane border $\psi$.*

Rotated to the UTM system with

$$R' = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.18}$$

yields to

$$\Sigma_{lane} = R' \begin{bmatrix} \sigma^2_{lon} & 0 & 0 \\ 0 & \sigma^2_{lat} & 0 \\ 0 & 0 & \sigma^2_{\Phi} \end{bmatrix} R'^{\,T}, \tag{5.19}$$

with $\phi$ as angle between the variance ellipsoids major component and the easting axis of the UTM system.
When performing lateral correction updates, the particle weights describe the probability density function for state $x_k$. The density function can be multi modal (figure 5.7), if there are multiple updates.

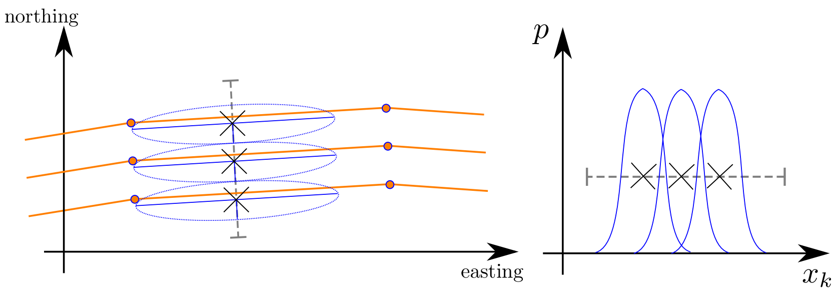


Figure 5.7.: *Multi modal density function for state $x_k$ after applying multiple correction updates. The example shows an environment with three parallel lanes and the resulting position hypothesis (X) at left image in UTM plane. The image on the right illustrates the resulting superimposed, multi-modal probability density function for states $x_k$.*

This leads to the second task. The determined ellipsoids and thus, the density functions for each probable segment must be sorted and scaled according to the recent estimated state. This can be achieved, with use of a cost function, that describes the distance between ellipse $\epsilon$ and current estimated state $\hat{x}$. Therefore, the current states position and orientation is compared to center and angle of the modeled ellipsoids. The distance part of score $s_i$ is normalized with the current standard deviation of the states first position component. It can be determined by the eigenvalues from the upper left $2 \times 2$ sub matrix of $\hat{\Sigma}$. The angle deviance is normalized by the third entry of the diagonal from $\hat{\Sigma}$, that represents the standard deviation of the states angle component.

$$s_i = \begin{bmatrix} |\hat{x}(e,n) - \epsilon_i(e,n)| \\ |\hat{x}(\Phi) - \epsilon_i(\Phi)| \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\lambda_1(\hat{\Sigma}_{2\times2})} \\ \frac{1}{\hat{\Sigma}_{3,3}} \end{bmatrix} \tag{5.20}$$

The evaluation of each ellipsoid according to the current state, yields to a sorted set of variance ellipsoids. Considering a distance-based metric, the best-fit ellipsoid is the first entry of the ascend sorted list (the lower the score/cost, the better the match). If multiple and ambiguous lane correction updates must be performed, the ellipsoids score can be used to scale the corresponding co-variance matrix

$$\Sigma_{lane}(s) = s \cdot \Sigma_{lane} \tag{5.21}$$

Using this method for the illustrated example in 5.7, the result for density $p$ is illustrated in 5.8.

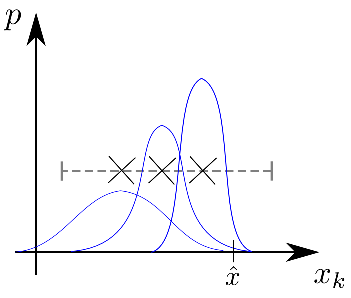


Figure 5.8.: *Multi modal probability density function for state $x_k$ after applying* scaled *multiple correction updates.*

To prevent ambiguous correction updates, the lane class can be used, if provided by the scene interpreter. The lane class is the most probable driving prescription, as result from detection, classification and interpretation. As described in chapters 4.2 and chapter 4.4, the system distinguishes five different arrow classes and thus lane classes. If class information is available, it is possible to apply additional scale to the variance ellipsoids before using them for correction update. In best case, the ambiguity can be solved and the one appropriate ellipse is used for correction. The selection is done in a straight forward manner. The lane class information from scene interpretation module for own and adjacent lanes, is verified against the available map lane information. This can be performed with kind of sliding window containing three cells. The three window cells hold the detected classes for own and adjacent lanes. If no lane class was detected, the corresponding cell stays empty. A verification for the provided classes is not made and may be wrong. Only the confidence value for detected arrow-classes is used to reject the class information. For each window shift, the matches over all lanes are counted. The result is a score for each map lane being the one, the vehicle currently drives on, according to the perceptual information of the system. The information can be used to extend the distance based weighting from (5.21) to

$$\Sigma_{lane}(s_i, h_i) = \frac{s_i}{h_i} \cdot \Sigma_{lane}, \tag{5.22}$$

with $h_i$ as hit count from arrow class verification at each step and $s_i$ the corresponding score from distance based evaluation. In case of three observed lanes, it can have one or up to three hits. Hypothesis with no hits are neglected.
The result of this procedure is a distance-class-combined enlargement or narrowing of the hypothesis variance ellipsoid, by adjusting the corresponding $\sigma$. To decide, whether to trust more distance score or classification results, the weight components can be scaled ($c_s, c_h; \in \mathbb{R}$). For this thesis, both variables have equal scale, $c_s = c_h = 1.0$.

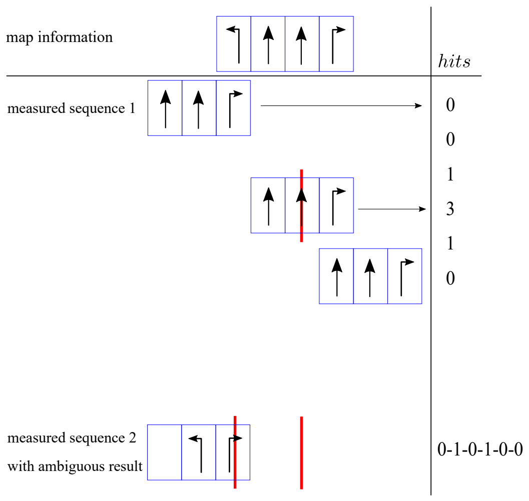


Figure 5.9.: *Evaluation of map lanes by detected lane classes as correlation test. The table header shows the information as retrieved from map. The measured sequence 1 of lane classes results into an unique assignment of the measured data to the map (most hits at one position, colored red). The sequence 2 instead, would lead to an ambiguous lane assignment. There are two lane assignments with equal probability (number of hits). The positions are colored red.*

### 5.2.4. Hypothesis tests - outlier identification and suppression

One of the crucial but hardest tasks, is the outlier identification in the measurements $z_k$, to avoid updating the particle set with wrong hypothesis and thus causing unstable estimations. Therefore, all measurement must be evaluated before correction update. This procedure is called hypothesis test. If a measurement is unlikely, it will be neglected after passing the test. Such tests are necessary, if the input data is noisy and thus, measurements can occur that are not covered by the error model. With detailed view to the position estimation modules input data, there is first the GNSS. Using the GNSS solution for periodical updates, can produce jumpy position updates. This can be caused by multipath propagation effects and varying number of satellites in sight, especially in urban surroundings, see figure 6.28 c). Second, there is the landmark detection, that is influenced by environmental conditions like lightning, weather or infrastructural disturbances (distorted markings). The strongest interference is caused by the moving camera. If driving on bumpy roads, when accelerating and braking, measurements from a static calibrated camera can be heavy interfered. The third disturbance source at estimator input is map errors. The system relies on maps as reference. What means, that map information is taken as truth. It can not be granted that all landmarks are noticed with the expected precision, not even that a detected landmark is registered at all.
All these types of erroneous inputs should be recognized and neglected by the system, before correction update is performed. To avoid, being to hard at this de-

cision and rejecting to many measurements, a compromise must be found. Simply put, seriously wrong measurements should be neglected, but the overall number of rejected measurements should be as small as possible.
An applicable method to compare the measurements (and their variances) with the current state (and its variance), is the *chi square test*. A similar way to perform the hypothesis tests, was introduced in [32]. The chi square test ($\chi^2$), provides a statistical metric for the relation between two distributions in form of a significance assessment. Therefore, the current MMSE estimate $\hat{x}_k$ (5.3) and the corresponding MSE matrix $\hat{\Sigma}_k$ (5.4) is used. Additionally, there is the measurement $z_k$ and its error covariance matrix $\Sigma_z$, that needs to be evaluated. With use of the $\chi^2$ test (5.23), now a statistical approximation $\epsilon_k$ for the innovation (goodness of fit) of $z_k$ can be determined:

$$\epsilon_k = (z_k - \hat{x}_k)^T \ (\hat{\Sigma}_k + \Sigma_z)^{-1} \ (z_k - \hat{x}_k). \tag{5.23}$$

. Assuming, measurements and particle set being normal distributed, $\epsilon_k$ is a $\chi^2$ random variable with three (GNSS, lane information) or two (landmarks) degrees of freedom. For a given number of degrees of freedom, pre-calculated tables exist, to select the appropriate threshold for $\epsilon_k$ for a desired significance level $\alpha$. Varying this value correspond to the *compromise* between neglecting wrong measurements, but keeping as much as possible, mentioned earlier in this section. A good compromise, is to adjusting the threshold to a value, that the probability for rejecting an actually correct measurement is smaller than $p = 0.05 \simeq \alpha$. This yields to a threshold for $\epsilon_k$ with ~3.84 at two degrees of freedom and ~5.99 at three degrees of freedom. Be aware, when using $\chi^2$ tables for tests with probability distributions, the degree of freedom $f$ is $n - 1$, with $n$ as the number of random variables.
For instance, a hypotheses-test with a normal distributed position guess and the two variables easting and northing, has a degree of freedom of $f = n - 1 = 1$.
Remember at this point: To evaluate the hypotheses for correction, an approximation is used with preconditions and assumptions, that might not be satisfied, e.g. normal distributed position probability. Hence, the significance evaluation for measurements within the process of landmark-based position estimation, stays a challenge. Further investigations are presented in chapter 6.

### 5.2.5. Particle re-sampling

The goal for resampling is, to concentrate most of the particles in that part of state space with the highest probability, and thus to avoid degeneration problems 3.5.
The occurrence of only few or no measurements in parts of the state space, let particles move to regions with high probability over time. Hence, this space is leaved underrepresented (*sample impoverishment*). If then, measurements propose the actual position inside this leaved area, it will take a long time, to let the MMSE estimate converge to that point.
To avoid this behavior, resamplings must be invoked, indicated by the effective

sample size $N_{Eff}$ of the particle set. At every resampling, a number of particles is spread inside areas with low state probability. Initially, only a negligible weight is applied to these particles. If measurements inside this area occur, the MMSE can converge quickly into this region, because particles are already available and weights can increase.
The implementation of the resampling procedure is described hereafter. For resampling, as *proposal distribution* the initial pdf is chosen. Therefore, at first the cumulative probability distribution $q(n),\ 1 \leq n \in \mathbb{N} \leq \mathrm{N}$ of a sorted particle set is calculated and stored in a vector. In the next step, for every new particle an equally distributed number $0 < r \in \mathbb{R} \leq 1$ is generated. The corresponding particle of the prior pdf has now the index $n$ of the vector, where $q(n)$ exceeds $r$. On this particle, a noise $e$ according to $e \sim N(0, \Sigma_c)$ is applied. To sample from this normal distribution, the method introduced in A.5 is used. At the next step, three different subsets of particles are sampled, with varied error co-variance and with varied contribution to the overall weight of the particle set. Within a subset, all weights are equal. The error co-variances and subsets fulfill the following requirements:

$$\Sigma_c = \begin{bmatrix} \sigma_{c.1}^2 & 0 & 0 \\ 0 & \sigma_{c.2}^2 & 0 \\ 0 & 0 & \sigma_{c.3}^2 \end{bmatrix},\ 1 \leq c \in \mathbb{N} \leq 3 \tag{5.24}$$

- subset 1: 0.75 of all particles with small noise $\sigma_{1.1} = 0.5m$, $\sigma_{1.2} = 0.5m$ and $\sigma_{1.3} = 2°$ and 0.99 of total weight
- subset 2: 0.2 of all particles with medium noise $\sigma_{2.1} = 2.5m$, $\sigma_{2.2} = 2.5m$ and $\sigma_{2.3} = 10°$ and 0.0099 of total weight
- subset 3: 0.05 of all particles with $\sigma_{3.1} = 25.0m$, $\sigma_{3.2} = 25.0m$ and $\sigma_{3.3} = 90°$ and 0.0001 of total weight.

With this method, the modeled noise on every fourth of all sampled particles leads to a larger deviation of the state, as if it would if generated only from prior pdf. Consequently, particles will spread into less probable regions of the state space. With their small weight, they will not influence the estimation negatively. But if measurements occur in this region, their impact on the estimation will increase quickly. This method yields to robust and stable estimations, but without being slow at changing conditions.

# 6. Evaluation

In this chapter, the system components for *landmark retrieval* and *position estimation* are evaluated. In the first section 6.1, the results for the implemented computer vision algorithms are discussed. Additionally, the results from feature selection is presented.
To evaluate the classifier performances, data sets were created to evaluate and compare the marking classification based on geometrical features and the HOG approach. Furthermore, the classifier performance results and a summary is presented.
The second section is split into lane level position measurement 6.1.6 and 6.2. In the first subsection - as an essential input for the pose estimation - the performance of the lane measurement is shown. It is presented, that the use of a camera set (in comparison to single sensor), improves the availability of lane information.
In the second subsection, the global pose estimation is evaluated. The adjustments, needed to make the estimation capable for the special characteristics of urban surroundings, are introduced. Furthermore, it is shown how lane measurements can be used to perform single corrections and how to treat multiple options for correction. The creation of multi-modal density functions with particle weights, is presented in detail. The impacts of this multi-modal approach on resampling, lane assessment and mapping, error modeling and finally pose estimation, is presented. Additionally, the creation process for ground truth data is explained and the test tracks are introduced in detail.

## 6.1. Landmark retrieval and classification

### 6.1.1. Evaluation of marking extraction

The detection performance is essential for the geometrical approach to classify road markings. If no contour is extracted, no hypothesis exists and no classification can be performed. Extracted contours, representing non-marking parts of the image, can lead to wrong classification output. For quantitative evaluation of the detection and classification part, the data must be categorized (labeling). For the extraction step, this can be a pixel-wise categorization into *marking* and *negative*. This process is very slow and causes - typically non automatized - efforts. Nevertheless, the significance is small, because the detection part is implicitly evaluated with separation in residual data and contour classes later in this section. There is a subset that stays unknown. It contains markings, which are not extracted (*false negatives*) by the system, e.g. faded or blinded road markings.

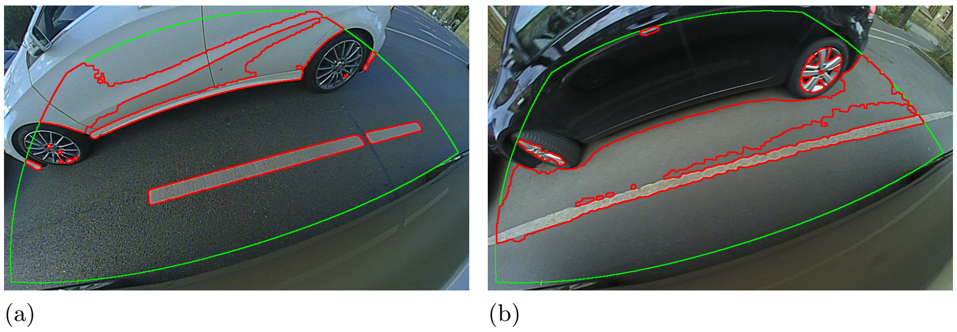


Figure 6.1.: *Extraction, influenced by objects in field of view. The objects inside the detection area violating the basic assumption, that only street surface and markings occur inside this area. The local average intensity value for contour threshold is disturbed by the two cars. (green: observed area for detection, red: extracted contour)*

For direct image features like HOG, it is possible to evaluate the detection performance with hypotheses matched to residual data, but actually containing markings (false negatives). There is no dedicated *extraction* step, but a feature calculation step for the entire image with a subsequent scan process.
For contour extraction, the region based method shows best results. The method is robust to different types and disturbed markings. Small cracks and gaps on the marking outline can be closed with morphological operations. For markings, separately applied to the road surface, this ensures a continuous border around all of its parts. In A.6 illustrations are presented to show results.
Surroundings, where markings not differentiate from street surface stay challenging, e.g. non-asphalt surfaces, side walks, roadside structures and other traffic participants. The assumption, that white markings can be differentiated explicitly from darker surface is violated in this case and false detection occur. This situation is shown in figures 6.1a and 6.1b, where light and dark colored vehicles influencing the local average intensity in different ways. To solve this issue, the detection area should be generated adaptive to the vehicles surrounding. This can be achieved with the use of an upstream object recognition. The expanded detection area of the windshield camera, causes increased probability for false detections, compared to fisheye cameras, observing the near field. There are generally more disturbing objects in a wider field of view. Because, the same processing chain is applied to every camera for system scale reasons, different parameters must be used, especially for image pre-filtering. Additionally, the automotive fisheye cameras images have compression artifacts. For the front camera (windshield) a $3 \times 3$ median filter is used and a $7 \times 7$ median filter for the fisheye cameras.

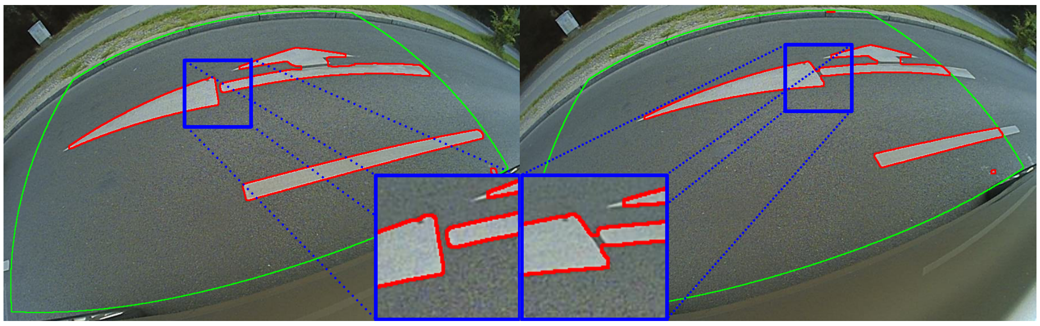


Figure 6.2.: *Impact of ground resolution. Along with the strong varying ground resolution for different distances to the fisheye camera, the same morphological operations yield to different results. (green: observed area, red: extracted contours)*

### 6.1.2. Contour data set and feature selection

The feature selection is based on labeled contours extracted from recorded scenes in urban environment. Table 6.1 shows the content of the contours data set and the number of samples for each class. To expand the data set for arrow class types, the classes *left* and *right* arrow, as well as *straight-left* and *straight-right* were mirrored axis-symmetric on $y$ and additionally registered for the opposite class type. Subsequently, the data set was divided into two parts, for training and for validation. The training data set is used for feature selection and classifier training and the validation data set to evaluate the classifiers performance. Because contours in the validation data set are unknown to the classifier, the evaluation with them gives a good measurement for generalization capabilities.

Table 6.1.: *The contours data set.*

| class | count | mirrored |
|---|---|---|
| *residual data* | 11233 | - |
| *line segment* | 9079 | - |
| *straight arrow* | 1601 | - |
| *left arrow* | 532 | 864 |
| *right arrow* | 332 | 864 |
| *straight-left arrow* | 133 | 839 |
| *straight-right arrow* | 706 | 839 |
| $\Sigma$ | 23616 | 25319 |

Before starting the selection procedure, all features must be normalized as described in section 4.2.1. This must be done over the full training set. Goal of the following feature selection, is to select a small descriptive set of features out of all feasible features. This should result in low processing time, but robust performance for classification. The *Knock-Out-Method* is not suitable to process all features at once, because of correlations between features and thus the fail for determine the trace-2 criterion. Hence, the following feature groups are combined and checked for descriptiveness.

- geometrical features
- Fourier Descriptors
- Hu-Moments
- normalized, centralized moments

The results are shown in table 6.2.

Table 6.2.: *Results from feature selection.*

| feature group | result from 10 selected features |
|---|---|
| Fourier Descriptor + norm. centr. moments | Fourier descriptors remain |
| Fourier Descriptor + Hu-Moments | Fourier descriptors remain |
| Fourier Descriptor + geom. features | Fourier descriptors remain |
| norm. centr. moments + Hu-Moments | not used, because HU-Moments can be derived from norm. centr. moments, and thus being correlated |
| norm. centr. moments + geometrical features | feature mix, $J_2 \approx 5,5$ |
| Hu-Moments + geometrical features | feature mix, $J_2 \approx 4,4$ |
| Fourier Descriptor | $J_2 \approx 10$ |

The dominating group in this set is Fourier coefficients. It is - under both methods - the strongest feature group for the separation task. In figure 6.3 the separability value is shown over the number of coefficients. A saturation can be determined around the use of ten coefficients. A lager set will not lead to a substantially better separation performance.

The selected features were tested with the validation data set, using a SVM with a radial basis function kernel and default parameters for $\gamma$ and $C$. Contrary to the expectation, the result was an overall class performance (true positives) of 70%. It can be shown, that the additional use of geometrical features and moments, increase the classification precision above 80%. The following features were used for second test run:

- normalized, centralized moments: $\eta_{11}, \eta_{02}, \eta_{30}$

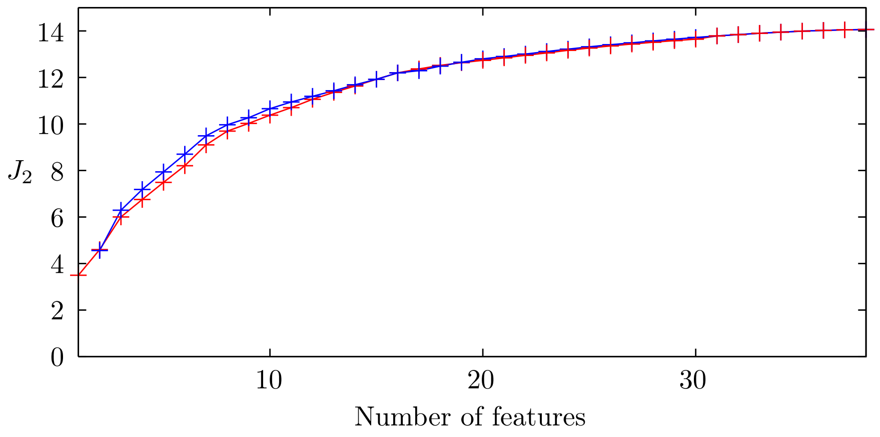


Figure 6.3.: *Increase of separability value $J_2$ over number of Fourier descriptors. For definition of $J_2$ see 4.11. The value for* Add-On-Method *(red) and* Knock-Out-Method *(blue) only differ for less then 15 features.*

- major to minor axis ratio $R_{ab}$
- enclosing rectangle to contour area ratio $R_{RA}$
- convex hull to area ratio $R_{HA}$
- gravity balance $\beta$

For the *Add-On-Method* $\eta_{12}$ and $\eta_{03}$ were additionally selected. For the *Knock-Out-Method* $\eta_{21}$ and roundness $R$ were selected. The results for pure Fourier descriptor based classification and the classification based on additional features, is shown in table 6.3. The reason for better classification results, despite a smaller theoretic separability value, is that $J_2$ is a pure statistical value of the feature set. It is an indication, but not an absolute degree for the expectable classifier performance. Parameter tuning and class weight adaption showed, that with chosen feature set, the class *line segments* has high inter-covariance with the class for residual data. This is not surprising, because of the simple and therefore less descriptive shape of a line segment. The desired true positive rate is a trade-off and one has to decide, whether to perform better on residual data and thus cause less false positives, or to reach higher true positive rate on the line segment class. That gives the opportunity, to perform a dedicated *'traditional'* lane border detection using scan line approach, and to focus on precise detection and classification of arrow markings.

