Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

THE UNIVERSITY OF WESTERN AUSTRALIA

A robust Lane Detection and Tracking System based on Bayesian State Estimation

Master's Thesis
by

Markus Kohler

at Robotic and Automation Lab
Centre of Intelligent Information Processing Systems
University of Western Australia
Perth

Supervisor:	Assoc. Prof. Dr. rer. nat. habil. Thomas Bräunl
1st Examiner:	Prof. Dr. rer. nat. Thorsten Leize
2nd Examiner:	Prof. Dr.-Ing. Ulrich Grünhaupt

Submitted in June 2010

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Perth, den 20. Juni 2010

Acknowledgments

I would like to use the opportunity to thank all those people who supported me and made this thesis possible.

First of all, I would like to express my special thanks to my supervisor Associate Professor Dr. Thomas Bräunl for his continual support and guidance, for providing such an interesting and challenging topic, and for the chance of writing my Master's thesis at the University of Western Australia in Perth.

Many thanks to Professor Dr. Leize and Professor Dr. Grünhaupt for providing me great and uncomplicated support to carry out my thesis in Australia.

Furthermore, I would like to thank the DAAD (German Academic Exchange Service) for the financial support which made my stay in Perth possible.

Thanks to Jonathan for being always on the ball in the REV lab and to Linda for all organizational support and proof-reading my thesis.

Special thanks to Georgie for heaps of great advices to improve my thesis structure, for proof-reading, for being such a great house mate and for countless philosophical discussions.

A very special thank you to Shaun and Monika for accompanying me on this exciting journey here in Australia.

Many thanks to Frank, Jithin, Teoh, Roozbeh, Sushil and Lixin for having a great time in the lab and for refreshing discussions in the lunch room.

Finally I would like to thank my father, my mother and my brother for all kind of support in the last years and all my friends back home for their sympathy and support during my time abroad.

Perth, 20^{th} of June, 2010

Abstract

Lane recognition is one of the essential components of modern driver assistance systems. It enables the system to form a comprehensive "understanding" of the current situation, on which further decisions are based.

Lane recognition is best performed by vision systems and is composed of two parts. The first detects the lane in the image and the second tracks the detected lane in time-domain. The combination of these comprises a challenging task in terms of image processing and non-linear state estimation.

There are two approaches which are commonly used for tracking algorithms in image processing: Kalman filtering and Particle filtering. This project focused on Particle filtering for two reasons; firstly, Particle filtering is an approximation of an optimal non-linear estimator, whereas Kalman filtering is an optimal linear estimator which approximates a non-linear system. The second reason is that Particle filtering is able to introduce non-Gaussian measurement likelihoods. Both of these reasons, make Particle filtering the better choice for dealing with the estimation problem of LDT algorithms. Its barrier to widespread application so far has been the high computation power that is needed to run the system in real-time. Fortunately, with steadily increasing computer hardware performance, this barrier is diminishing, and Particle filtering is gaining more and more significance in state estimation.

In this thesis work a robust lane detection and tracking (LDT) algorithm for highway applications was designed and successfully implemented. The Particle filter design is supported by a detailed description of the mathematical background, which points out the connections between the background and the LDT implementation. In addition, the major implementation issues of Particle filters in general and the ways these affect the LDT algorithm are discussed. Finally, the algorithm accuracy and performance is analyzed and illustrated to demonstrate the possibility of using Particle filtering for real-time LDT algorithms.

Contents

1. Introduction

Short by after the introduction of the automobile, the dream of accident-free transportation arose. Now, about 100 years later, leading car manufacturers and scientists around the world work on the dream's fulfillment.

Nevertheless, 35.000 people died in car accidents during the year 2009 in Europe. Worldwide, an estimated 1.2 million people are killed and 50 million people are injured in road crashes annually. Since the 1970's however, there has been a steady decline in accidents even though the absolute number of registered cars has increased.

Reasons for this development within the last years are manifold: introduction of safety systems and the legislation for obligatory usage contributed, as well as regulations for alcohol consumption and improved road conditions.

Most of these safety systems protected the driver passively by reducing the injury in situations of an accident. The most effective innovations have been the increase of the collapsible zone, safety belts, airbag and the head rest.

In 1978, the first anti block system was produced in series, and with it a new era of active safety systems was introduced. It was followed by the electronic stability program, electronic brakeforce distribution, dynamic traction control and others. These systems improved the driver's ability to control the vehicle in already dangerous situations.

In the beginning of the 21^{st} century a new trend became apparent: active safety systems that recognize possible dangerous situations in advance and inform the driver at an early stage or even interact autonomously to prevent dangerous situations.

The research in driver assistance systems has evolved to high diversity among the systems. Two main categories are distinguished by the method of perceiving the environment: active and passive sensor elements.

The motivation to build these systems based on passive sensor elements is given in the following section. After a brief statement