### 6.1.3. Classification with geometrical features

Classification of contours was investigated for SVM and random forest classifiers. Both designed as multi-class and single-class detectors. See 6.4 for the training set

Table 6.3.: *Classification results for Fourier coefficient and mixed feature sets.*

| | class recall | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Fourier desc. | 0.70 | 0.83 | 0.59 | 0.60 | 0.71 | 0.78 | 0.77 |
| mixed features | 0.72 | 0.81 | 0.96 | 0.86 | 0.86 | 0.93 | 0.91 |

and its containing contours for classification by geometrical features and results afterwards. Optimal classifier parameters were selected by a grid search process.

| class | sample count |
|---|---|
| 0: residual | 8417 |
| 1: line segment | 6808 |
| 2: straight arrow | 1198 |
| 3: left arrow | 647 |
| 4: right arrow | 647 |
| 5: straight-left arrow | 629 |
| 6: straight-right arrow | 629 |

Table 6.4.: *Training data set for geometrical classification.*

In the following tables, the results of the geometrical approach is presented. For both type of classifiers,

- the confusion matrix for all classes (6.5),
- the single class performance (6.6),
- followed by a figure (6.4) that illustrates the increasing detection rate on expanded training set sizes (learning curve) is presented.

In the confusion matrix, each row represents the actual class (ground truth) and each column the predicted class. Thus, the diagonal represents the correct matching rate (true positive rate) and all other values the misclassification rates. For example, see table 6.5, row 3 (results for *left arrow*). 86% of all *left arrow* samples were classified correctly. 7% of them were classified as class 4 (right arrow) and 6% as residual data (false negative). The single class detection performance was evaluated exclusively for arrow classes (class 2 – 6). The classifier was trained in a way, that all inputs except the selected class were treated as residual data. The FP-rate (false positive) indicates the rate for the misclassification. For example, in table 6.6 the results for class 2 show a false positive rate of 0.007. That means that 0.7% of all *non-class-2-samples* were predicted incorrect as a class-2 sample. For random tree based classification, the detector was optimized regarding the overall class performance and an acceptable false positive rate. The same training and evaluation sets as for SVM evaluation were used.

Table 6.5.: *Confusion matrix for multi-class classifier based on C-SVM (Radial Base Function (RBF)) with full training set in geometrical classification.*

| | | prediction | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | rates | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| ground truth | 0 | **0.72** | 0.24 | 0.01 | 0.01 | 0.01 | 0.01 | 0.00 |
| | 1 | 0.18 | **0.81** | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 |
| | 2 | 0.03 | 0.01 | **0.96** | 0.00 | 0.00 | 0.00 | 0.00 |
| | 3 | 0.06 | 0.00 | 0.00 | **0.86** | 0.07 | 0.01 | 0.00 |
| | 4 | 0.11 | 0.00 | 0.00 | 0.02 | **0.86** | 0.01 | 0.00 |
| | 5 | 0.03 | 0.00 | 0.00 | 0.00 | 0.00 | **0.93** | 0.03 |
| | 6 | 0.04 | 0.00 | 0.00 | 0.00 | 0.00 | 0.04 | **0.91** |

Table 6.6.: *Single class detector performance (C-SVM) on arrow classes [2,...,6] with geometrical features and radial base function kernel.*

| class | TP-rate | FP-rate |
|---|---|---|
| 2 | 0.95 | 0.007 |
| 3 | 0.85 | 0.004 |
| 4 | 0.84 | 0.002 |
| 5 | 0.96 | 0.004 |
| 6 | 0.92 | 0.002 |

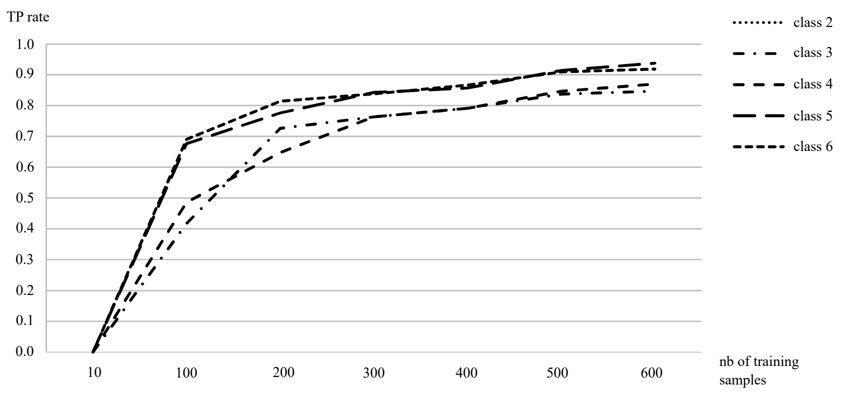


Figure 6.4.: *TP-rates for varying training set sizes on geometrical features (SVM)*

### 6.1.4. Classification with HOG descriptors

This section shows the results of the HOG based classification process using SVM and random forest classifiers. The data set and the meaning of the class labels is introduced in table 6.9. The result tables (table 6.10 and 6.11) are presented in a similar way, as introduced in section 6.1.3.

Table 6.7.: *Confusion matrix for multi-class classifier based on random forest and geometrical features.*

| | | prediction | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | rates | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| ground truth | 0 | **0.88** | 0.09 | 0.01 | 0.01 | 0.00 | 0.00 | 0.00 |
| | 1 | 0.10 | **0.90** | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 |
| | 2 | 0.03 | 0.02 | **0.95** | 0.00 | 0.01 | 0.00 | 0.00 |
| | 3 | 0.03 | 0.00 | 0.00 | **0.95** | 0.06 | 0.00 | 0.00 |
| | 4 | 0.08 | 0.00 | 0.00 | 0.02 | **0.92** | 0.02 | 0.00 |
| | 5 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | **0.98** | 0.01 |
| | 6 | 0.02 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | **0.97** |

Table 6.8.: *Single class detector performance (random forest) with geometrical features. class TP-rate FP-rate*

| class | TP-rate | FP-rate |
|---|---|---|
| 2 | 0.95 | 0.003 |
| 3 | 0.91 | 0.002 |
| 4 | 0.92 | 0.004 |
| 5 | 0.95 | 0.003 |
| 6 | 0.94 | 0.005 |

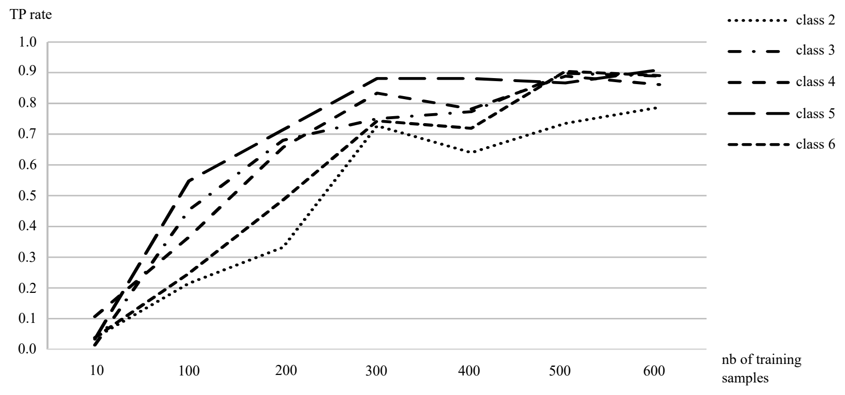


Figure 6.5.: *TP-rates for varying training set sizes on geometrical features (random forest)*

### 6.1.5. Discussions on results

#### Detection performance vs processing time

The computation time is very depending on the size of the ground area that is used for detection. The wider the area is selected, the more contours come into detection range and the more computations must be done. Therefore, this section shows the results for feature calculation and prediction of one contour. Separately,

| class | sample count |
|---|---|
| 0: residual (includes line segments, class 1 | 1912 |
| 2: straight arrow | 106 |
| 3: left arrow | 35 |
| 4: right arrow | 35 |
| 5: straight-left arrow | 27 |
| 6: straight-right arrow | 27 |

Table 6.9.: *Data set for HOG feature training.*

Table 6.10.: *Confusion matrix for multi-class classifier based on C-SVM (linear kernel) andHOG features.*

| | | prediction | | | | | |
|---|---|---|---|---|---|---|---|
| | rates | 0 | 2 | 3 | 4 | 5 | 6 |
| ground truth | 0 | **0.99** | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 |
| | 2 | 0.15 | **0.85** | 0.00 | 0.00 | 0.00 | 0.00 |
| | 3 | 0.12 | 0.11 | **0.77** | 0.00 | 0.00 | 0.00 |
| | 4 | 0.18 | 0.00 | 0.00 | **0.82** | 0.00 | 0.00 |
| | 5 | 0.01 | 0.00 | 0.00 | 0.00 | **0.99** | 0.00 |
| | 6 | 0.07 | 0.00 | 0.00 | 0.00 | 0.00 | **0.93** |

Table 6.11.: *Single class rates (SVM, linear kernel) with HOG features class TP-rate FP-rate*

| class | TP-rate | FP-rate |
|---|---|---|
| 2 | 0.82 | 0.003 |
| 3 | 0.77 | 0.000 |
| 4 | 0.82 | 0.000 |
| 5 | 0.93 | 0.000 |
| 6 | 0.92 | 0.000 |

Table 6.12.: *Confusion matrix for multi-class classifier (random forest) with HOG features.*

| | | prediction | | | | | |
|---|---|---|---|---|---|---|---|
| | rates | 0 | 2 | 3 | 4 | 5 | 6 |
| ground truth | 0 | **0.91** | 0.06 | 0.02 | 0.00 | 0.00 | 0.00 |
| | 2 | 0.09 | **0.87** | 0.02 | 0.02 | 0.00 | 0.00 |
| | 3 | 0.12 | 0.00 | **0.82** | 0.06 | 0.00 | 0.00 |
| | 4 | 0.06 | 0.12 | 0.06 | **0.76** | 0.00 | 0.00 |
| | 5 | 0.21 | 0.00 | 0.00 | 0.00 | **0.79** | 0.00 |
| | 6 | 0.00 | 0.14 | 0.00 | 0.00 | 0.00 | **0.86** |

the results for a whole classification process on an input image, from first computer vision component up to the classified object list output is presented. The second is a more relevant value for practical use, 6.13. The results are separated for each classifier type and feature combination. Test machine was a multi core processor with one activated core at 2.90 GHz.

Table 6.13.: *Evaluation results arrow-classes. FP means non-arrow sample (negative) miss-classified as arrow.*

| Method | Geometric Model | | HOG | |
|---|---|---|---|---|
| Classifier | SVM RBF | RTree | SVM LINEAR | RTree |
| Avg recall rate | 0.92 | 0.95 | 0.87 | 0.82 |
| FP rate | 0.04 | 0.03 | 0.01 | 0.08 |
| Avg feature calculation time [ms] | 0.198 | | 0.336 | |
| Avg prediction time [ms] | 0.08 | 0.01 | 0.03 | 0.01 |
| number of calculations per frame | 19.02 (avg) | | 180 (fix) | |
| Avg processing time (complete process chain) [ms] | 16.4 | 15.2 | 47.1 | 45.4 |

It is obvious, that the main difference between the two approaches is neither the rare classification result, nor the calculation time for one sample to classify. It is more a question of how often the methods need to run to get an understanding for the whole input image. The segmentation step of the geometrical approach lead to a small number of samples to classify ($\sim 20$) and therefore to a fast processing, to complete the work on an input image. One big advantage of geometrical analysis, is the complete scaling and rotation in-variance. That makes it possible to natively detect other rotated or combined marking types like stop-lines or pedestrian crossings. The HOG based method shows very good results, even when using small training sets ( 6.9). Using the SVM, it is the best way to avoid false positives.

A low false positive rate has tremendous importance to make algorithms available for real world applications, while slightly decreased true positive rates can be handled with tracking methods. The processing time of HOG for one sample is nearly the same to the contour based approach. However, when processing the whole image – depending on the chosen ground patch size and the sliding window configuration – a much increased number of calculations is needed to get sufficient and dense detection results as precondition to understand the scene.

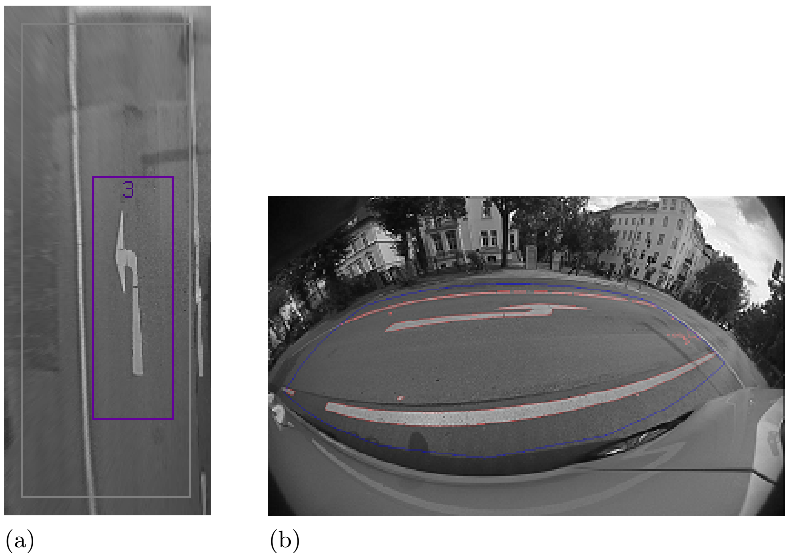


Figure 6.6.: *Images from the evaluation. (a) The IPM image from left mirror camera. The size of the ground patch is* $15 \times 6.15$ *m and the resolution is 20 px/m. The resulting IPM image size is thus* $300 \times 123$ *px. The classifier result for the image sub-patch including the arrow is shown (purple colored window, class 3 for left arrow). (b) View of the left mirror camera. The result from contour finding process is visualized. Contour edges marked red.*

### Robustness of the detection process

One goal of every sensor based detection system is high robustness against disorders. Especially on road markings the disturbances mentioned in section 4.1.1 in this work become relevant. Because a set of cameras was used for this work, most of occlusion problems can be handled. Beside the occluded area on one camera, there is often enough free space around the car to get lane information from the other cameras of the set. At least while passing, the markings are visible to the system. The influence of shadows can be minimized with adaptive threshold techniques. Nevertheless, it is a system boundary especially when using the geometrical approach for detection. The HOG based method however, shows more robustness on shadows and avoids the problem of finding good thresholds.
Distorted markings can be dealed with morphological operations after thresholding and before contour finding. It is important that the distorted marking still have its characterizing shape. Heavy distortions will result to miss-classifications.

## 6.1.6. Landmark availability and classification of driving prescriptions

The recorded scene for evaluation is about 19.5 seconds and causes 589 processing cycles. The drive maneuver starts at the left of three lanes and changes to the middle lane. The speed is $\approx 50$ km/h. The left lane border is defined by a center strip made of cobblestones. A marked lane border is not present. At the end of the scene, traffic appears without any influence to the system (false detection).

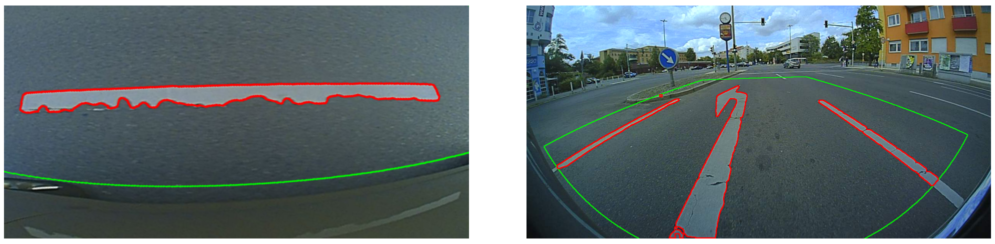
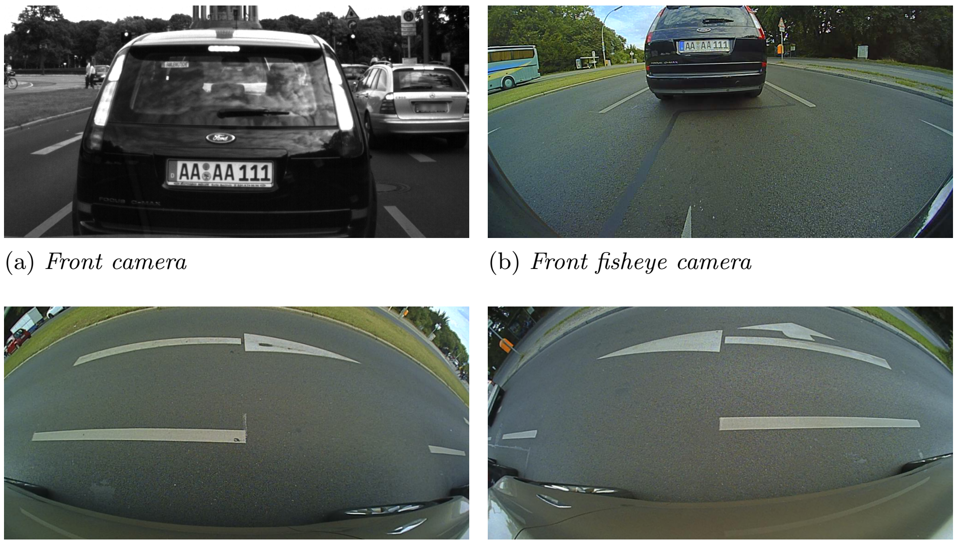

(a) (b)

Figure 6.7.: *Examples for heavy (a) and light (b) distorted markings.*

The scene was investigated with three camera configurations. The first with front camera only, second with fisheye cameras and the third with complete set. Figure 6.10 shows the measured border deviation and lane width for all tests. In all



(a) *Front camera* (b) *Front fisheye camera*

(c) *Left fisheye camera* (d) *Right fisheye camera*

Figure 6.8.: *Comparison of availability for lane information in urban surroundings, using different types of camera on varying mounting positions.*

cases, if lane borders are available, the resulting lane width is $b \approx 3$m. This is congruent to the expected lane width. Considering the plots, the measurements from the fisheye camera have lower oscillation and are more stable than from front camera. The disadvantage of the front camera is caused by lower image resolution, but particularly from the larger distance to measured lane borders. The scene interpreter-module sets the measurement to a valid state after five effective detection cycles. This is granted for both camera settings. At lane change, the delay time for front camera is five cycles. The fisheye cameras lose the right lane border after 2.9 s, as lane border moves underneath the car. The front fisheye camera does not contribute enough measurements for a sufficient detection and tracking in that case. Passing additional 1.3 s, the left and right lane border from the new lane (middle) are available. Using the complete set of cameras, yields in higher availability with more stable measurements. Figure 6.10c shows a smooth plot with increased availability. As mentioned, the traffic in this scene was low. With dense traffic or even jams, the availability for the front camera can be substantially limited, see figure 6.8a. In this case, the fisheye cameras contribute the essential information for a valid and stable output of lane information, see figure 6.8b to 6.8d). The evaluation of class availability for arrow markings was

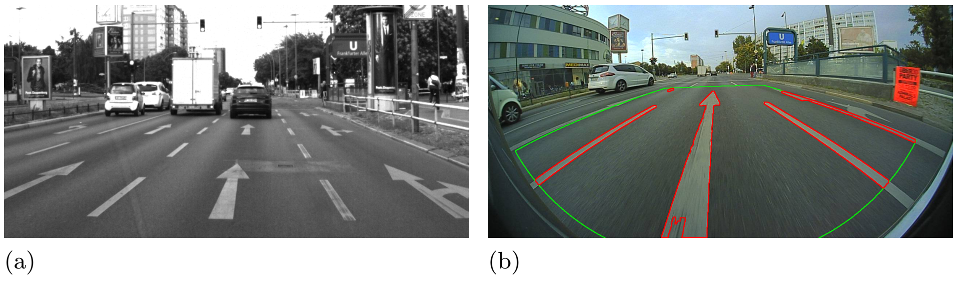
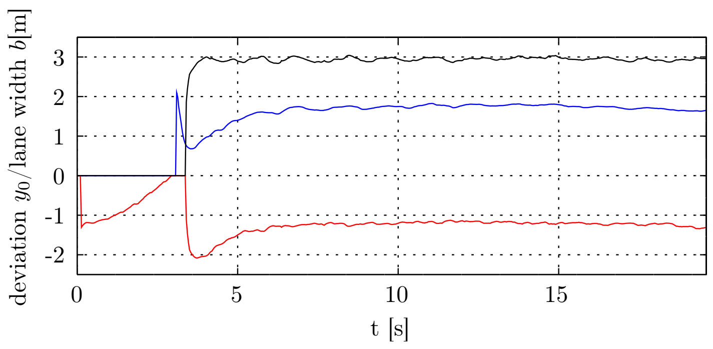
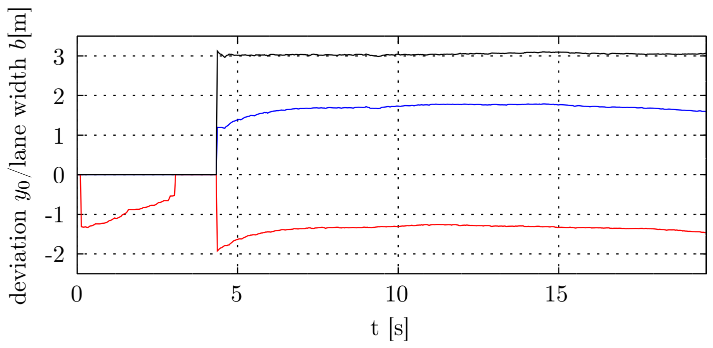

(a) (b)

Figure 6.9.: *Evaluation of lane type classification. (a) Shows a snapshot from one of the the used scene for evaluation. (b) Shows a false detection on right adjacent lane. The arrow is not fully covered by the observation area (green), what leads to a miss-classification*

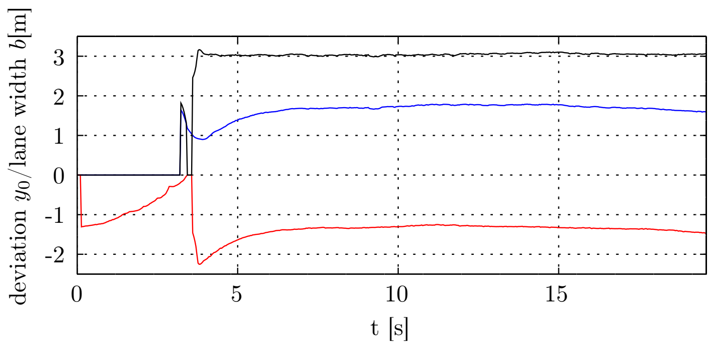
performed on a recording with the following specification. The test vehicle approaches a multiple-lane crossing. The velocity is about 50 km/h. At left adjacent and current lane, there are arrows of class *straight*. At the right adjacent lane the markings are from type *straight+right*. The evaluation was performed with same camera setting as from previous section. Figure 6.11 shows the results from lane type classification with the use of complete camera set. The recognized lane types are plotted for every lane. Beginning at time $t = 2s$, the front cam is able to detect the markings. Due to traffic on currently driven lane (ego-lane), the detection is slightly delayed. The classification result for the left lane is continuous and correct. Up to the point, as the last arrow leaves the covered area of the front camera and still no arrow is in sight of the left mirror-camera (blind spot). This results in a briefly interrupted typification for the left adjacent lane. For ego-lane, the classification is continuous and correct, because the covered area of front camera and front fisheye camera overlap. For right adjacent lane, the result is somewhat different. There are temporally interrupted and even false classifications. With a detailed view (figure 6.9b), a miss-classification on the front fisheye camera is shown. At the right border, the visible arrow marking is truncated and compares to a halved straight arrow. The classification module provides class *straight* for it. In appendix A.7 the results for other camera sets are presented. It is shown that with both, front camera and fisheye cameras, it is possible to detect and decide the lane type. However, the combination of all cameras gives the best availability



(a) *Using front camera (windshield)*



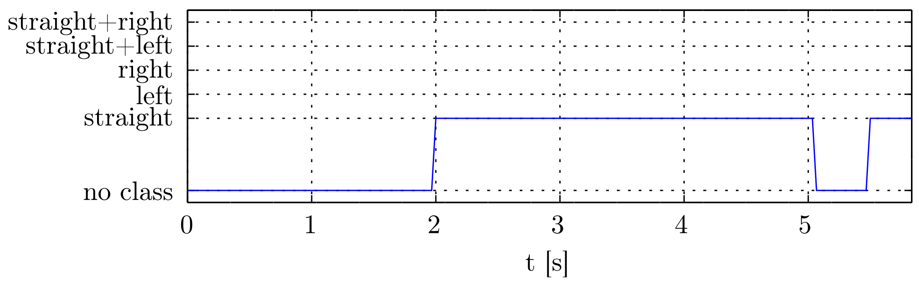
(b) *Using fisheye cameras*



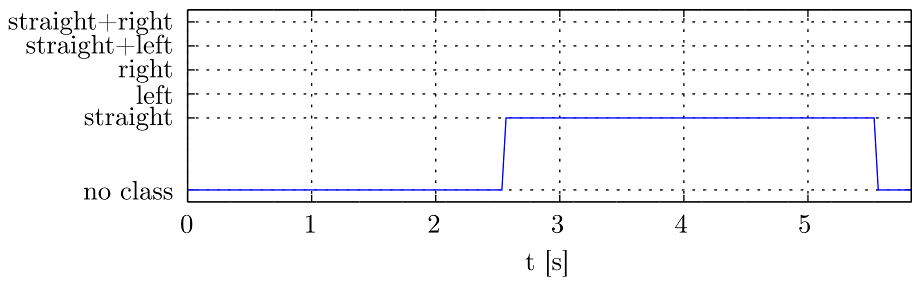
(c) *Using front camera and fisheye camera*

Figure 6.10.: *Deviation (left: blue, right: red) and lane width (black): The lane measurement of the fisheye camera leads to more stable output with less oscillation. Compared to that, the measurements from the front camera are available earlier. A combination to an integrated set of camera sensors, leads to boosted detection accuracy and availability.*

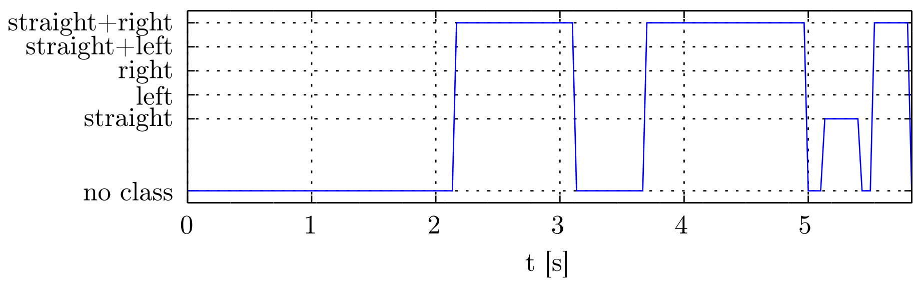
and most stable result for lane class information.



(a) *Left adjacent lane*



(b) *Ego-lane*



(c) *Right adjacent lane*

Figure 6.11.: *Evaluation of lane classification, when using the full camera set. The diagrams show the class availability for each lane over time.*

## 6.2. Preliminary remarks to the evaluation of position estimation

The position estimation - as its name implies - is the attempt to make a (positions) guess, that meets reality as far as possible. Therefore, the estimation process is a conglomeration of several assumptions and concepts for combining multiple, often inconsistent, information to a stable and reliable result. For this thesis, as Bayes filter a particle filter is used, where the filter weights give a quantized shape of the positions probability density function. The particle filter is influenced by multiple inputs and must aggregate them to a single output. The main definitions that must be set for the interaction of updates and corrections are:

- Quantization of the estimators state space and the time domain,
- Trigger-Strategy for incoming events (detections),
- Concept for event assessment (e.g. lane assignment, outlier identification),
- Quantification of the events error margin,
- Concept to manage the particle set, in terms of being an optimal filter to represent the process of consecutive position guesses.

These main definitions will be investigated in the current section and the influence of different concepts and adjustments is shown.
Before going into details, there is an important preliminary note: All evaluations were performed with GNSS signal switched off after the initial positions guess. The position estimation is solely based on landmarks, detected by the image vision module and the odometry derived from cars power-train CAN messages according to 3.4. The evaluation was performed with two recordings, representing a wide spectrum of possible environment and traffic situations. Evaluation of different weather and lightning conditions was excluded. It is more a sensor specific issue, that evince the limitations of a sensor set and therefore the quality and availability of information. Discussions on boundaries of the detection module - hence for input of landmarks, signals and odometry - is presented in chapter 7.
The first scene was recorded in Berlin and proceeds from Königin-Louise-Platz via Grunewaldstraße to Schlossstraße. The data is from summer 2017. It is day and the weather is dry and clear. The sun produces cast shadows from trees and buildings on the road surface. In some cases the surface is shiny or overexposed so that markings disappear. The constructions are medium dense, but continuous like in a mid- or small-town. It becomes more dense at the end of the scene, having a more city-like characteristic. The vegetation is dense, with mainly a closed line of treetops. The road is hilly and bumpy and mostly the outer lane border is missing. There are only few detectable stop lines. At the end of the scene, the vehicle approaches a big crossing with multiple lanes and detectable arrow markings. The last maneuver is a left turn on that crossing. The length

of the driven path is around 1,2 km. In table 6.14, the provided input signals are presented. The following figures illustrate the content of the used test-scene.

| Signal | Frames/Packets per second | Specification |
|---|---|---|
| fisheye cameras | 33 | 1280x800 color image |
| front camera | 25 | 752x480 grayscale image |
| CAN | 50 | odometry relevant CAN message cycle |
| GPS | 1 | unfiltered solution from acGNSS receiver |

Table 6.14.: *Recorded signals for evaluation and test.*

The second test sequence was recorded in Berlin on the way from Tegeler Weg to freeway A111. Thereby, a roundabout with multiple lanes (up to four) and driving prescriptions must be passed at Jakob-Kaiser-Platz. The weather was clear and the sun produces several cast shadows on the road surface. On the way from roundabout to freeway (Kurt-Schumacher-Damm), the road surface appears shiny and no marking can be detected for several seconds ($\approx 10$). Next to this, a more detailed perspective from mirror cameras shows, that the markings are demolished at the right and soiled up to complete occlusion on the left side of the car. The construction in this scene is not dense, but special. There is a wide bridge at the beginning and later in that scene, a curved tunnel ($l = 967m$) beneath the Tegel airport, including a turn maneuver. A second tunnel 'Ortskern Tegel' ($l = 755m$) is passed later in that scene.
Leaving the roundabout, no further arrow markings appear, but a stop line at the last crossing before freeway. At the end of the scene (after the tunnel) the vehicle is stopped by a traffic jam. There are high walls for noise prevention next to the freeway, which cause demonstrably multipath errors on the GPS-solution. Generally, there a a few junctions and exits into and from the freeway in this recording.

(a)

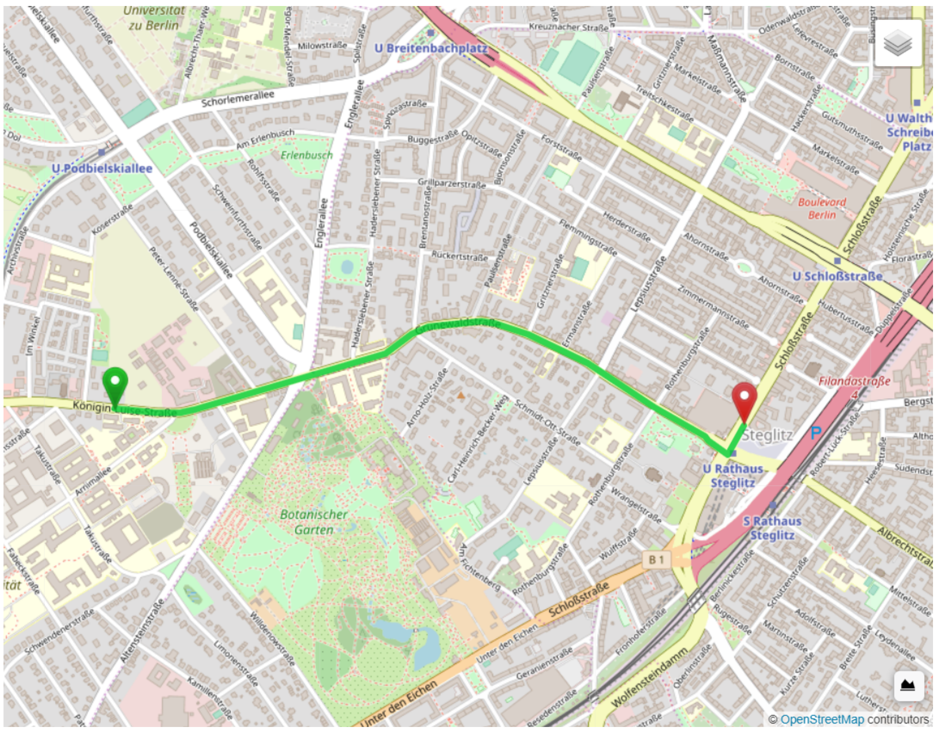
(b)

(c)

(d)

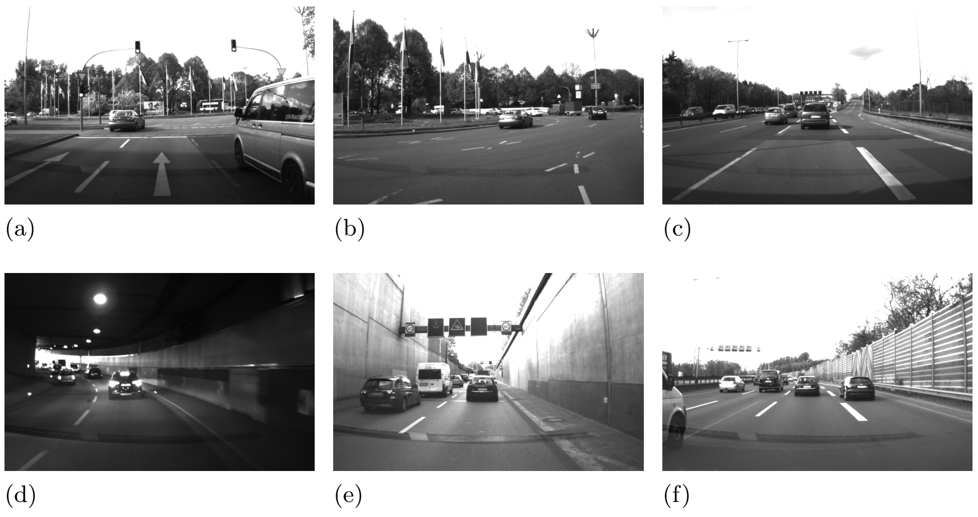
(e)

(f)

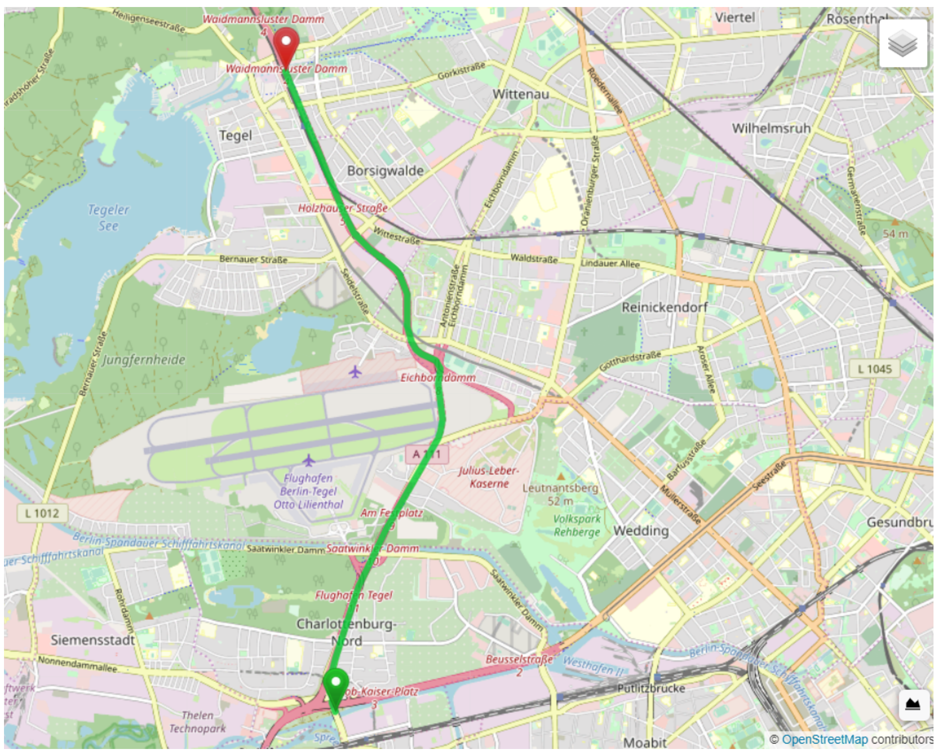


(g)

Figure 6.12.: *Impressions and map from the first test track. [35]*


(a) (b) (c)

(d) (e) (f)



(g)

Figure 6.13.: *Impressions and map of the second test track. [35]*

## 6.3. Strategies on updates and corrections in spacial, time and semantic domain

The questions to answer in this section are: How often an update/prediction should be applied to the particle set. In which way corrections should be done. And finally: What are the special conditions that must be considered? At first, the trigger strategy is presented. The following trigger modes are conceivable

- event based
- timer based
- and distance based.

The event based trigger, effects as an immediate interrupt to the current state. Each time, the detection module outputs a landmark, it will be used for correction immediately. This leads to a permanently changing state of the set. The state changes depend on the cadence of the detection-modules output. For example: When having a 50 Hz output of current stop line measurements, the particle weights are adjusted with the same cycle. The resulting weight from correction step must be determined for each particle, what causes an immense computation effort if using a large set of particles. Furthermore, there is the effect of a very fast particle weight concentration. This can be moderated with re-sampling. This yields in a permanent alternation between correction and re-sampling and leads to a very jumpy, less stable output. Measurement errors or jumps at detection side interfere directly the estimated pose. Adding space between updates, leads

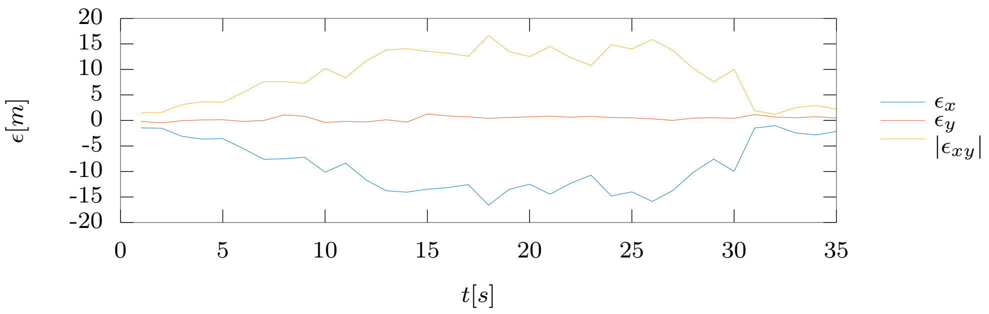


Figure 6.14.: *Event triggered correction results.*

to a smoother output and smaller error. Because the particle set represents more history now, instead of permanent re-sampling. With less correction updates, the prediction step gains more importance. Since the odometry is more precise to determine than landmark positions, it should have more influence to the estimation process. The added space between updates can be in time domain and/or spacial. A time based correction trigger has the advantage, that it continuously causes corrections, where the distance based trigger depends on the cars speed. Thus, the time based correction should be preferred, especially for the lateral correction with lanes. For sparse information like stop lines, no difference between the methods could be determined. In figures 6.14 to 6.16 the errors between reference path and test path is shown. The test starts with the first erroneous GNSS pose for initialization, driving along a left curve to a traffic sign with stop line. From $t = 20s$ to $t = 30s$, the vehicle is waiting and in standstill. As the stop line is visible to the detection system, the correction is performed. Thus, there is a smaller error from $t = 31s$. The internal filter state is different at each performed evaluation test, because the corrections were caused by different measurements and the interaction between correction and re-sampling differs. This yields in different correction points, as the first accepted measurement of the stop line occurs.

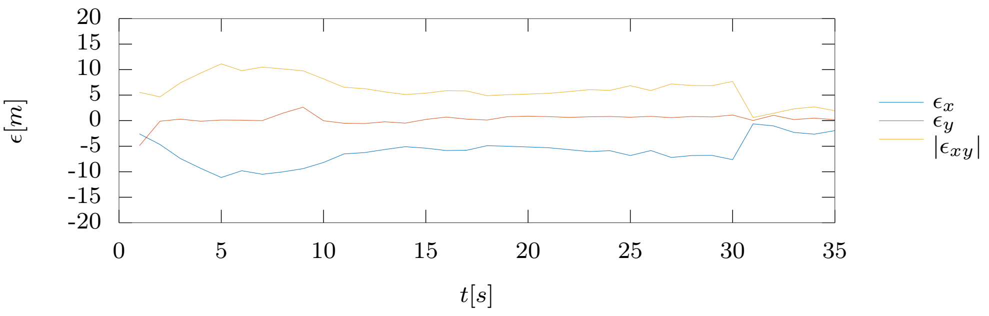


Figure 6.15.: *Timer triggered correction results. Update trigger was at 2 Hz (500 ms)*

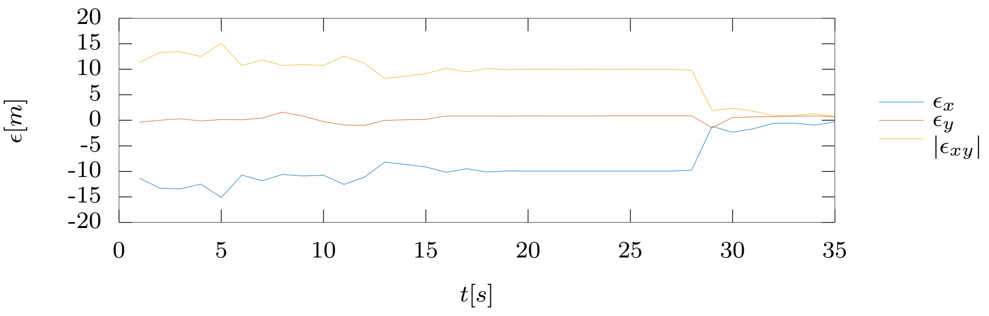


Figure 6.16.: *Distance triggered correction results. Update trigger was every 10 m*

The prediction step is to pursue the current estimated position with odometry. The prediction and all correction cycles work asynchronous. In terms of vehicle positioning and thus for this thesis, it is by far the portion with the biggest influence to the estimation result. In the prediction step, the particle set is moved, along with a modeled movement error, according to the odometry. Therefore, a translation vector with movement and yaw deltas (5.5) is applied to all particles.

The process is time triggered. It is essential, that additional motion prediction

steps are applied as precondition for every correction. Otherwise, the particle set and the state it represents, becomes out of date and the correction is applied to obsolete data (see synchronizing topics in chapter 4.3). Another opportunity to guarantee recent data, is to take care of a sufficient high cadence of prediction steps. This comes with a highly increased amount of additional computations, because motion deltas must be applied to all particles separately.
With the predicted motion, representing a movement along map lane data, emerged that modeled map data, needs 3D-information or better an additional attribute, describing the lever of the segment. In hilly environment, there is an effect if applying the vehicle motion data (determined from wheel movement) to flat map data, causing noticeable errors. Simply put: The traveled path on surface could be substantially longer than map-data implies. See figures 6.17 and 6.18 for profiles of the test tracks. The effect causes an increasing longitudinal error when driving. With sparse landmark information for longitudinal corrections, the measurement to map allocation can fail the $\chi^2$ test in this case. Correct allocated landmarks will be rejected and no corrections performed. The effect can be seen particularly at long straights, but is moderate on curved paths, when having a sufficient number of corrections by lane segments.

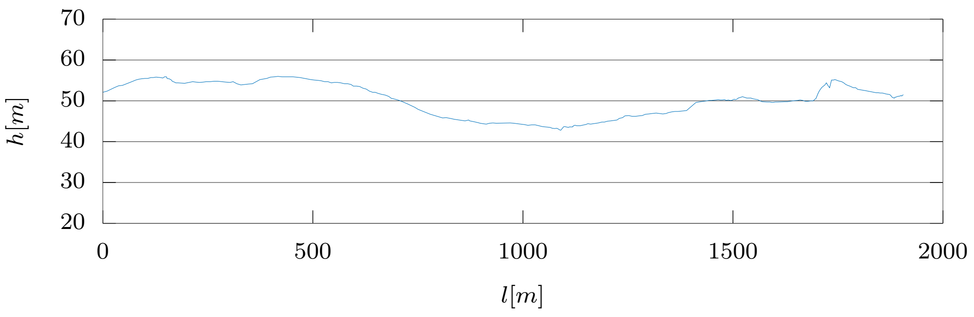


Figure 6.17.: *Profile of first test track.*

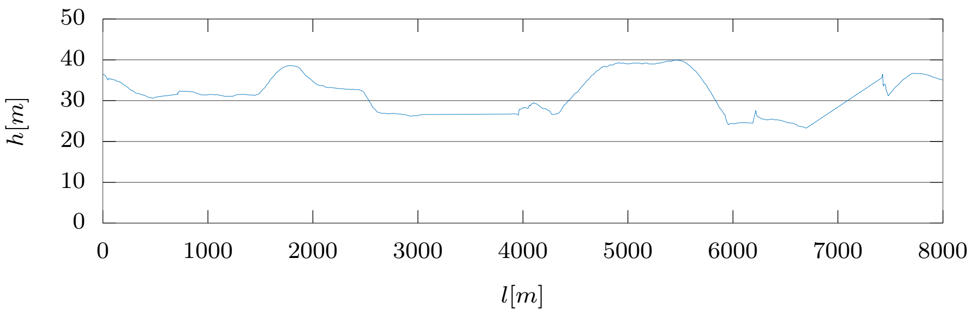


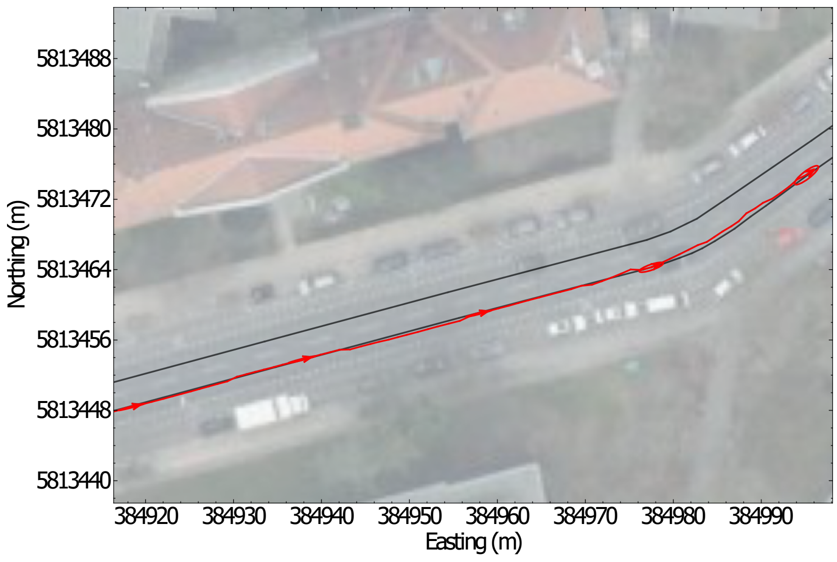
Figure 6.18.: *Profile of second test track.*

The next point to evaluate, is the semantic term of corrections. Therefore, the lane selection process and assessment will be investigated. Furthermore, lane changes and their impact on the position estimation and thus for the particles states will be evaluated.
As introduced in 5.2.3, for each lane segment candidat,e a weighted score/cost function containing distance and angle differences and further the lane direction type, if available, is calculated. In practical use it was shown, that sort the candidates by score only, leads to a mainly distance-influenced particle weight at correction step. Preceding and consecutive lane segments cause a substantially weight gain. Parallel lanes, e.g. at multiple lane crossings instead, suffer from a much to less weight gain. As result, the estimated pose moves along one of the map lanes, often the nearest, but not necessary the correct one, especially at multiple lanes. The filter behavior is often too rigid, to react properly on multiple lane situations. The defined scaling options for the cost function, were not sufficient to solve this issue by parameterization.
I decided, to add a *lane scope* cost term to the score functions. The scope term describes a measure for the spacial relation of a lane segment to the current position. This can be done straight forward, with projecting the current estimate $\vec{p}_c$ orthogonal to lane candidates and evaluating the value $t$ from the linear equation through start $\vec{p}_s$ and end $\vec{p}_e$ of segment

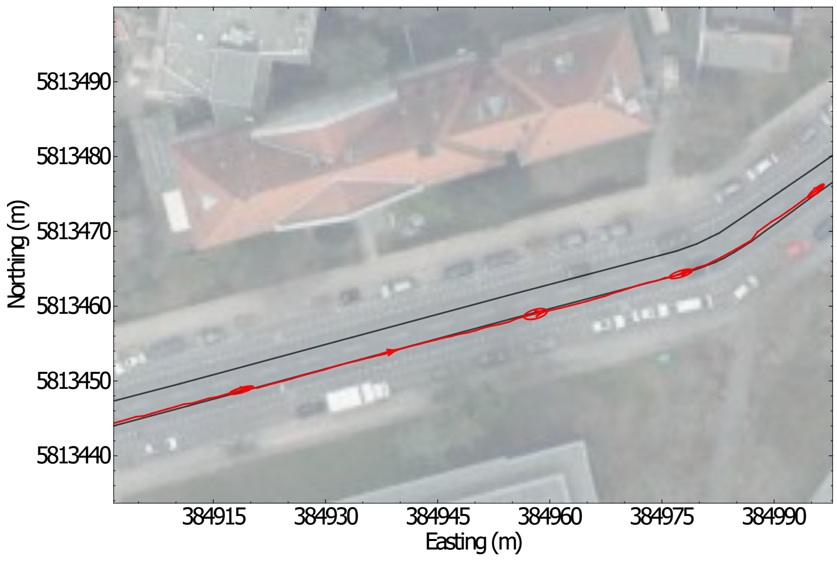
$$\begin{aligned} g : \vec{q} &= \vec{p}_s + t\vec{p}_e \\ t &= \frac{(\vec{p}_c - \vec{p}_s) \cdot (\vec{p}_e - \vec{p}_s)}{|\vec{p}_e - \vec{p}_s|} \\ t \in \mathbb{R} &= [0..1] \longmapsto \text{inside lane scope} \end{aligned} \tag{6.1}$$

The procedure cause only few additional computations, because the projection must be done for the distance to map lane evaluations anyway. Using the additional scope score, extracts the parallel lanes for multi modal correction and prevents unintended multiple corrections on lane-segment-transitions. The weights for distance and angle of a lane segment should be adjusted to a balanced impact.
In some cases, corrections should be suppressed for a specific period. This is especially important during lane changes. Corrections during a lane change causes missalignments on the multivariate density function formed by the particle weights. The resulting position guess, tends to keep the estimate at the lane it was. It causes the correction (landmark) contradicting the prediction (odometry), because lane measurements are not clear, when passing the lane border. The leaved lane is already measured and the opposite lane border of the adjacent lane is newly measured. In this situation, left and right lane border are available, but the estimator has no knowledge, about the information comes from adjacent lanes and not from a single lane.
Performing only prediction steps and suppress corrections in such situations solves this issue. Detecting the lane change, is realized by evaluation of first coefficient in lane models polynomial, representing the lane border deciation (4.19). An in-

creasing standard deviation on this value for a specific period, indicates either a lane change or unstable measurements. Both cases suppress any correction steps.



(a)



(b)

Figure 6.19.: *a) Event triggered corrections lead to jumpy path due to the direct impact of measurement variances. b) Shows a time triggered correction behaviour (200 ms) with smoother path and more confidence to odometry.[5].*

## 6.4. Strategies on re-sampling

Regarding section 5.2.5, the goal for re-sampling is to concentrate the set of particles in a region of the state space with highest probability and to avoid degeneration problems. As proposed in related work, this can be achieved with the use of effective sample size

$$N_{eff} \approx \frac{1}{\sum_{i=1}^{N}(\omega_k^i)^2} \tag{6.2}$$

as described in 3.5, with $N$ the number of particles and $\omega_k^i$ as their weights. Generally, the $N_{eff}$ is a good indicator, for a degenerated particle set. It is suitable for pose estimation systems, with a correction rate, that is high enough and with only small errors contributed from prediction cycles. This can be guaranteed with integration of an IMU and/or LiDAR to the sensor set.
With the given sensor set and the sparse landmarks, the confidence on that single value reveals an insufficient re-sampling strategy. The most important limitation is, that $N_{eff}$ only gives an indication for the distribution of weights, but not for the distribution of states. A filter state, with a good balanced weight distribution can allow very large differences between the positions guess and actual position. Actually, incoming corrections will be rejected if this gap becomes to large. The behavior is shown, when the particle set suffers on absence of corrections, for example when lane information is missing for longer period of time. This is immanent for every localization system, that uses sparse or uncertain information for correction. To deal with this problem, I pointed out two strategies:

- A situation based adaptive re-sampling in spatial and probability domain, with option to keep particles.
- Implementing a time- and plausibility-triggered re-sampling, additionally to the existing by $N_{eff}$.

An effective way to keep the particle set being a good representation for the real vehicles position state and variance, is *probability dense function shaping*. Therefore, the following options were investigated:

- Use distributions mean or most probable state for sampling-center,
- sampling along lanes vs. centralized sampling,
- definition of a history-proportion for the particle set.

The first option is an opportunity, to make the particle set more responsive to new observations, especially after a longer period ($t > 5.0s$) without correction steps. Furthermore, it prevents the whole particle state to drift away from real position, by sampling new particles around the latest corrections. The disadvantage is a jumpier output pose.
The second option, is to give the particle set a pre-shaping, by sampling along map

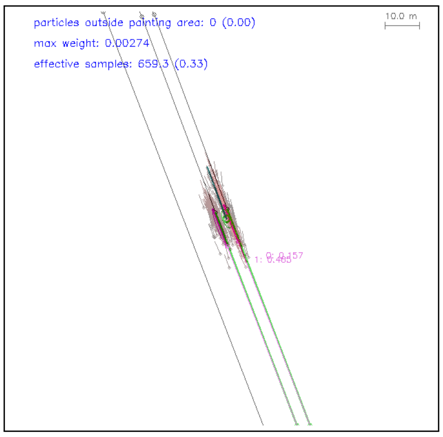
segments. This should make the set a better representation of the state space, by concentrating the position estimation on driveable path options. The idea is not new and proposed in related work. This was tested for both recorded scenes and showed divergent results. On one hand, it showed good results at multi-lane crossings and other nearly straight multi-lane roads. On the other hand, at multi-lanes with strong curvature, it reveals some problems. It can be illustrated, passing the roundabout in test track two. Sampling new particles according to lane segments at high lane curvature, is a good representation of the state space for quite a short time. With the interaction and more often caused prediction - based on vehicle motion - the particles states do not fit the lane course already after short time. The particle set and its weights, is then an inappropriate representation of the actual probability dense function of the vehicles motion process. Because, the more a particle is distanced from real position, the more the vehicle motion does not fit the lane course. The set drifts apart, further corrections could not be applied properly and the deviation of positions estimate increases.
The method with the most positive impact on stability and precision to the output pose, is to keep a part of the particle set as history during resampling. Contrary and new to the proposals in other works, a particle management is required. The particle management must ensure, that the strongest particles are kept, weak or drifted particles are substituted against new ones and the distribution and probability density stays balanced and consistent ($\Sigma p_i = 1$). To have a sufficient number of free floating particles, the subset for history should be essentially smaller then the floating particles. I choose a 4:1 relation ($f = 0.2 : N_{sub} = 0.2 * N_{all}$). This showed a good balance between the capability to aggregate state and weights along multiple lane options and being flexible to re-sample a significant amount of particles into new states, when incoming corrections indicate a new position. A further task, is to keep the weights of the history subset in balance to the whole set. With multiple applied corrections, affecting the history subset, the weights tend to $\Sigma p_{sub} \simeq 1$. It is clear, that re-sampling 75 percent of the particles with weights near null is far away from being balanced. To avoid this, the strategy is to not let the summed weights of the history set overrun a specific probability. For the given problem, I choose a maximum probability of $\sigma p_{sub} = 3 \times f = 0.60$. In words: the history set should hold 0.6 of the overall weight as maximum, but can be lower. In a simple set of 5 particles, with one history- and four sampled particles, the state of the one history sample can reach a maximum probability of $p = 0.60$. Generally speaking, the particle weights in history set must be normalized to $p = 3f$. The remaining weight is sampled into states, according to the following error model (covariance matrix) for re-sampling.

$$\Sigma = \begin{bmatrix} \sigma_a^2 & 0 & 0 \\ 0 & \sigma_b^2 & 0 \\ 0 & 0 & \sigma_\phi^2 \end{bmatrix} \tag{6.3}$$

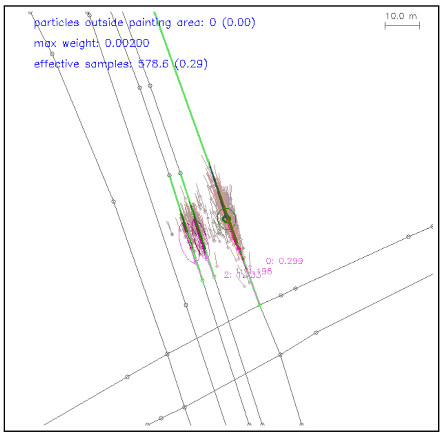
The set is therefore divided into two parts (each $N_{sub} = 0.4 \cdot N$), the first part holds 90 percent of the remaining weight ($p \geq 0.54$), and the second part 10 per-

cent ($p \geq 0.06$). The error model for the fist part is $\sigma_a = 1.5m$ as error on first principal component, $\sigma_b = 1.5m$ as error on second principal component of the multivariate distributed position and $\sigma_\phi = 5°$ as model error for vehicles heading. For the second part, the $\sigma$ is doubled. The goal is, to have a large amount of particles next to the current pose with medium weight, and samples to fill a wider state space properly, but low weighted. A distinction in standard deviation on principal components for the multivariate distribution is not necessary. One of the advantages using this method is, that keeping a subset of particles and applying multiple corrections, leads to an enrichment of particles along the lane segment path with highest probability, figure 6.20. The density function, the particle weights represent, becomes multi-modal. It is obvious, that this requires another strategy for position estimations out of this set. Using the whole set, leads to a weighted mean somewhere between the local maximums of the probability dense function. Therefore, a subset of particles around the global maximum is selected and the positions guess from this subset of particles is determined in known manner. The aggregation of strong particles around the real pose, ensures at roads with strong curvature, a particle set motion, that is common with the lane segment course and therefore with the real driven path.
The behavior of particles drifting apart and interfere the positions guess in curvy lanes, as mentioned for direct path sampling, is at least moderated with this method. The estimated position is more stable and the motion of it, is smooth and a well representation for the vehicles motion.

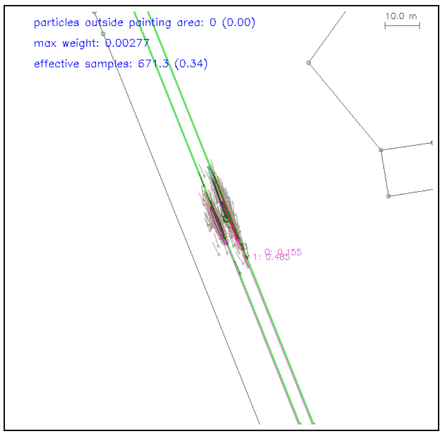
Next to the probability dense function shaping, I decided to implement a time and plausibility based re-sample indication. The time based indication is straight forward. If activated, the re-sampling is performed according to $N_{eff}$, but with minimum duration between re-sample activities. This prevents the filter to re-sample after corrections with stable measurements. A low cost (good fit) from lane selection process, leads to small sigmas for correction and thus to an unbalanced distribution of weights. The effective sample size would indicate re-sampling, but in fact, the particle weights represent the real state very precise. In the case of corrections with good matching measurements, a permanent re-sampling is prevented by applying a minimum time span. A plausibility check over all particles can solve this. Therefore, single particle states can indicate an invalid overall state. It can be shown, that in some cases, when having poor landmark information and simultaneously increased error on prediction (wet, bumpy or slippery ground), the estimated pose drifts away from real position. Nevertheless, correction updates cause an increasing weight to some of the particles in the near of the actual position. Depending on the number of particles affected by this correction, the impact to $N_{eff}$ is small. The situation can be recognized, by detecting large gaps between the position estimate (based on MMSE) and the area with the strongest particles. Therefore, the particle set is sorted by weight and a MMSE-pose from the strongest particles is determined. The poses must be compared. I defined a distance $\geq 3.0$ m as indication for re-sampling, what corresponds approximately to typical lane


particles outside painting area: 0 (0.00)
max weight: 0.00274
effective samples: 659.3 (0.33)
10.0 m

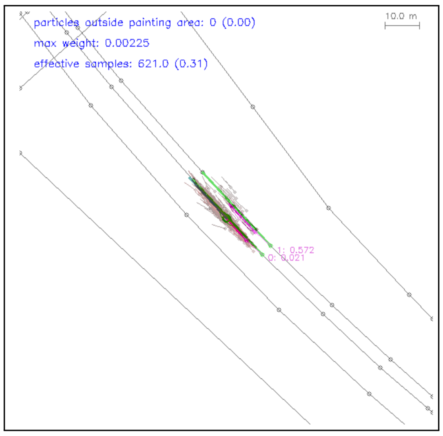
(a)


particles outside painting area: 0 (0.00)
max weight: 0.00200
effective samples: 578.6 (0.29)
10.0 m
0: 0.299

(b)


particles outside painting area: 0 (0.00)
max weight: 0.00277
effective samples: 671.3 (0.34)
10.0 m

(c)

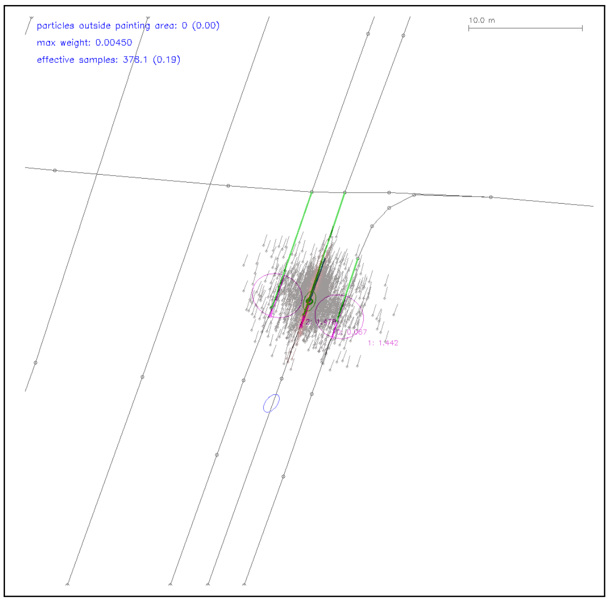

particles outside painting area: 0 (0.00)
max weight: 0.00225
effective samples: 621.0 (0.31)
10.0 m
1: 0.572

(d)

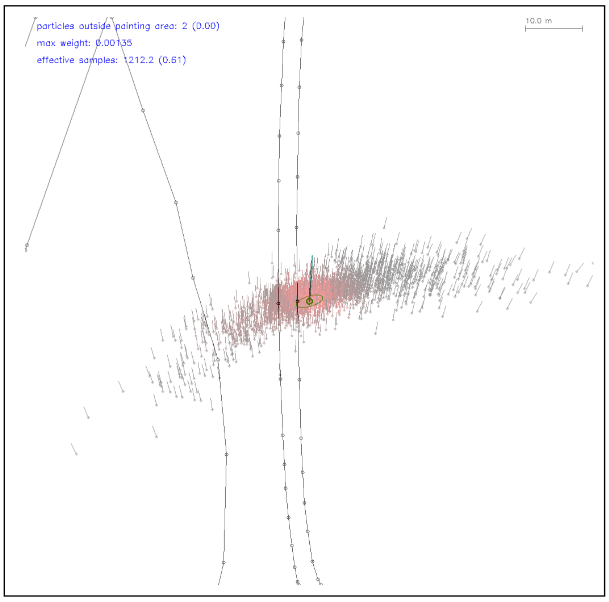
Figure 6.20.: *Illustrations for enrichment of particles along map lane segments at different situations. The visualization shows the concentration of particles (shown as pins) along (black) lane segments.*

width on urban streets.

The impact, the way of re-sampling has to correction cycles, should not be underestimated. The 'shape' of the multivariate distribution embodied by the particle weights has a significant impact on the $\chi^2$ tests. Generally spoken, the more the set weights converge, the more the filter becomes unresponsive for further corrections, up to a complete blockage. Otherwise, a widely balanced distribution, that converge fast into new states, often shows jumpy behavior. Spontaneous reactions on correction updates, especially from lane segments, cause the risk to 'follow a wrong lane segment path'. In 6.6 discussions on that issue and the interaction of predictions, corrections and re-sampling are presented.


particles outside painting area: 0 (0.00)
max weight: 0.00450
effective samples: 378.1 (0.19)
10.0 m
1: 1.442

(a)


particles outside painting area: 2 (0.00)
max weight: 0.00135
effective samples: 1212.2 (0.61)
10.0 m

(b)

Figure 6.21.: *a) Shows failed $\chi^2$ test. The measured landmark was near the car, and therefore the projected pose modeled with small variance (blue circle). The particle set converged to small variance as well. As result, the projected pose from landmark is neglected from the $\chi^2$ test. b) Illustration of the particle set and after a longer period of time without correction updates due to sparse landmark information. The actual course is the left of the two parallel lanes (black). The effective sample size is $N_{eff} = 1212.2$ (max 2000). Thus, it indicates that the particle set seems to be effective and estimation is reliable. Actually, the estimation it is around five meters away from correct position.*

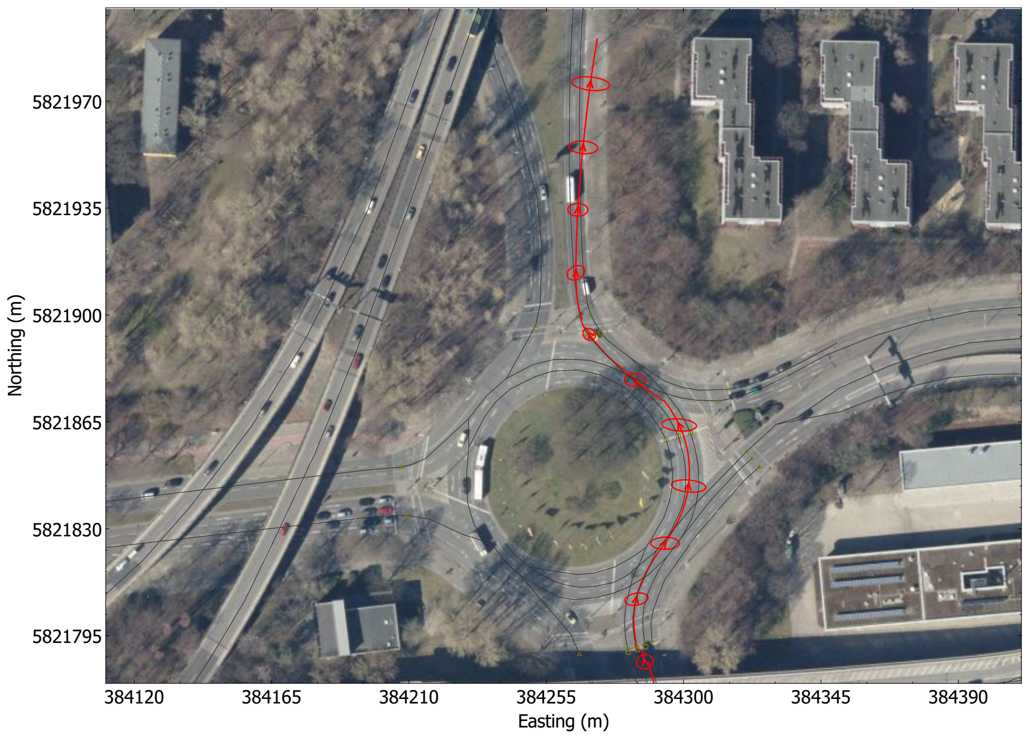


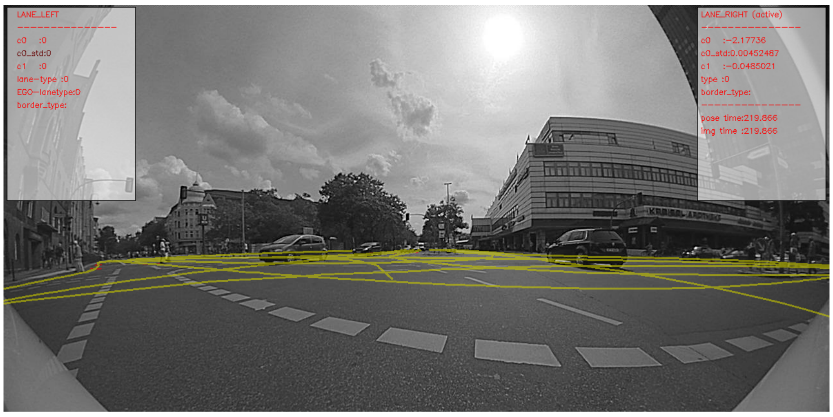
Figure 6.22.: *The illustration shows the estimated path (red) at driving around a roundabout. The driven direction is from south to north (bottom to top in image). Lane information inside the roundabout is sparse and is completely missing at the exit. The map data is shown with black color. One can see diverse effects: An increasing error (enlarging variance ellipses) by odometry updates. A path deviation, that is corrected short before leaving the roundabout. The actual path was on middle lane. And finally the missing lane information at exit, that causes a heading error and therefore a wrong path (top of image).*

## 6.5. Precision

To perform an precision evaluation, a path reference is needed. On test sites it is the normal choice to use DGPS signal in combination with an IMU. In urban surroundings, DGPS validity is given only in a small area around the DGPS station, where multi-path scattering does not interfere. Actually, this work was dedicated to do research on this issue and think about solutions. A localization system for urban areas cannot be based on DGPS in a sensible way. Hence, another method to get a reference path must be found.
I decided to use a method for this work as described in the following sentences. I used IPM to generate a local image around vehicle. This image was used to match into image data from digital orthographic photos and to derive the pose, from where the IPM image was shot. The process is handcrafted and even for humans it can be challenging to align the IPM image according to the areal image data. The precision, one can achieve with this method depends on grade of detail of the areal photos. In this thesis, one pixel corresponds to $0.2 \times 0.2$ m, what can be assumed as minimal error for the reference path. In future work, this process should be automated, using alignment methods from computer vision, that can achieve sub-pixel precision and therefore a smaller error. At each full second in the recording, one reference point was set. The estimated pose, is interpolated if necessary, to meet exactly the equidistant 1.0 s - resolution of the reference path. There is an advantage, in using image signals to generate the ground truth. Reference path and test subject, bases on the same input. No further synchronization efforts are needed. The problem of synchronizing reference sensor data (e.g. from IMU) with test data does not exist.

To evaluate the differences between estimated pose and reference point, not only the easting and northing coordinates from the UTM system is used, but car relative coordinates. This is more intuitive and considers the vehicles heading. For example, a pose error in car coordinates with $\epsilon_x = 10.0m$ and $\epsilon_y = 0.5$ can be read as a good lateral pose estimation, but with serious error on longitudinal component. Compared to that, the easting and northing differences would be quite cumbersome, except the driven path would have exactly east-west or north-south alignment.

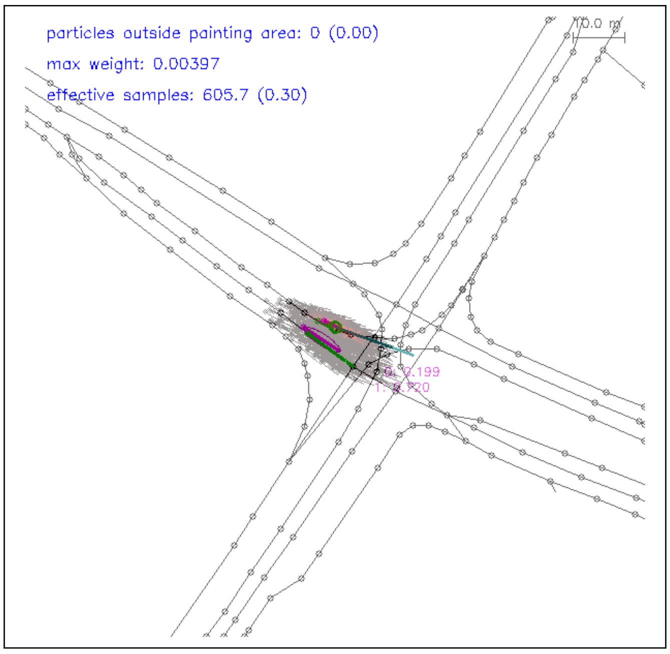
For precision evaluations, one must keep in mind, that the estimation process depend on the measurement precision of input data, the error of that input data and particularly, if the error model meets that data. Since real input data is disturbed or even missing, the estimated pose will always be different to real pose and stays an approximation. A best guess, as far as the known history combined with new information allows. Using a particle filter, the state space is quantized by the particles and the particle set with weights embodies the multivariate distribution, or rather the probability dense function for the observed process: The change of vehicle pose over time.



```
LANE_LEFT
------------------
c0    :0
c0_std:0
c1    :0
lane-type :0
EGO-lanetype:0
border_type:
```

```
LANE_RIGHT (active)
------------------
c0    :-2.17736
c0_std:0.00452487
c1    :-0.0485021
type :0
border_type:
------------------
pose time:219.866
img time :219.866
```

Figure 6.23.: *A crossing scene in practice. Reprojected map lane segments shown in yellow. Information from lane measurement is shown inside the highlighted boxes.*



```
particles outside painting area: 0 (0.00)
max weight: 0.00397
effective samples: 605.7 (0.30)
```

Figure 6.24.: *Visualized particle set in front of the multiple lane crossing (three lanes). Same scene as in figure before, 6.23.*

It is obvious, that the first screw to adjust the process is manipulating the number of particles. There is a trade-off between a good representation of the dense function, where fine granularity is desired, and the computational efforts that comes with it.

The number of corrections that can be performed depends on the availability of landmarks, real existing and detectable. Therefore, one has to consider, that er-

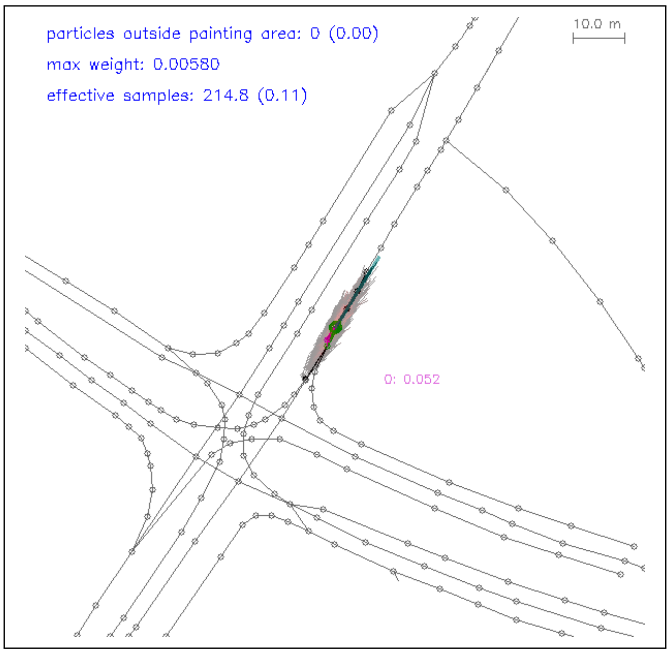

particles outside painting area: 0 (0.00)
max weight: 0.00580
effective samples: 214.8 (0.11)
10.0 m
0: 0.052

Figure 6.25.: *Particle convergence afterwards passing the intersection.*

ror determination is not disassociated from the track conditions at every reference point. It is not like on test site, where one can expect an error that is nearly constant on the whole course with consistent track characteristic. Thus, the presented tables distinguish between overall values, without consideration of varying conditions and the positions on track, where corrections were performed. Finally, the resulting position error is shown, immediately afterwards correction.

The way of error modeling and especially the assumed errors on differently mea-

Table 6.15.: *Position estimation results. The error is determined between estimation and reference.All values in meters [m]. The results are from a stable system. E.g. waiting at crossings. Thus, it corresponds to an ideal system state and shows the best possible results on perfect track conditions. There is no maximum longitudinal error, because it increases without correction on straight tracks, until arrive to next landmark. See table 6.17 for correction capability of longitudinal errors.*

| nb of particles | $max\ \lvert\epsilon_y\rvert$ | $\overline{\lvert\epsilon_y\rvert}$ | $\sigma^2_{\epsilon_x}$ | $\sigma^2_{\epsilon_y}$ | $\sigma_{\epsilon_x}$ | $\sigma_{\epsilon_y}$ |
|---|---|---|---|---|---|---|
| 100 | 1.004 | 0.421 | 0.065 | 0.042 | 0.242 | 0.203 |
| 500 | 0.778 | 0.332 | 0.011 | 0.005 | 0.103 | 0.074 |
| 1000 | 0.853 | 0.416 | 0.035 | 0.010 | 0.136 | 0.095 |
| 2000 | 0.730 | 0.340 | 0.001 | 0.001 | 0.036 | 0.038 |
| 3000 | 0.741 | 0.526 | 0.001 | 0.001 | 0.033 | 0.028 |
| 5000 | 0.809 | 0.492 | 0.002 | 0.001 | 0.039 | 0.034 |

Table 6.16.: *Position estimation results. Lateral error between estimation and reference, in [m], is shown during a drive through urban environment. Longitudinal positioning error increases between longitudinal corrections, as expected.*

| nb of particles | *max* $\epsilon_y$ | $\overline{\lvert\epsilon_y\rvert}$ | $\sigma^2_{\epsilon_y}$ | $\sigma_{\epsilon_y}$ |
|---|---|---|---|---|
| 100 | 3.368 | 0.560 | 0.719 | 0.848 |
| 500 | 4.493 | 0.662 | 1.069 | 1.034 |
| 1000 | 2.829 | 0.429 | 0.396 | 0.629 |
| 2000 | 4.539 | 0.559 | 0.807 | 0.898 |
| 3000 | 3.400 | 0.601 | 0.837 | 0.915 |
| 5000 | 4.841 | 0.493 | 0.814 | 0.902 |

Table 6.17.: *Position estimation results. Longitudinal positioning error before and after longitudinal correction at stop lines. The last two columns show the error after whole track was driven in absolute value and in percentage over track length. All values in [m].*

| | before | after | | |
|---|---|---|---|---|
| nb of particles | $\lvert\epsilon_x\rvert$ | $\lvert\epsilon_x\rvert$ | $\lvert\epsilon_x y\rvert_{track}$ | error over track length [%] |
| 100 | 8.820 | 0.150 | 2.699 | 0.225 |
| 500 | 8.263 | 0.324 | 6.192 | 0.516 |
| 1000 | 5.835 | 0.229 | 2.364 | 0.197 |
| 2000 | 9.041 | 0.041 | 2.560 | 0.213 |
| 3000 | 6.516 | 0.050 | 1.747 | 0.146 |
| 5000 | 5.252 | 0.075 | 2.622 | 0.219 |

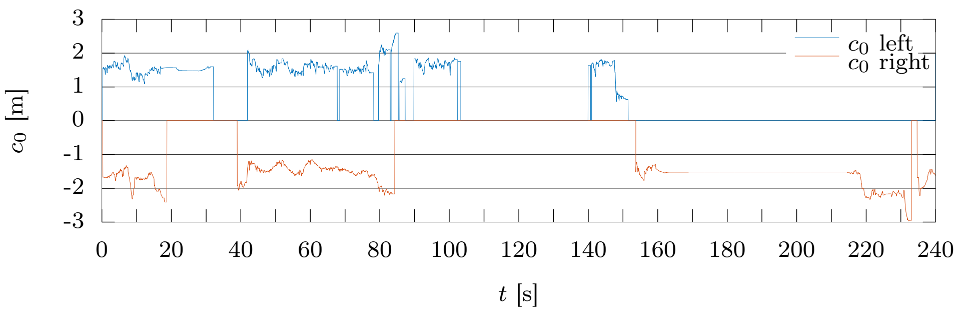


Figure 6.26.: *Availability of lane information on test track. If curves fall to zero, there is no lane border information available.*

sured landmarks and odometry inputs, further the difference in occurrence for each input, affect the position estimation process massively. It could be observed, that error modeling on corrections, with real practical values, leads into two troubles. The first is the very fast converge of particle weights into states, where corrections have its maxima, and thus to a very jumpy output for estimated pose. The second - more serious - issue is the fact, that the strong aggregation leads to problems in hypothesis significance test. As showed in section before, the parameters for

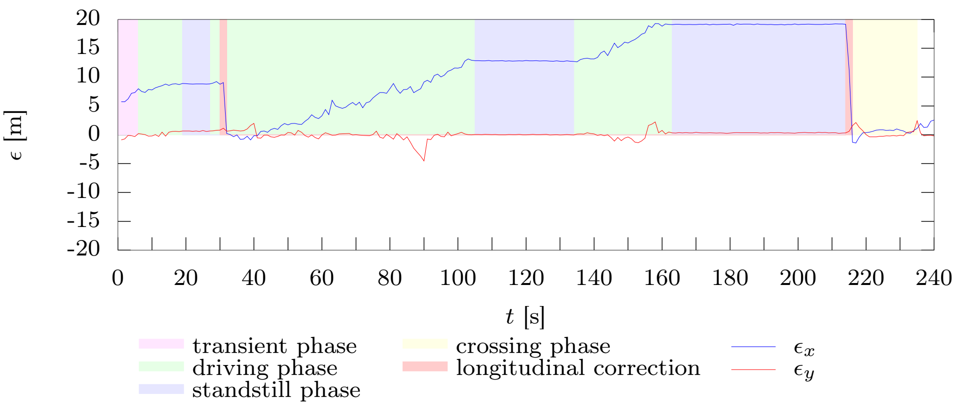
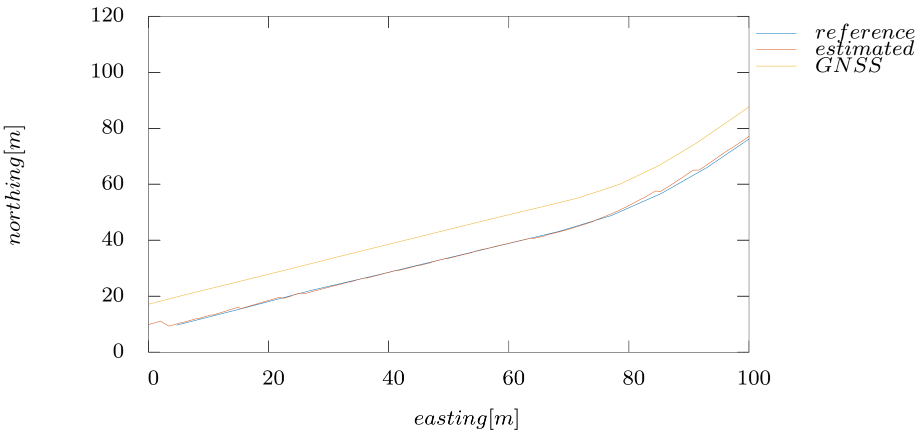
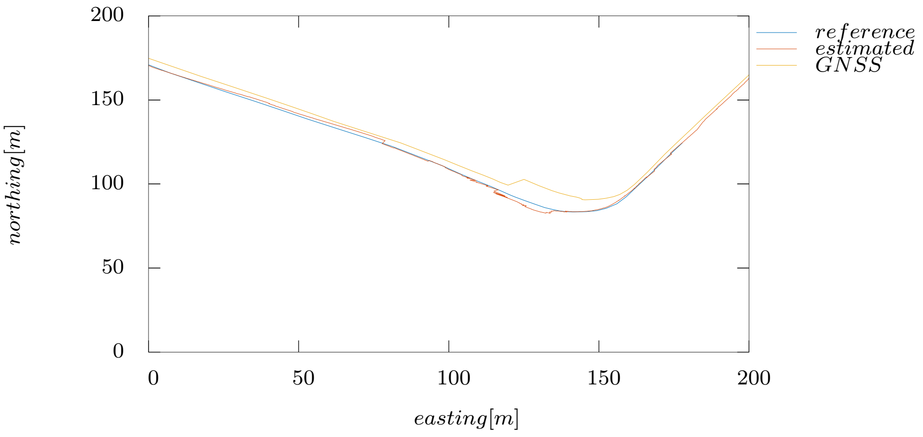


Figure 6.27.: *Deviations between reference and estimation. The different scene states and events are colored and explained inside legend. The longitudinal correction capability is shown on x error falling back to an error near null.*

the $\chi^2$ test can not be used as known from mathematical problems. The upper boundary has to be set in a way, that more acceptance is achieved. For example, using a standard deviation of 0,10 m to model the lane measurement error, leads to comparison of a 0-hypothesis (the position estimate and its $\Sigma$) and a position hypothesis for correction, with at least one very narrow probability dense function. The significance test will fail, even though the new position hypothesis is correct and should be accepted.
To prevent the described behavior, error modeling becomes a more experimental task, than using observed signal deviations. I decided to adjust the error modeling in a way, that filter output stays smooth, but corrections have enough impact, to prevent the filter drifting apart.

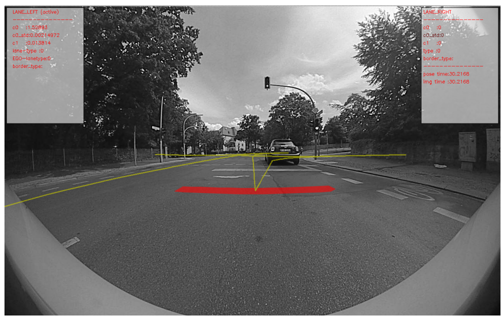


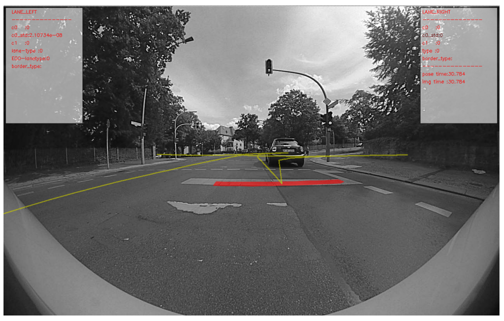
(a)

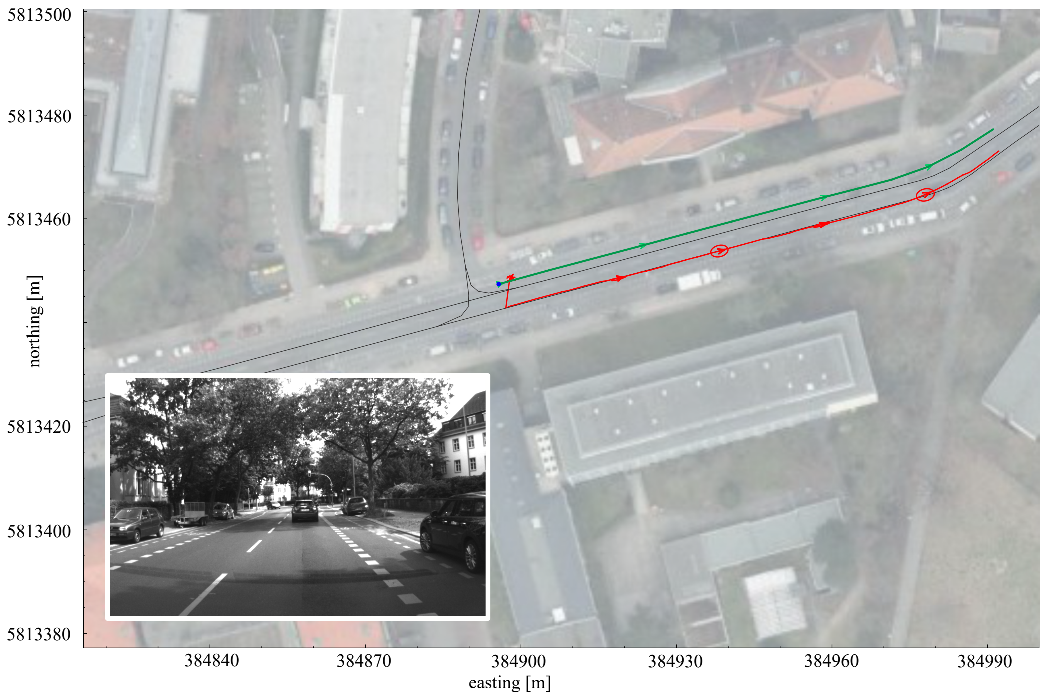


(b)

Figure 6.28.: *Track plots of reference, estimation and GNSS path. Illustration aa) shows the deviation of pure GNSS positioning and estimated pose to reference path in detail. b) Shows the paths at the multiple lane crossing near the end of the track. The jumps on GNSS path when entering the crossing due to changes in signal propagation are visible, as well as multiple corrections on the estimated path. Furthermore, a temporary miss-alignment to wrong lane, when entering the crossing is visible on the estimated path.*


(a)


(b)

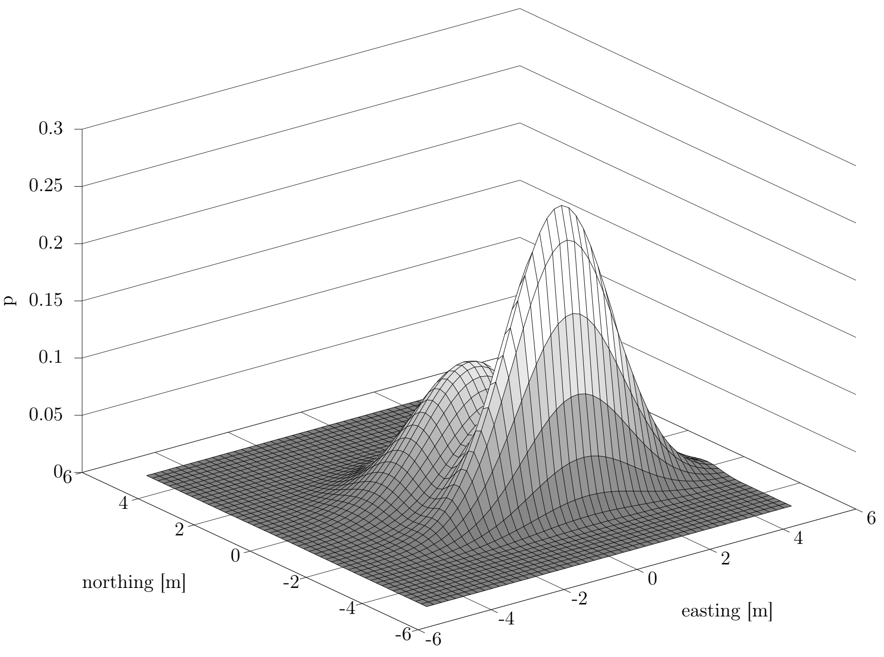


(c)

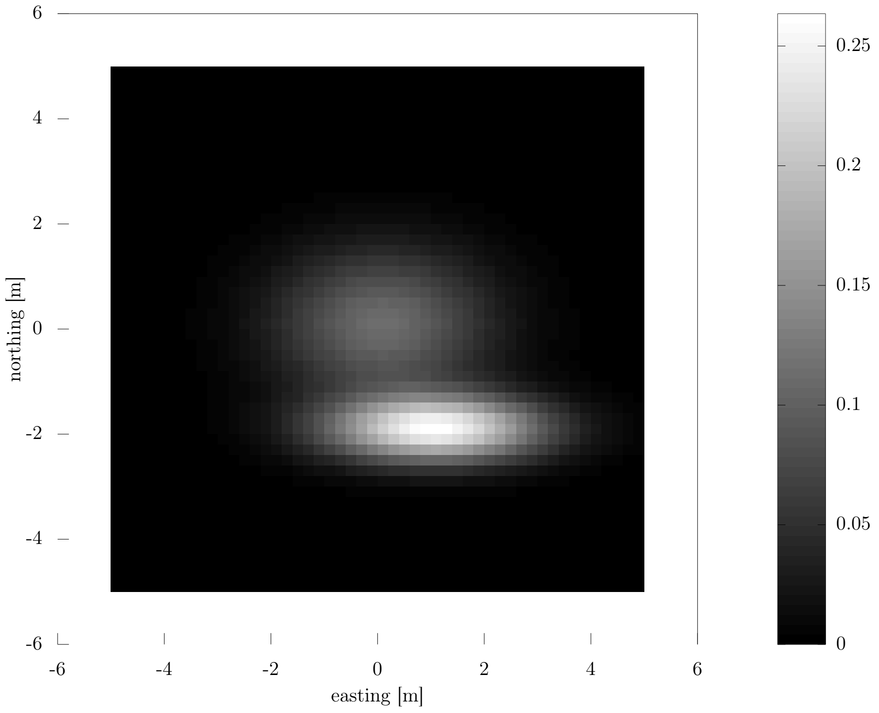
Figure 6.29.: *Correction step and re-projections. Reprojected map lane segments shown in yellow (lane) and red (stop line).*
*a) At the stop line, a longitudinal correction is performed. The longitudinal error is visible. The red colored stop line (map data) does not meet its actual position.*
*b) Shows the situation after correction is performed. Map structure and environment are aligned.*
*c) Illustrates the error of the GNSS path (green) against the map-corrected path (red) from the localization system in this thesis.*

## 6.6. Concluding remarks

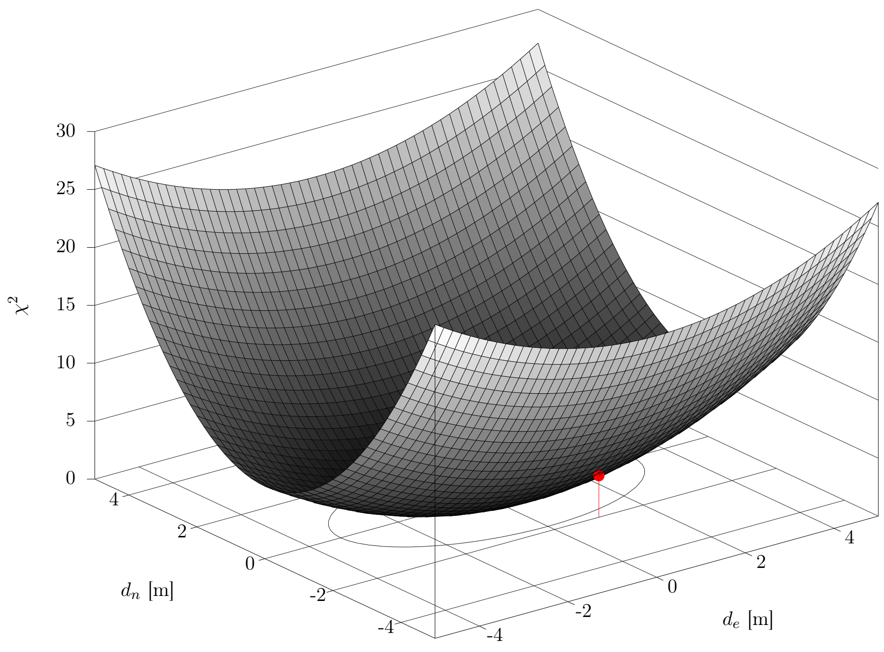
At the end of this section it is clear, that the interaction between prediction, correction, resampling and its timings is a complex process and hard to control. The influence of the different set screws is shown exemplary in 6.28. There, a probable positions guess and a (landmark) measurement are shown along with their multivariate density functions. Both states are described with the state variables $\mu$ and their co-variances $\Sigma$. For more clarity, only the first two dimensions (easting and northing) of the state is visualized. The shown situation is derived from a realistic event as it occurs permanently. There is the estimated position described by the particles distribution at coordinates origin and the measurement (derived position from landmark) at $\mu_{meas} = (1, -2)$ with a distance of $d = |(e, n)|$. The distance between current guess and hypothese is $\sqrt{5} = 2.24m$ and the distribution parameters meet practical values. One can take this constellation for a valid measurement, especially, when last correction on particles was several time ago. Using the $\chi^2$ criterion with a high upper bound at $p = 0.9$, yields for the given example to $\chi^2 = 3.583$, what is outside the acceptance area. The measurement would be classified as outlier. The area of acceptance for this special constellation of current guess and hypotheses, is shown with the bottom ellipse in the lower image of figure 6.28. The plot illustrates the $\chi^2$ test results for distances up to $d = |(4, 4)| = 5.66m$ and the given distribution parameters. The result for the exemplary performed test is marked with a stem.
One can observe, that the suggested $\chi^2$ test [32], as used for mathematical questions, is not directly applicable for the outlier problem in position estimation tasks with sparse landmarks. The upper bound of the $\chi^2$ test must be used in a more open way. However, it is a statistical value to face (normal) distributions and can be used with adjustments.


p
0.3
0.25
0.2
0.15
0.1
0.05
0
northing [m]
easting [m]
-6
-4
-2
0
2
4
6

(a)


northing [m]
easting [m]
6
4
2
0
-2
-4
-6
0.25
0.2
0.15
0.1
0.05
0

(b)



(c)

Figure 6.28.: *Distribution of $\chi^2$ test results for different distances d between landmark and current particle average, for exemplary multivariate distributions $N_{set}(\mu_{set}, \Sigma_{set})$ and $N_{meas}(\mu_{meas}, \Sigma_{meas})$ with $\mu_{set} = (0, 0)$, $\Sigma_{set} = diag(2.0, 1.0)$ ; $\mu_{meas} = (1, -2)$, $\Sigma_{meas} = diag(2.0, 0.2)$. The bottom line marks the area where $X^2_{(0.9,1)} = 2.71$, with $p = 0.9$ and $f = 1$. The measurement $\mu_{meas} = (1, -2)$, shown as red pin, is outside the acceptance area and thus neglected as outlier for this example.*

# 7. Conclusions and future work

I have shown in this thesis, that extracted road markings can improve localization substantially, through investigating two approaches. The first classification approach is based on contour extraction and the determination of geometrical values like moments and relations, on the other hand, the second classification approach is based on HOG descriptor for road markings.
The geometrical method is a fast and reliable way to classify markings. One advantage is, that this method not only detects arrows, but line segments too. Thereby, stop lines and pedestrian crossings can be detected in a straight forward way with additional spatial analysis.
The HOG method shows more robustness against disturbances and combined with a linear kernel SVM a very good relation between recall and precision (F-score=0.93). A detailed view on strengths and weaknesses of the methods is presented in Table 7.1
The influence of the classifier type (SVM vs. random forest) is not as big as expected. Both methods have nearly similar results, when using the same training and validation sets. The random forest classifier processes slightly faster and therefore it should be preferred together with use of geometrical features.
Generally, understanding the road markings by its shape has advantage over classical lane detection approach. It is typically based on gradient grouping and finding light-dark-transitions. But especially in urban areas, where the shape of road markings often violate the desired model restrictions, the approach of understanding shapes leads to more robust detection performance.
A sufficient marking detection and classification is a good base for landmark based position estimation. Specially at the moment when the vehicle passes the markings, a high measurement precision can be achieved. With increasing vehicle speed, rolling shutter effects and motion blur become relevant. For typical driving velocity in urban areas, no disturbing influences were noticed. In the following steps, the provided information were used to estimate the pose of the car. With switched of GNSS, promising results for position accuracy could be achieved while driving in urban environment. Within this process, lane boundary information is used for lateral correction, stop lines and pedestrian crossings for longitudinal path correction. Furthermore, the detected arrow markings are usefull to get the correct lane at crossings with multiple lanes.
The comparison of multi-class detection and single class detection did not show potential, that warrants the additional computation efforts.
Beyond positioning, road markings can provide the meaning of a lane. The detection of bicycle lanes, parking spots for disabled people and other special markings

can be a useful information for manifold driving tasks.

| geometrical | HOG |
|---|---|
| + | + |
| fast | strong increasing recall rates (good results on small training sets) |
| good recall rates | avoids the thresholding problem of contour finding |
| rotation invariant, without additional algorithm specialization | detection of complex shaped markings possible (e.g. wheelchair or bicycle signs) |
|  | better robustness towards disturbances |
| rotated line segments can directly be interpreted as stop lines or pedestrian crossings | better robustness towards disturbances |
| - | - |
| Thresholding problem (prone to changing environment, shadows) | slow, when using a dense sliding window |
|  | Not scale or rotation invariant in standard implementation |

Table 7.1.: *A brief results sum up for classification of road markings*

It is was shown, that the extracted road markings can be used to perform corrections in a particle filter, that is updated with vehicle motion. The achieved precision is presented in table 7.2. Lane level localization needs a positioning error below 1,0 meter for well working lane association. That is derived from a typical lane width of minimum 3,0 m and a remaining alternation on driving path around 0,5 m to every direction. Passing that restriction means passing the lane border. As shown, the presented localization method typically stays below the 1,0 meter constraint, even with a small number of particles. Furthermore, it is not essential to have much particles for a sufficient precision, because in practice the localization error depends more on availability of landmark information, a precise motion model for vehicle, suitable error modeling and the driven road itself with its specific properties like flatness and surface. Nevertheless, at time of correction or with enduring availability of landmark information, the number of particles matters a lot. This is important for systems equipped with sensors that support a more dense information from environment (e.g. LiDAR), on test tracks and on well constructed roads, fully provided with excellent markings. This can be seen, on the MSE and standard deviation in ideal situations (durable lane information at stand still), 6.15. The more particles, the more stable the localization solution. In transition from 1000 to 2000 particles, the errors standard deviation falls from >0,1 m to <0,04 m. For longitudinal correction, the number of particles always influences the correction capabilities of the systems. After longitudinal correction occurs, the error always decreases to a low sub-meter level. With 2000 and more particles below 0,05 m, 7.2.

Table 7.2.: *Results overview for pose estimation precision with various number of particles. All values are metric.*

| particles | 100 | 500 | 2000 | 3000 |
|---|---|---|---|---|
| best case | | | | |
| $\sigma_{\epsilon_x}$ | 0.242 | 0.103 | 0.036 | 0.033 |
| $\sigma_{\epsilon_y}$ | 0.203 | 0.074 | 0.038 | 0.028 |
| practical values | | | | |
| $\overline{\lvert\epsilon_y\rvert}$ | 0.560 | 0.662 | 0.559 | 0.601 |
| $\sigma_{\epsilon_y}$ | 0.848 | 1.034 | 0.898 | 0.915 |
| $\epsilon$ after lon correction | | | | |
| $\lvert\epsilon_x\rvert$ | 0.150 | 0.324 | 0.041 | 0.050 |

It could be shown, that the used landmarks (lane borders, stop lines, crossings) are typically not available in a density, that is needed for an optimal estimation process in real urban environment. Methods were shown to bridge these lacks of track information. Furthermore, the thesis showed, that the error modeling must be adjusted to this issues in a sophisticated way. It is not sufficient to determine error covariance matrices with typically expected precision on each variable. An assessment for landmarks is needed, not only to detect outliers, but to score each landmark and lane proposal and perform variance modeling related to landmark score. An approach for lane assessment and validation during the mapping procedure was introduced.
It could be revealed, that the effective sample size is an indication to resample the particle set, but not sufficient since one has to deal with missing road information in combination with the absence or a wrong GPS signal. Even a particle filter with a high effective particle distribution can drift apart due to missing corrections. Different ways to prevent this behavior, were shown in this thesis. Furthermore, the resampling should be extended to a procedure with history, where a set of strong particles is kept. In combination with positions guess from strong particles subset, the converge of particles to multiple lane segments, and thus modeling

multi-modal density functions for multi-lane streets could be achieved. The direct sampling into multiple parallel lane segments is suitable for crossing situations, but has disadvantages on roads with strong curvature and roundabouts. Furthermore, the occurrence of 'special' situations when moving in urban environment was shown. One of these challenging situations is the lane change, because update and correction can work in antagonistic way. At the moment, when the leaved lane still gets highest mapping score and the new lane position (in adjacent lane) is not yet determined, or a lane model determined from left border of new lane and right border of old lane, the correction step let converge the particle weights back to the lane that is currently leaved. This is contrary to the vehicle motion used for particle updates. The result is, that particles (and weights) and therefore the estimated pose can not follow lane changes beyond this point. To deal with this, such situation must be detected and the corrections temporally suppressed. When the change is complete and the new lane measurement is established, corrections to the particle set can be applied.

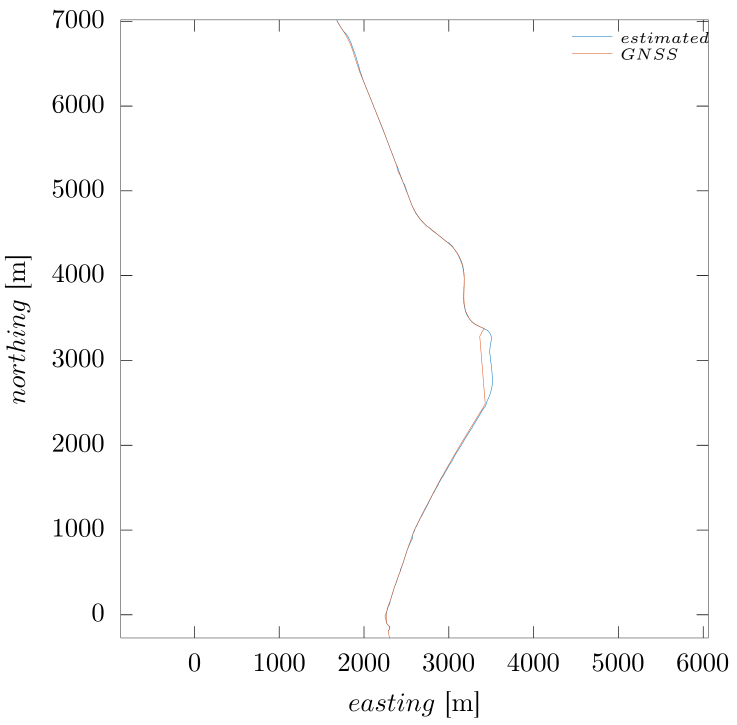


Figure 7.1.: *Estimation and GNSS information at blackout. The figure shows the complete path on second test track. Inside the airport tunnel the GNSS signal get lost (middle).*

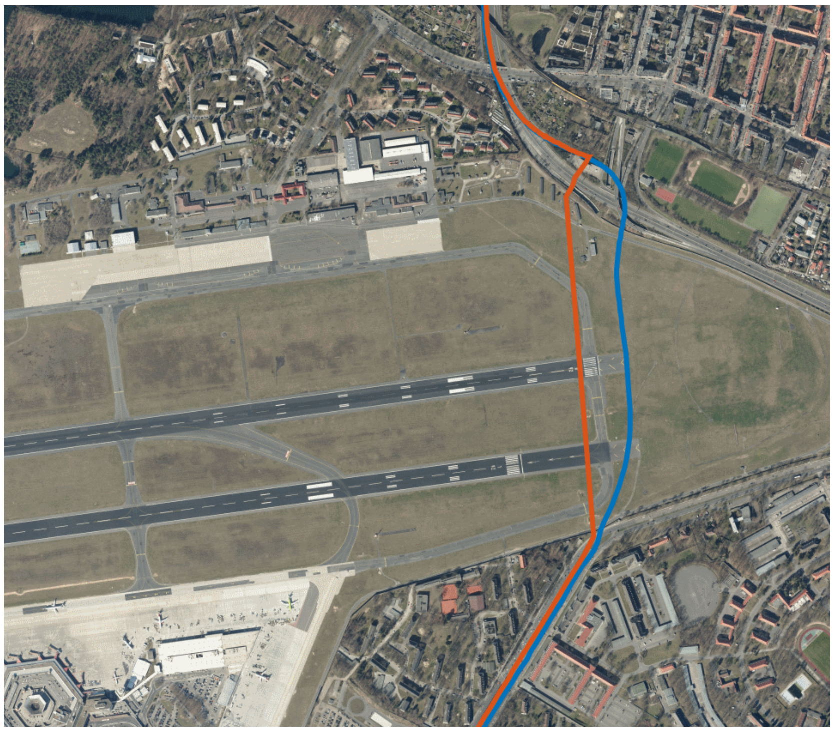


Figure 7.2.: *A detailed view to the path during the GNSS blackout inside the airport tunnel. The estimated path follows the actual path on lane, while GNSS signal shows a jump between tunnel entrance and exit.*

A localization system that relies on sensor data, has always the strengths and weeks of the sensor set within. The introduced system, estimates its relation to the urban environment by the use of landmarks, extracted from cameras. Therefore, it suffers on the same disadvantages as other cameras based systems. Furthermore, the system is prone to the availability of landmarks and additionally to the performance of landmark detection. The following situations can lead to a reduced performance or complete loss:

- challenging illumination (night),
- heavy cast shadows on road surface,
- reduced visibility depending on weather conditions,
- missing or heavy disturbed markings,
- fragile wheel tick based odometry (slippy or bumpy surface),
- and ambivalent kind of road markings (shape, color, classes).

It is essential for the system to perform at least one initial pose guess with the use of a GNSS. The maximum error that can be handled on initialization procedure was investigated and is presented in table 7.3. It has to be remarked, that this evaluation cannot be generalized and depend on local road structure, availability of landmarks and filter parametrization (responsive vs. conservative).

| error | correction success/failed |
|---|---|
| $\pm 25m$ | $\surd$ |
| $\pm 30m$ | $-$ |
| $\measuredangle \pm 45°$ | $\surd$ |
| $\measuredangle \pm 45°$ and $\pm 10m$ | $-$ |

Table 7.3.: *Correction on different initialization errors, applied as offset on initial GNSS pose. The first landmark (pedestrian crossing) was reached after 100 m path length.*

It could be shown, that the introduced positioning system is capable to perform lane level localization and establish a relation between vehicle and street map. Nevertheless, a few issues exist, that should be worked in future. In this section, I will work out the parts of the system, that have the most promising potential for further development.

The first point, is to optimize the computation efforts in the overall system. There are a lot of procedures that must be done with all pixels, all contours, all particles and so on. It is obvious, that greater parts of the processing chain can be parallelized. This applies the detection module as well as the position estimation. With the current implementation, this opportunity is not used. When speaking about parallelization, the use of Graphics Processing Unit (GPU) is suggested, especially if the central processing steps can be reduced to matrix operations. This is often the case. The implementation on a GPU can furthermore enable the use of neural networks to fulfill the task of retrieve street data out of the camera images. While working on this thesis, deep learning and its use for image classification, became state of the art for manifold computer vision tasks.
Having a look at the data fusion, strong efforts are necessary to synchronize the data streams with contours to a unified view on environment. At the transition zone between cameras, one have discontinuities in contours that belong together. These must be dissolved by logical class and geometry checks. With a large number of contours this comes with rapidly increasing computation efforts, because it is needed to compare contours lists to each other and with all its permutations. The idea is, to fuse the data very early at pixel level before the extraction of contours is performed. Synchronizing images (shot at different times), can be achieved with an affine transformation of the IPM images (translation and rotation according to a common time-stamp). If provided, the stitching can be done with sufficient precision and a unified view on road surface around the car can be achieved. The resulting image would be the input for the known contour extraction. The current data fusion module (marking map, 4.3.1) has to perform the assignment of detections between image cycles and the related tracking over time. One must

mention, hat this methods has a catch. There will be small discontinuities, caused by a never perfect camera calibration and vehicle movements like roll or pitch, that cannot be described with camera extrinsics only. Furthermore, one has to ensure that all cameras capture with synchronized exposure or to equalize the intensity differences on images with other methods.

The HOG descriptor was introduced to perform the classification of arrow markings directly from IPM images. The HOG based classifier shows a good performance on precision and recall (high F-score), but it is slow, due to much higher computation efforts to provide the HOG feature vectors. With small restrictions, this process can be accelerated rapidly. In current implementation, every sliding window is threatened as a single image patch and the processing chain for classification is performed for each patch completely. With the step-width being smaller than window size (what is common), many operations are done multiple times for the same image block. An adjustment of the window step corresponding to the HOG cell grid ($8 \times 8$), would allow a single computation of HOG features for the whole image. Then, the sliding window should no longer perform on the image itself, but on the HOG vector grid out of the $8 \times 8$ cells. The aggregated feature vector can directly be used as input for classification. It stays open, if the HOG cell size and the corresponding window size is sufficiently fine granular to achieve the desired detection performance. Otherwise, the cells size have to be adapted. This is paid, with eventually decreasing detector performance, because cell size and shape characteristics of the object to detect must match, to get optimal classification results.

The first point when thinking about further development on estimator, is the use of its output to improve input. After establishing a stable pose output, one has quite a good knowledge about the surrounding map information. For example the lane path is known, where to expect stop lines or other landmarks and furthermore the lane characteristics like multiple lanes, width and even the marking positions. Summarized: one already knows where to expect detectable objects. This makes completely new approaches possible for the detection module and furthermore for modeling the environment. For example, one can get a very precise lane model out of the street map, instead from sole detection by camera without a-priori knowledge. Thus, a more stable and reliable position estimation can be established.

The map data for this thesis is modeled in a way, that lane center corresponds to the track and a given lane width defines the position of lane borders. This width, currently doesnt represent the real width, but the expected - with consideration to the guidelines for lane construction - in Germany. If a lane border is missed by detection or even do not exist, a pre-defined width is used. A better way, but with bigger effort at map creation, would be to define the course of all lane borders separately in the map. This corresponds to the proposals in [40]. The position of the vehicle on the map-lane can then be performed with higher precision at

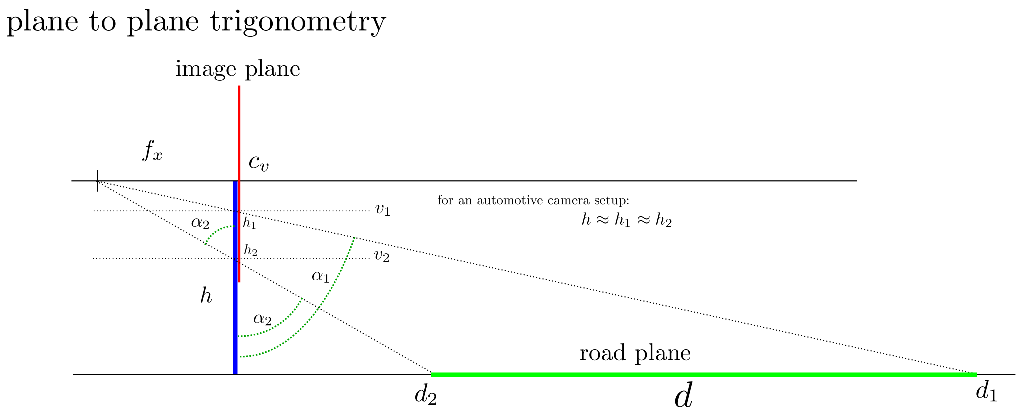
every measurement point. Furthermore, the border type can be used to solve the assignment problem at streets with multiple lanes. Especially, when arrow markings are missing and one has to deal with re-positioning after larger lacks of lane measurements. With lane border information like *solid*, *dashed* or even color, a lane assignment can be performed with much better reliability.
As mentioned in 6 the $\chi^2$-test is defined for normal distributed measurements, but not for multi-modal probability density functions as proposed in my method. I have shown the fragility of the $\chi^2$-method, when evaluating landmarks concerning their relevance and the adaptions that must be applied for its practical use. The next step would be, to find a more reliable and particularly more practically relevant way, to evaluate pose hypotheses with multi-modal probability dense functions.
The selection of landmarks is currently realized with a distance based test. The nearest one wins - even if wrong. That can happen when having a couple of stop lines at large crossings and a current estimated position with high variance. In future work, one would use additional (meta-)information like the lane-relation of landmarks for assessment. Additionally, the results should be weighted and applied regarding the score/cost.For landmarks error modeling, I use semi-static sigmas to create the error co-variance matrices. This is done under consideration of distance based error approximations under specific assumptions. Actually, the measurement variance for each object is known from object tracking and should be used.
In the evaluation chapter (6), a method is presented to generate a ground truth path by the use of IPM and digital areal images. The process is handcrafted. It leads to the approach, using this method as a self-localization procedure. The process chain would be: calculate and synchronize the IPM from each camera and stitch them to one super-IPM image. Try to find a best fitting alignment of the IPM to the areal photo. The result with minimized error is taken as positions guess. The areal photo quality became more and more detailed over the last years, so this seems to be a good opportunity. The images can be requested by worldwide available image servers, what still needs some time nowadays, but can be bridged with use of vehicle motion during download. If the mapping result is unique - what can be expected - one can use a more simple Bayes filter than particle filter, e.g. Extended Kalman Filter (EKF). The disadvantage, is the lack of a clear view on road surface on areal images. This causes problems at dense vegetation, tunnels or other surface covering constructions. Dealing with these 'blackout' situations stays a challenge.

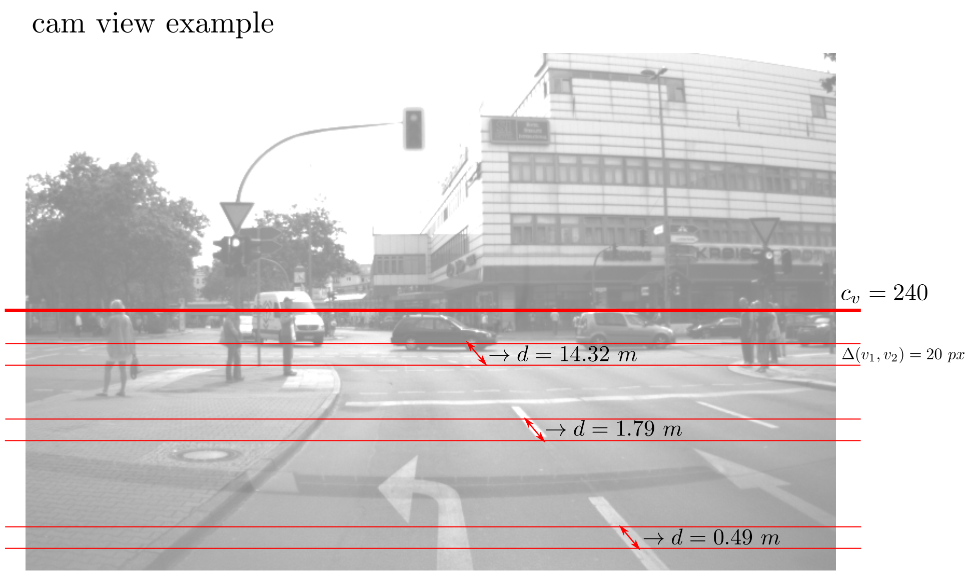
# List of Figures

# List of Tables

# A. Appendix

## A.1. Projection from vehicle camera to ground plane





cam view example
$c_v = 240$
$\Delta(v_1, v_2) = 20\ px$
$\rightarrow d = 14.32\ m$
$\rightarrow d = 1.79\ m$
$\rightarrow d = 0.49\ m$

Figure A.1.: *Ground plane to camera projection. For illustration and the exemplary calculations, the (windshield) camera and it's parameters were used. Equidistant pixel rows (20px) in different image areas give the projected error on ground plane. Hint for reading: a measurement error of 20 px from image in the upper area, cause a measurement error of 14.3 m in car coordinate system.*

## A.2. Lens model equations for fisheye cameras

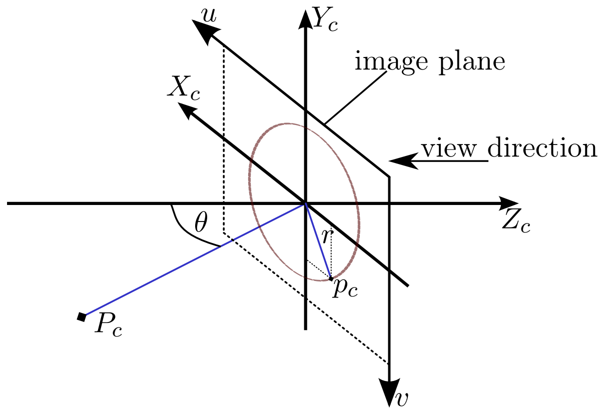


Figure A.2.: *Lens geometry of fisheye camera.*

Figure A.2 illustrates the projection of a point $P_c = (x_c, y_c, z_c)$ from camera coordinate system to image coordinates (pixels) $p = (u, v)$. At first, $P_c$ is projected to the image plane, with $z_c = 0$ in camera coordinate system. This can be done with

$$p_c = \begin{pmatrix} \frac{F(\theta)}{R_{xy}} \cdot x_c \\ \frac{F(\theta)}{R_{xy}} \cdot x_c \end{pmatrix}. \tag{A.1}$$

$F(\theta)$ is the polynomial from equation 3.6 and $R_{xy} = \sqrt{x_c^2 + y_c^2}$. The transformation to image coordinates can be performed with inversion of $Y$ component and shifting via cameras principle point ($c_x$ and $c_y$) to the upper left image corner. Thus, there is the equation

$$p = (u, v)^T = \begin{pmatrix} \frac{F(\theta)}{R_{xy}} \cdot x_c + c_x \\ (-1) \cdot \frac{F(\theta)}{R_{xy}} \cdot y_c + c_y \\ 0 \end{pmatrix}. \tag{A.2}$$

For an extrinsic calibration, partial derivations are required from the projection model to the corresponding extrinsic parameters in $\vec{t}$ and $\vec{r}$. The vector $\vec{t} = (x, y, z)$ describes the translation (shift) to the reference coordinate system and $\vec{r} = (r_1, r_2, r_3)$ the rodrigues vector, a compact description for spatial rotations of vectors in $\mathbb{R}^3$.

$$\frac{\partial p}{\partial \vec{t}} = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \end{pmatrix} \text{ und } \frac{\partial p}{\partial \vec{r}} = \begin{pmatrix} \frac{\partial u}{\partial r_1} & \frac{\partial u}{\partial r_2} & \frac{\partial u}{\partial r_3} \\ \frac{\partial v}{\partial r_1} & \frac{\partial v}{\partial r_2} & \frac{\partial v}{\partial r_3} \end{pmatrix}$$

All derivations are based on the partial derivations of the camera coordinates ($\frac{\partial x_c}{\partial x}, \frac{\partial x_c}{\partial y}$...). For example, let perform this on $\frac{\partial u}{\partial x}$. The remaining derivations can be determined congruently, with the flipped sign for $v$ component in consideration.

$$\frac{\partial u}{\partial x} = \frac{\partial \left( \frac{F(\theta)}{R_{xy}} \cdot x_c + c_x \right)}{\partial x} = \frac{\partial \frac{F(\theta)}{R_{xy}}}{\partial x} \cdot x_c + \frac{F(\theta)}{R_{xy}} \cdot \frac{\partial x_c}{\partial x}$$

$c_x$ is constant and can be neglected. Applying products rule on first term and having the second addend consisting of known variables. Now applying quotients rule to first addend yields to

$$\frac{\partial \frac{F(\theta)}{R_{xy}}}{\partial x} = \frac{x_c}{R_{xy}^2} \left[ \frac{\partial F(\theta)}{\partial x} \cdot R_{xy} - F(\theta) \cdot \frac{\partial R_{xy}}{\partial x} \right].$$

Furthermore, this leads to

$$\frac{\partial F(\theta)}{\partial x} = \frac{\partial \left( a_1\theta + a_2\theta^2 + a_3\theta^3 + a_4\theta^4 \right)}{\partial x} = \frac{\partial a_1\theta}{\partial x} + \frac{\partial a_2\theta^2}{\partial x} + \frac{\partial a_3\theta^3}{\partial x} + \frac{\partial a_4\theta^4}{\partial x}$$

$$\frac{\partial a_n\theta^n}{\partial X} = a_n \cdot (n-1) \cdot \theta^{n-1} \cdot \frac{\partial \theta}{\partial x}$$

$$\frac{\partial \theta}{\partial x} = \frac{\partial \arcsin \frac{z_c}{R_{xyz}}}{\partial x} = \frac{1}{\sqrt{1 - \frac{z_c}{R_{xyz}}}} \cdot \frac{\frac{z_c}{R_{xyz}}}{x} = \frac{1}{\sqrt{1 - \frac{z_c}{R_{xyz}}}} \cdot \frac{\frac{\partial z_c}{\partial x} \cdot R_{xyz} - z_c \cdot \frac{\partial R_{xyz}}{\partial x}}{R_{xyz}^2}$$

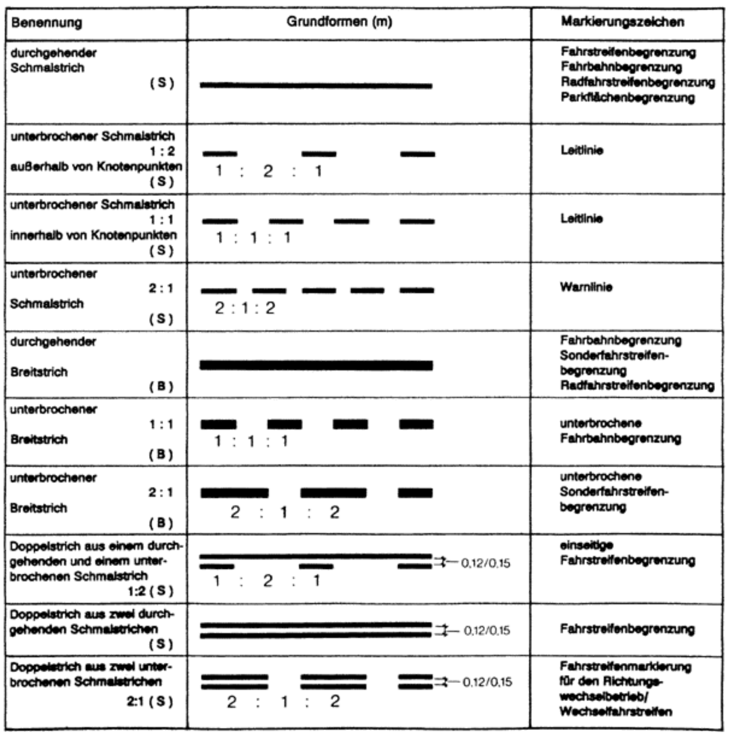
$$\frac{\partial R_{xyz}}{\partial x} = \frac{\partial \sqrt{x_c K^2 + y_c^2 + z_c^2}}{\partial x} = \frac{1}{2} \frac{\frac{\partial x_c^2}{\partial x} + \frac{\partial y_c^2}{\partial x} + \frac{\partial z_c^2}{\partial x}}{\sqrt{x_c^2 + y_c^2 + z_c^2}}$$

$$\frac{\partial x_c^2}{\partial x} = 2x_c \frac{\partial x_c}{\partial x} \quad \text{as well for } y_c \text{ and } z_c$$

$R_{xy}$ can be determined, corresponding to $R_{xyz}$, but without $z_c$ therms.

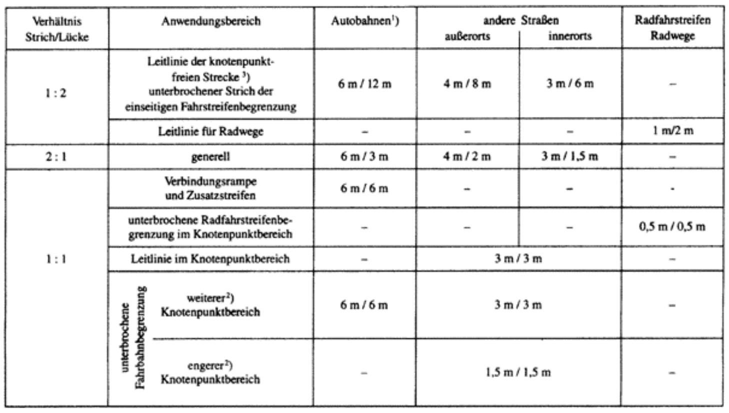
## A.3. Extractions from "Richtlinien für Markierung von Straßen (RMS)" (Guidelines for German road markings)

Table A.1.: *Markings according to RMS [41]*



| Benennung | Grundformen (m) | Markierungszeichen |
|---|---|---|
| durchgehender Schmalstrich (S) | | Fahrstreifenbegrenzung Fahrbahnbegrenzung Radfahrstreifenbegrenzung Parkflächenbegrenzung |
| unterbrochener Schmalstrich 1 : 2 außerhalb von Knotenpunkten (S) | 1 : 2 : 1 | Leitlinie |
| unterbrochener Schmalstrich 1 : 1 innerhalb von Knotenpunkten (S) | 1 : 1 : 1 | Leitlinie |
| unterbrochener 2 : 1 Schmalstrich (S) | 2 : 1 : 2 | Warnlinie |
| durchgehender Breitstrich (B) | | Fahrbahnbegrenzung Sonderfahrstreifen-begrenzung Radfahrstreifenbegrenzung |
| unterbrochener 1 : 1 Breitstrich (B) | 1 : 1 : 1 | unterbrochene Fahrbahnbegrenzung |
| unterbrochener 2 : 1 Breitstrich (B) | 2 : 1 : 2 | unterbrochene Sonderfahrstreifen-begrenzung |
| Doppelstrich aus einem durch-gehenden und einem unter-brochenen Schmalstrich 1:2 (S) | 0,12/0,15<br>1 : 2 : 1 | einseitige Fahrstreifenbegrenzung |
| Doppelstrich aus zwei durch-gehenden Schmalstrichen (S) | 0,12/0,15 | Fahrstreifenbegrenzung |
| Doppelstrich aus zwei unter-brochenen Schmalstrichen 2:1 (S) | 0,12/0,15<br>2 : 1 : 2 | Fahrstreifenmarkierung für den Richtungs-wechselbetrieb/ Wechselfahrstreifen |

Table A.2.: *Line lengths according RMS [42]*



| Verhältnis Strich/Lücke | Anwendungsbereich | | Autobahnen¹) | andere Straßen | | Radfahrstreifen Radwege |
|---|---|---|---|---|---|---|
| | | | | außerorts | innerorts | |
| 1 : 2 | Leitlinie der knotenpunkt-freien Strecke ³) unterbrochener Strich der einseitigen Fahrstreifenbegrenzung | | 6 m / 12 m | 4 m / 8 m | 3 m / 6 m | – |
| | Leitlinie für Radwege | | – | – | – | 1 m/2 m |
| 2 : 1 | generell | | 6 m / 3 m | 4 m / 2 m | 3 m / 1,5 m | – |
| 1 : 1 | Verbindungsrampe und Zusatzstreifen | | 6 m / 6 m | – | – | - |
| | unterbrochene Radfahrstreifenbe-grenzung im Knotenpunktbereich | | – | – | – | 0,5 m / 0,5 m |
| | Leitlinie im Knotenpunktbereich | | – | 3 m / 3 m | | – |
| | unterbrochene Fahrbahnbegrenzung | weiterer²) Knotenpunktbereich | 6 m / 6 m | 3 m / 3 m | | – |
| | | engerer²) Knotenpunktbereich | – | 1,5 m / 1,5 m | | – |

## A.4. Classifier for road markings

### A.4.1. Support Vector Machines

The SVM is a classification method that is based on statistical learning. The information provided in this section is oriented on [47]. The interested reader is referenced to this book for a very detailed and comprehensible view into SVMs.

At the end of this section the reader will have an understanding about the mapping of inputs to responses (prediction) and how the learning process of a SVM is mathematically defined.
For example, in the given case a relation between an input vector $x$ (contour features) and its response $y$ (class) should be found. The function $f(x)$ describes this relation and is called *prediction*. Often it is not possible to find a functional relationship between an input and a desired response. To resolve this, statistical learning or machine leaning principles are used for approximation. A finite set of inputs and its corresponding responses must be known - the training set. Depending on the complexity of the mapping problem the function $f(x)$ will cause errors. The goal of the SVM learning is to find a good approximation for $f(x)$, that maps the response $y$ to an arbitrary $x$ with minimal error. To evaluate the mapping quality a *loss function* $L = (x, y, f(x))$ is used. It is obvious that the average *loss* for the whole training set should be small.

$$\frac{1}{m-n}\sum_{i=n+1}^{m} L\left(x_i, y_i, f(x_i)\right) \tag{A.3}$$

The learning process of a SVM does not generally produces a good approximation for $f(x)$, even if it minimizes $L$ on a closed training set $D := ((x_1, y_1), ..., (x_n, y_n))$ up to 0. This phenomenon is called *overfitting*. The result of $f(x)$ makes a minimal error on $D$ but a poor classification performance on future (unknown) data. A common way to resolve this, is to increase the training set to an appropriate size.
Considering a binary classification problem with responses $y$ from space $Y = \{-1, +1\}$ and the inputs $x$ from space $X$ out of the Euclidean space $\mathbb{R}^d$. Additionally a closed training set $D := ((x_1, y_1), ..., (x_n, y_n))$ is assumed, for which an element $w \in \mathbb{R}^d$ with $\|w\|_2 = 1$ and a real number $b \in \mathbb{R}$ exist such that

$$\begin{aligned} \langle w, x_i\rangle + b &> 0, \forall \{i | y_i = +1\} \\ \langle w, x_i\rangle + b &< 0, \forall \{i | y_i = -1\}\,. \end{aligned} \tag{A.4}$$

Thus, $w$ and $b$ describe a *hyperplane* that separates the training set $D$ into the two groups $\{(x_i, y_i) \in D : y_i = +1\}$ and $\{(x_i, y_i) \in D : y_i = -1\}$ with the decision function

$$f(x) := sign(\langle w, x\rangle + b). \tag{A.5}$$

The minimization problem becomes

$$\begin{aligned} &\text{minimize } \langle w, w\rangle && \text{over } w \in \mathbb{R}, b \in \mathbb{R} \\ &\text{subject to } y_i(\langle w, x_i\rangle + b) \geq 1 && i = 1, ..., n. \end{aligned} \tag{A.6}$$

The approach has two issues:

1. A linear form of the decision function may not be suitable for the classification task. The given training set $D$ cannot be separated linearly.

2. In the presence of noise, it can be possible to misclassify a few training points in order to avoid overfitting.

The first issue can be dealed, with mapping the input data $(x_1, ..., x_n)$ into a multi-dimensional space, called Hilbert space. This procedure is sometimes called *kernel-trick* and an appropriate method to handle non-linear-separable input data with linear classifiers. It is typically a non-linear map $\Phi : X \to H_0$. The mapped data set $(\Phi(x_i), y_i, ..., \Phi(x_n), y_n)$ is then applied in $H_0$ instead of $X$. The $x$ from (A.5) and the $x_i$ from (A.6), are replaced with $\Phi(x)$ and $\Phi(x_i)$ receptively. The hyperplane component $w$ from (A.6) is now chosen from the Hilbert space $H_0$.
With this knowledge it is now possible to define a kernel.
*When $X$ is a non-empty set, the function $k : X \times X \to K$ is called a kernel on $X$, if the K-Hilbert space $H$ and the map $\Phi : X \to H$ exists. And furthermore*

$$k(x, x') = \left\langle \Phi(x), \Phi(x') \right\rangle ; \forall x, x' \in X \tag{A.7}$$

*is given. $\Phi$ is called a feature map and $H$ a feature space of $k$.*
Next to the linear kernels a set of more flexible feature maps $\Phi$ exists. For example the *RBF* is one of the mostly used kernel types in practice. Each kernel has different classification behavior when applied to a training set and comes with different calculation complexity.
The second issue is addressed with relaxing the constraints from (A.4) by allowing errors on the training set $D$, with requiring only that $(w, b)$ satisfy $y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i$ for some so-called *slack variables* $\xi_i \geq 0$. This method is called *soft margin support vector machine*, Figure A.3. The summed up quadratic minimization problem then, is

$$\begin{aligned} &\text{minimize} \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{n}\xi_i && \text{for } w \in H_0, b \in \mathbb{R}, \xi \in \mathbb{R}^n \\ &\text{subject to } y_i(\langle w, \Phi(x_i)\rangle + b) \geq 1 - \xi_i, && i = 1, ..., n \\ &\qquad \xi_i \geq 0, && i = 1, ..., n. \end{aligned} \tag{A.8}$$

$C > 0$ is a free but fixed parameter to balance the first term of the objective function (A.8) with the second. Thus, it becomes possible to penalize not only wrong classifications (predictions to a wrong half space of the hyperplane), but margin errors for $x$ lying inside the tube around the hyperplane.
With the SVM defined in this section it is possible to have an optimal solution for the mapping problem (map contour feature to class labels). Additionally, there are parameters to balance the classification behavior of the SVM. For example, it

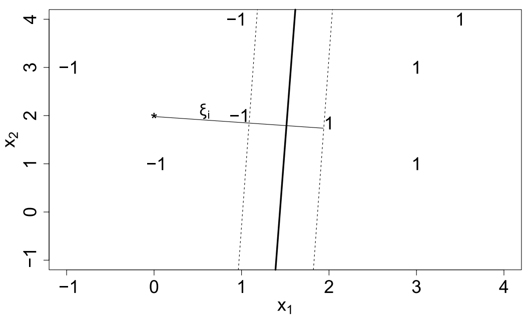


Figure A.3.: *Illustration of the soft margin SVM and it's slack variable $\xi_i$, [47]*

.

might be necessary to accept lower detection rates on some input data, while minimizing false positive classification. The last sentence implies one of the greatest drawbacks, when using SVM. Often it is difficult to find a good parameter set and a kernel that solves the classification problem perfectly. Moreover, the optimal hyperplane might not be found. In order to minimize this risk, a well selected closed set of features with minimized inter-class correlation is needed. In other words, the descriptive score of a feature should be high for each class In section 4.2.1 methods for optimal feature selection are shown. Another disadvantage is, that the inputs must be normalized. Not only the training set, but future inputs. Without knowing the future value range, the generated norm vector can only be derived from the training set. Normalizing future inputs with the norm vector obtained during training, will obviously lead to errors.
There are machine learning algorithms, that have advantages especially on these issues. One of them is the Random Forest classifier, that is introduced in the next section.

## A.4.2. Random Forest

The Random Forest is a machine learning method based on the principles of decision trees. This section strongly orientates on chapter 5 of the collection *Ensemble machine learning*, [8]. But it is more focused on using Random Forests for classification, compared to the original text that handles regression with same extent.
Random Forest were introduced by Leo Breiman in [6] as a competitor to *boosting* [1]. His work was inspired by [1]. Random Forests can be used to generate categorical responses (classification) or a continuous response (regression). From a computational point of view, Random Forest look attractive because they

- are relatively fast to train and to predict
- have small set of tuning parameters

[1]Boosting is a method of supervised learning with the idea, that a set of weak learners can be ensembled to a strong one.

- have built in estimate of generalization error
- are easy to parallelize.

Statistically, Random Forests provide features, such as

- measure of variable importance
- applying class weights for unbalanced training data
- unsupervised learning.

Especially the first point might be very useful, when having a set of features without knowing the impact of them for prediction. The feature selection 4.2.1 could be done native during training. Furthermore, the features have not to be normalized. Rather it helps for categorization to leave the features at their origin values.
Obviously the name suggests, that Random Forests is a tree-based ensemble, with each tree depending on a collection of random variables. Remember section A.4.1: the goal of a classifier is to find a mapping from a random valued input vector $x$ to a random variable $y$, representing the class. The process is called prediction and the training should find a good approximation for the prediction function (or mapping function) $f(x)$. The quality of the found function $f(x)$ can be measured with the average loss over all inputs.

$$E_{x,y}(L(y, f(x)) = \frac{1}{m-n} \sum_{i=n+1}^{m} L\left(x_i, y_i, f(x_i)\right) \tag{A.9}$$

The loss function $L$ is a measure how close $f(x_i)$ is to $y_i$; it penalizes results, that are fare away from corresponding $y$. Typical choice of $L$ is the *squared error loss* $L(y, f(x)) = (y - f(x))^2$ for regression and the *zero-one-loss* for classification.

$$L = (y_i, f(x_i)) = \begin{cases} 0 & \text{if } y_i = f(x_i) \\ 1 & \text{otherwise} \end{cases} \tag{A.10}$$

This corresponds to the hyperplane idea in (A.4). In case of classification, if the set of possible values of $y$ is denoted by $\vartheta$, minimizing the average loss $E_{x,y}(L(y, f(x))$ for zero-one loss method or in other words - maximizing the probability $P$ for correct classification on training set of size $n$ - gives

$$f(x) = \arg\max\ P(y = y_i | x = x_i). \quad \text{for } y \in \vartheta; i = 1, ..., n \tag{A.11}$$

In Random Forests the function $f$ is constructed as a collection of trees. Each tree itself is a predictor, a so-called *base learner.* The collection of all learners $h_1(x), ..., h_n(x)$ gives the *ensembled* predictor $f(x)$. In case of regression, the learners are averaged

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} h_i(x), \tag{A.12}$$

while in case of classification, $f(x)$ is the most frequently predicted class (*voting*).

$$f(x) = \arg\max \sum_{i=1}^{n} I(y = h_i(x)). \quad y \in \vartheta \tag{A.13}$$

Every base learner is a tree denoted $h_i(x, \Theta_i)$, where $\Theta_i$ is a collection of random variables and independent for $i = 1, ..., n$.
Trees used in Random Forest are based on binary partitioning trees. These trees divide the predictor space using a sequence of binary partitions (splits) on individual variables. The root node of the tree comprises the entire predictor space. Nodes that are not split are called *terminal nodes* and form the final partition of the predictor space. According to the value of the predictor variable, each node split into two descendant branches. Generally, the split is based on "smaller than - greater than" decisions. A categorical predictor variable $X_i$ takes values from a finite set of classes $S_i = \{s_{i,1}, ..., s_{i,m}\}$. The split sends a subset of these classes to the left and the remaining classes to the right (binary split), Figure A.4. The particular split the tree uses to partition a node into two descendants is chosen by considering every possible split on every predictor variable and choosing the "best" according to some criterion. In classification context where there are $K$ classes denoted $1, ...K$, a typical split-criterion is the *Gini index.*

$$Q = \sum_{k \neq k'}^{K} \hat{p}_k \hat{p}_{k'}, \tag{A.14}$$

Where $\hat{p}_k$ is the proportion of class $k$ observations in the node:

$$\hat{p}_k = \frac{1}{n} \sum_{i=1}^{n} I(y_i = k). \tag{A.15}$$

The splitting criterion gives a measure of purity for a node. Large values representing an impure node.
Once a split has been selected, the data is partitioned into the two child nodes and each of these nodes is treated in the same way as the parent node. This procedure continues recursively until a stop criterion is met. When the stop criterion is met, the split stops and the unsplit node is called *terminal node.* The predicted value in the terminal node is then obtained for all observations by computing the most frequent class.
The randomness is injected implicitly in two ways. Remember, every base learner is a tree $h_i(x, \Theta_i)$, where $\Theta$ is a collection of random variables. Then the random component $\theta_i$ as a particular realization of $\Theta_i$. It is used to implicitly inject the randomness. First, each tree is fit to an independent bootstrap sample from the original data (training set). This bootstrap sampling is randomized and thus the first part of $\theta_i$. Second, when splitting a node, the best split is found over a randomly selected subset of $m$ predictor variables instead of all $p$ predictors, independently at each node. The randomization used to sample the predictors gives

the second part of $\theta$. As mentioned before, the final prediction (category) of the forest, is an unweighted voting over all trees.

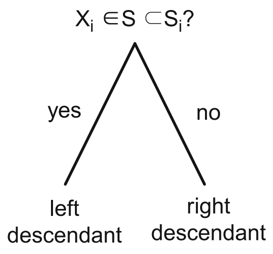


Figure A.4.: *Binary recursive partitioning tree. Splitting on a categorical predictor variable in $X_i$ using subset $S \subset S_i$.*

## A.5. Cholesky factorization for discretization of a multivariate normal distribution

When modeling stochastic processes, typically noise is applied to the process equation in form of an error-element $e$. For implementation, the error element is a function value out of a density function $p(e)$, that describes the probability in the error-space. For this work, the function corresponds to a multivariate normal distribution, defined with *mean* $\mu$ and the covariance matrix $\Sigma$. An efficient way to generate such discrete samples out of distributions is to transform samples from an unidimensional independent identically distributed (iid) sequence [16]. The probability of existence for all values in this sequence is equal and the generation process for them is independent from each other.
If the covariance matrix is positive definite, the Cholesky factorization can be used to perform the transformation of the iid sequence. A $n \times n$ matrix $M$ is positive definite if $x^T M x > 0$ is given for all $n$ column-vectors $x$ with $x \neq 0$. The transformation itself is finally given with

$$X = A\,Y + \mu, \tag{A.16}$$

where $A$ is the Cholesky factorization of $\Sigma$ an $Y$ is the vector containing the iid sequence (unidimensional, idependent and normal distributed) [15]. The resulting vector $X$ is a sample of the multivariate normal distribution, defined by mean $\mu$ and covariance matrix $\Sigma$.
In case the positive definite property is not given, the eigenvalues and eigenvectors of the covariance matrix $\Sigma$ have to be determined. The transformation can then be performed via the matrix $Q = \Lambda^{\frac{1}{2}}\ \Phi$ instead of $A$ from (A.16). $\Phi$ are the column wise normalized eigenvectors of $\Sigma$ and $\Lambda$ is the diagonal matrix of the eigenvalues of $\Sigma$ [22].

## A.6. Results from image processing

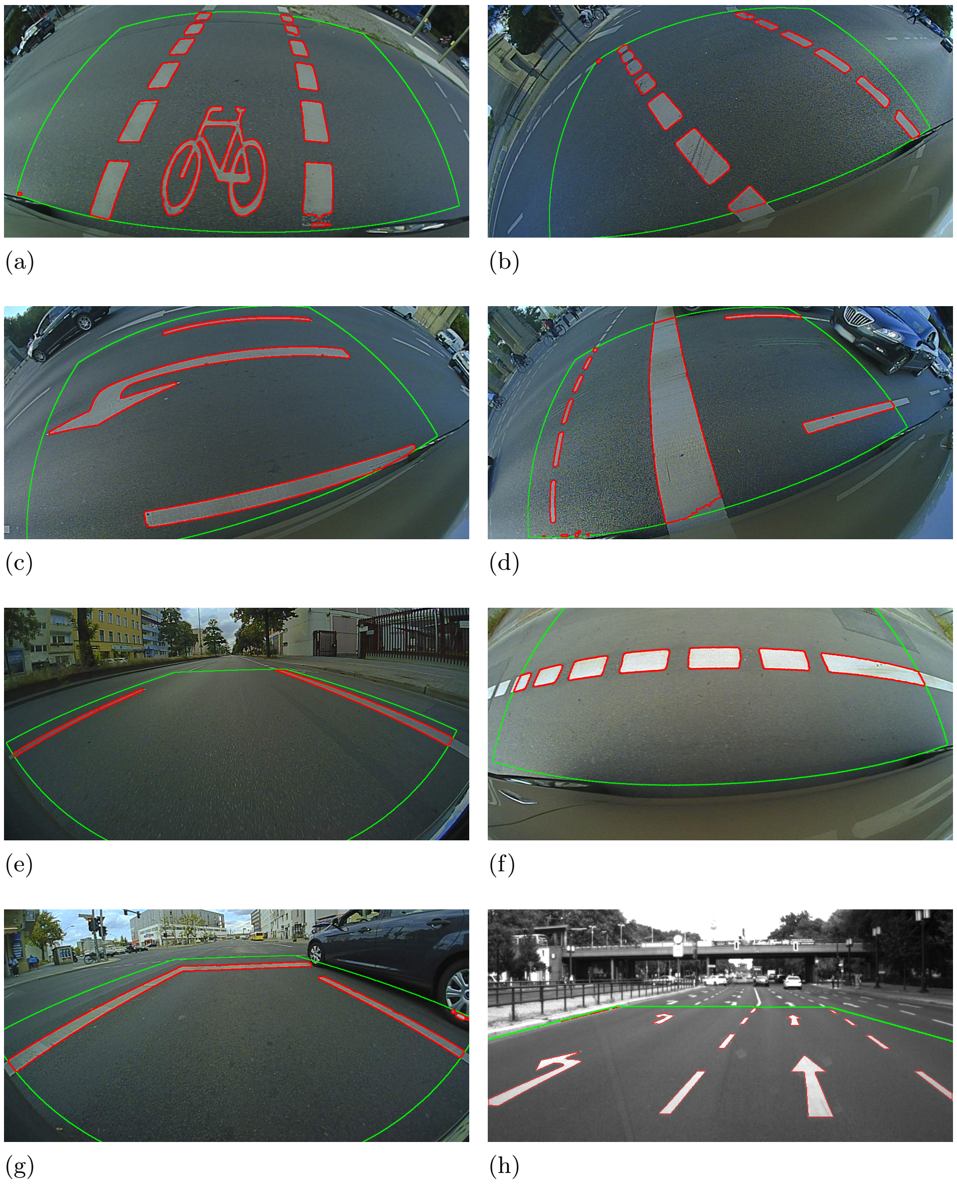


Figure A.5.: *Marking detection on ideal markings.*

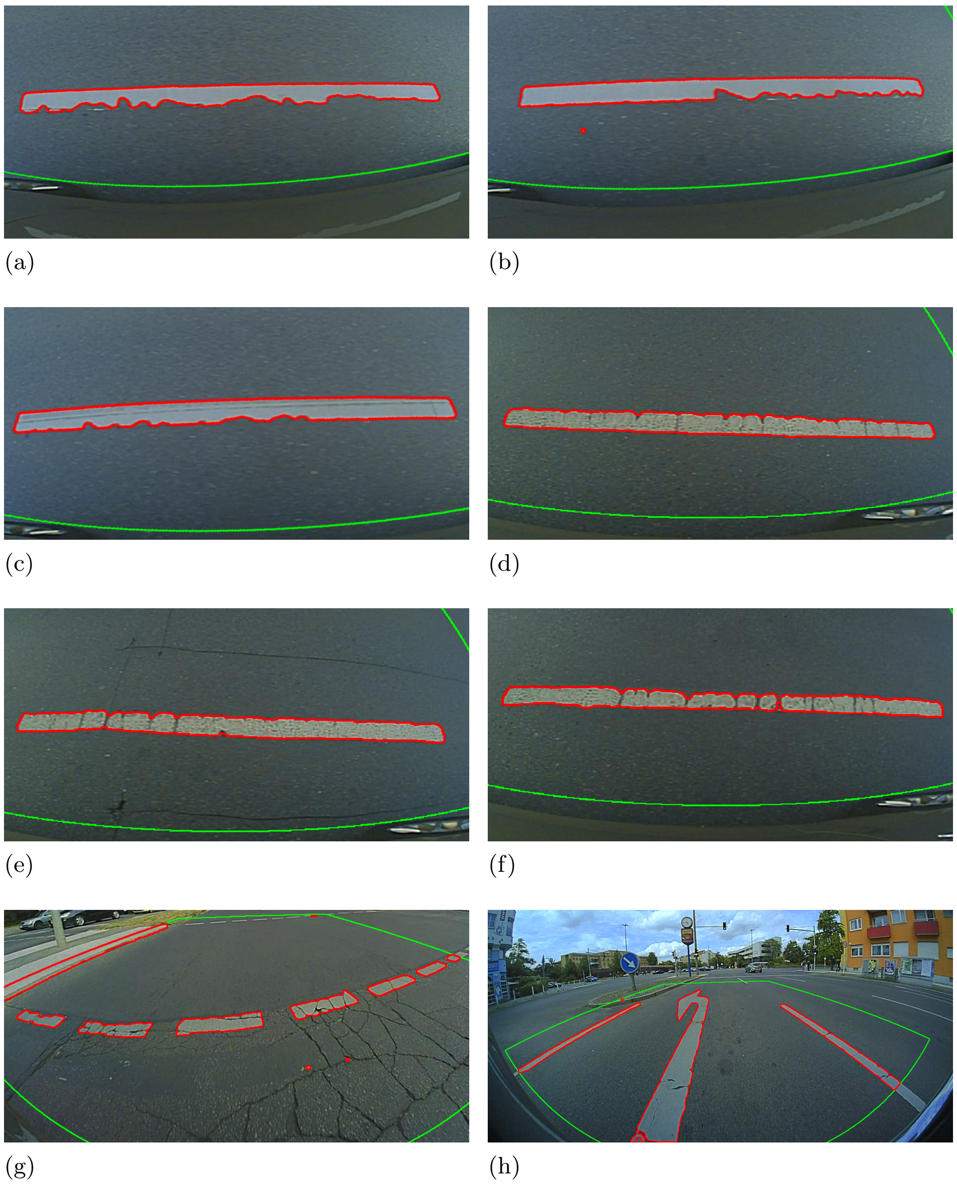
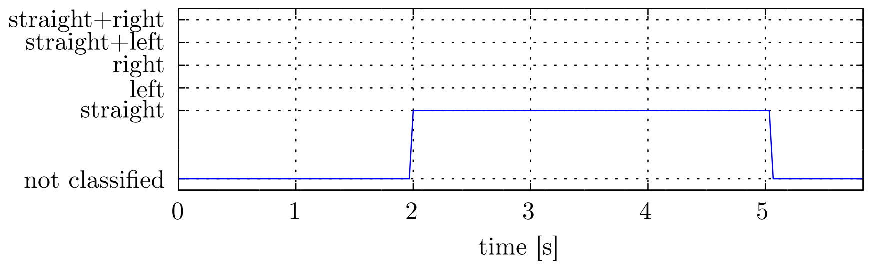
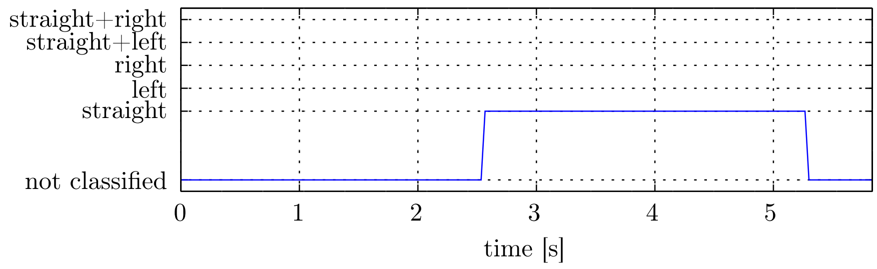


Figure A.6.: *Marking detection on non-ideal markings.*

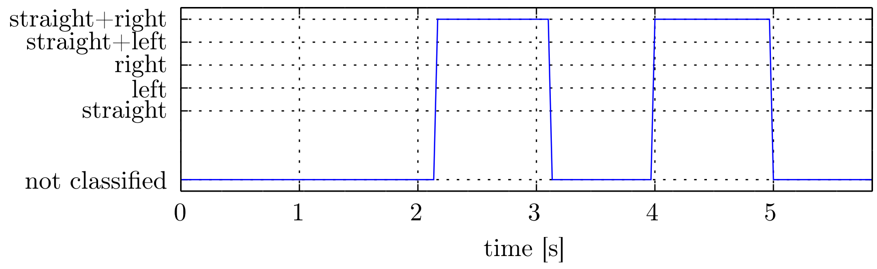
## A.7. Results from lane type classification



(a) *Left adjacent lane*

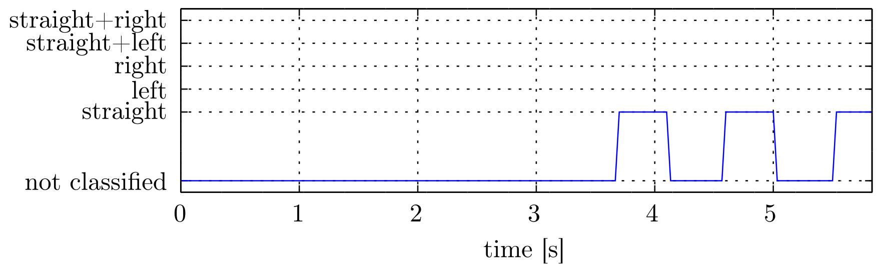


(b) *Ego-lane*

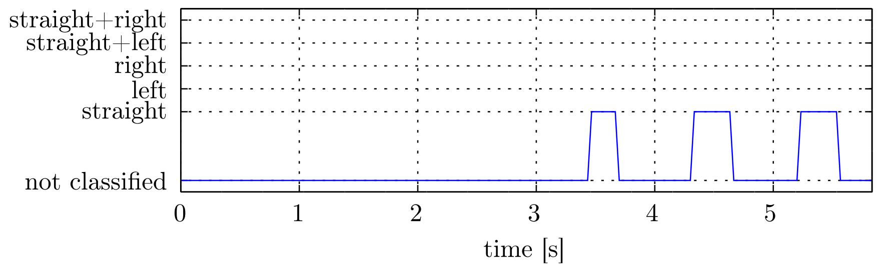


(c) *Right adjacent lane*

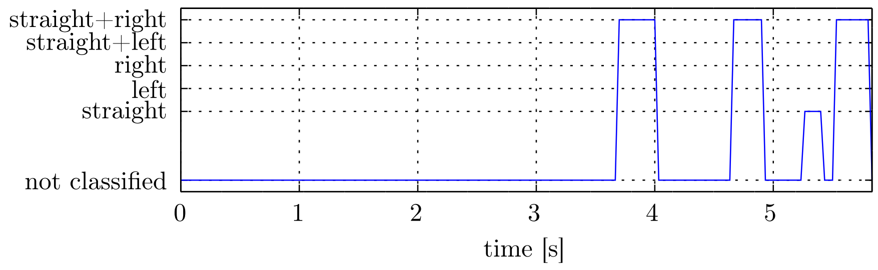
Figure A.7.: *Evaluation of lane type classification, using front camera only.*



(a) *Left adjacent lane*



(b) *Ego-lane*

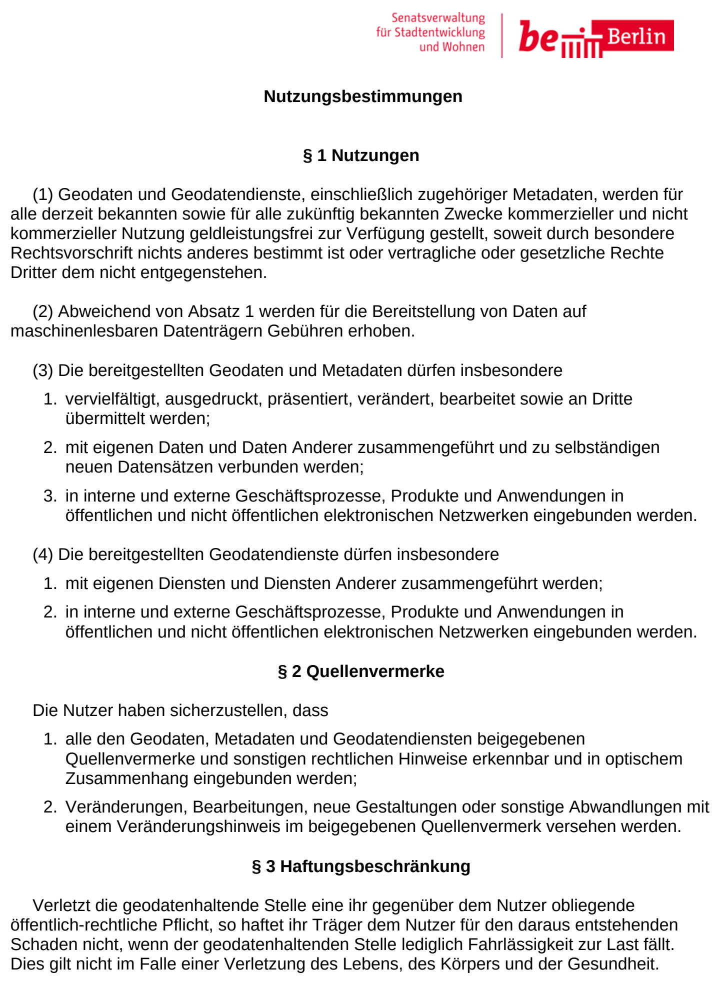


(c) *Right adjacent lane*

Figure A.8.: *Evaluation of lane type classification, using fisheye cameras only.*

## A.8. Terms of use for data sets from Geoportal Berlin

[5]



Senatsverwaltung
für Stadtentwicklung
und Wohnen

Berlin

**Nutzungsbestimmungen**

**§ 1 Nutzungen**

(1) Geodaten und Geodatendienste, einschließlich zugehöriger Metadaten, werden für alle derzeit bekannten sowie für alle zukünftig bekannten Zwecke kommerzieller und nicht kommerzieller Nutzung geldleistungsfrei zur Verfügung gestellt, soweit durch besondere Rechtsvorschrift nichts anderes bestimmt ist oder vertragliche oder gesetzliche Rechte Dritter dem nicht entgegenstehen.

(2) Abweichend von Absatz 1 werden für die Bereitstellung von Daten auf maschinenlesbaren Datenträgern Gebühren erhoben.

(3) Die bereitgestellten Geodaten und Metadaten dürfen insbesondere

1. vervielfältigt, ausgedruckt, präsentiert, verändert, bearbeitet sowie an Dritte übermittelt werden;
2. mit eigenen Daten und Daten Anderer zusammengeführt und zu selbständigen neuen Datensätzen verbunden werden;
3. in interne und externe Geschäftsprozesse, Produkte und Anwendungen in öffentlichen und nicht öffentlichen elektronischen Netzwerken eingebunden werden.

(4) Die bereitgestellten Geodatendienste dürfen insbesondere

1. mit eigenen Diensten und Diensten Anderer zusammengeführt werden;
2. in interne und externe Geschäftsprozesse, Produkte und Anwendungen in öffentlichen und nicht öffentlichen elektronischen Netzwerken eingebunden werden.

**§ 2 Quellenvermerke**

Die Nutzer haben sicherzustellen, dass

1. alle den Geodaten, Metadaten und Geodatendiensten beigegebenen Quellenvermerke und sonstigen rechtlichen Hinweise erkennbar und in optischem Zusammenhang eingebunden werden;
2. Veränderungen, Bearbeitungen, neue Gestaltungen oder sonstige Abwandlungen mit einem Veränderungshinweis im beigegebenen Quellenvermerk versehen werden.

**§ 3 Haftungsbeschränkung**

Verletzt die geodatenhaltende Stelle eine ihr gegenüber dem Nutzer obliegende öffentlich-rechtliche Pflicht, so haftet ihr Träger dem Nutzer für den daraus entstehenden Schaden nicht, wenn der geodatenhaltenden Stelle lediglich Fahrlässigkeit zur Last fällt. Dies gilt nicht im Falle einer Verletzung des Lebens, des Körpers und der Gesundheit.

# Bibliography

[1] Y. Amit and D. Geman. *Shape quantization and recognition with randomized trees.*, chapter 9. Massachusetts Institute of Technology, 1997.

[2] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, Feb 2002.

[3] N. M. E. Association. Nmea 0183 - standard for interfacing marine electronic devices, 2012.

[4] A. Basu and S. Licardie. Alternative models for fish-eye lenses. *Pattern Recognition Letters*, 16:433–441, 1995.

[5] G. Berlin. Digitale farbige orthophotos 2018 (dop20rgb). "`https://www.stadtentwicklung.berlin.de/geoinformation/fis-broker/index.shtml`", 2019.

[6] L. Breiman. Bagging predictors. pages 123–140, 2001.

[7] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8:679–698, 1986.

[8] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[9] Z. Chen. Bayesian filtering: From kalman filters to particle filters, and beyond. *McMaster University: Hamilton, Ontario, Canada*, pages 1–69, 2003.

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.

[11] E. D. Dickmanns. *Dynamic vision for perception and control of motion.* Springer, 2007.

[12] DIN Deutsches Institut für Normung e.V. Straßenfahrzeuge - fahrzeugdynamik und fahrverhalten - begriffe (iso 8855:2011), 2011.

[13] A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[14] J. D. Foley. *Computer graphics.* Addison-Wesley systems programming series. Addison-Wesley, 2nd ed edition, 1990.

[15] J. E. Gentle. Matrix transformations and factorizations. In *Matrix Algebra: Theory, Computations, and Applications in Statistics*, Springer Texts in Statistics, pages 173–200. Springer New York, 2007.

[16] J. E. Gentle. Generation of random numbers. In *Computational Statistics*, Statistics and Computing, pages 305–331. Springer New York, 2009.

[17] R. C. Gonzalez and R. E. Woods. *Digital image processing.* Prentice Hall, 2nd ed edition, 2002.

[18] G. Green. *An essay on the application of mathematical analysis to the theories of electricity and magnetism.* T. Wheelhouse, Nottingham, 1828.

[19] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2432–2437, April 2005.

[20] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, 1997.

[21] M. Hentschel and B. Wagner. Autonomous robot navigation based on openstreetmap geodata. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1645–1650, Sept 2010.

[22] I. Hernádvölgyi. Generating random vectors from the multivariate normal distribution. *University of Ottawa: Ottawa, Ontario, Canada*, pages 1–14, 1998.

[23] B. Jähne. *Digitale Bildverarbeitung.* Springer, 2005.

[24] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82:35–45, 1960.

[25] J. Kannala and S. Brandt. A generic camera calibration method for fisheye lenses. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*, pages 10–13 Vol.1.

[26] M. Kochem. *Ein Fahrerassistenzsystem zur Unterstützung des rückwärtigen Parkvorgangs für PKW.* PhD thesis, TU Darmstadt, Düsseldorf, Januar 2005.

[27] A. Kong, J. S. Liu, and W. H. Wong. Sequential imputations and bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288, 1994.

[28] C.-H. Kum, D.-C. Cho, M.-S. Ra, and W.-Y. Kim. Lane detection system with around view monitoring for intelligent vehicle. In *2013 International Soc Design Conference (ISOCC)*, pages 215–218, 2013.

[29] LVermGeo. Utm-zoneneinteilung. Landesamt für Vermessung und Geoinformation Sachsen-Anhalt, `https://www.lvermgeo.sachsen-anhalt.de/MediaLibrary/Content/de/veroeffentlichungen/news/archiv/2008/lsa_utmk.jpg`, 2008. Stand: 09.07.2015, 12:05 Uhr.

[30] G. Maier, S. Pangerl, and A. Schindler. Realtime detection and classification of arrow markings using curve-based prototype fitting. In *Proceedings of IEEE Intelligent Vehicles Symposium*, pages 442–447, 2011.

[31] R. McConnell. Method of and apparatus for pattern recognition, Jan. 28 1986. US Patent 4,567,610.

[32] I. Miller, M. Campbell, and D. Huttenlocher. Map-aided localization in sparse global positioning system environments using vision and particle filtering. *Journal of Field Robotics*, 28(5):619–643, 2011.

[33] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IEEE Transactions on Information Theory*, 8:179–187, 1962.

[34] S. Nedevschi and R. Danescu. Detection and classification of painted road objects for intersection assistance applications. In *13th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 433–438, 2010.

[35] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org`, 2019.

[36] F. Philipp, H. H. Tadjine, and S. Schumacher. Intelligente Fusion von mehreren Bildquellen für eine präzise Fahrspurvermessung und Lokalisierung des Fahrzeugs in der Fahrpspur. *AAET, 15. Braunschweiger Symposium, Braunschweig, Germany*, 2014.

[37] F. Philipp, H. H. Tadjine, S. Schumacher, and R. Rojas. Real time detection and classification of arrow markings in urban streets. *FISITA, World Automotive Congress, F2016-ACVA-007*, 2016.

[38] F. Philipp, H. H. Tadjine, S. Schumacher, and R. Rojas. Real time detection and classification of arrow markings in urban streets. *International Journal of Automotive Technology*, 19(2):379–386, 2018.

[39] J. Rabe, M. Hübner, M. Necker, and C. Stiller. Ego-lane estimation for downtown lane-level navigation. *IEEE Intelligent Vehicles Symposium (IV), June 11-14, Redondo Beach, CA, USA*, 2017.

[40] A. Richter and M. Scholz. *Road2Simulation Guidelines. Leitfaden zur Erhebung von Straßendaten für Simulation und Entwicklung.* Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Center), 2016.

[41] RMS-1. Teil 1: Abmessungen und geometrische anordnung von markierungszeichen. In *Richtlinien für die Markierung von Straßen.* Arbeitsgruppe Verkehrsführung und Verkehrssicherheit, Forschungsgesellschaft für Straßen- und Verkehrswesen e. V., 1993.

[42] RMS-2. Teil 2: Anwendung von fahrbahnmarkierungen. In *Richtlinien für die Markierung von Straßen.* Arbeitskreis Geometrie der Markierungen, Arbeitsausschusses Fahrbahnmarkierungen, Forschungsgesellschaft für das Straßenwesen e. V., 1980.

[43] J. Sample and E. Ioup. Map projections. In *Tile-Based Geospatial Information Systems*, pages 165–191. Springer US, 2010.

[44] D. Scaramuzza, A. Martinelli, and R. Siegwart. A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, page 45, 2006.

[45] M. Schreiber, C. Knöppel, and U. Franke. Laneloc: Lane marking based localization using highly accurate maps. In *Proceedings of IEEE Intelligent Vehicles Symposium*, pages 449–454, 2013.

[46] Shigang Li and Y. Shimomura. Lane marking detection by side fisheye camera. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 606–611, 2008.

[47] I. Steinwart and A. Christmann. *Support Vector Machines.* Springer, 1st edition, 2008.

[48] S. Suzuki and K. Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30:32–46, 1985.

[49] Z. Tao, P. Bonnifait, V. Fremont, and J. Ibanez-Guzman. Mapping and localization using gps, lane markings and proprioceptive sensors. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 406–412, Nov 2013.

[50] L. Turner. An introduction to particle filtering. *Lancaster University: Lancaster, Lancashire, United Kingdom*, pages 1–25, 2013.

[51] S. Vacek, C. Schimmel, and R. Dillmann. Road-marking analysis for autonomous vehicle guidance. In *ECMR,2007, no. 1*, pages 1–6, 2007.

[52] N. Wang, W. Liu, C. Zhang, H. Yuan, and J. Liu. The detection and recognition of arrow markings based on monocular vision. In *Proceedings of the 21st annual international conference on Chinese control and decision conference*, pages 416–442, 2009.

[53] G. Welch and G. Bishop. An introduction to the kalman filter. *University of North Carolina: Chapel Hill, North Carolina, US*, pages 1–16, 2006.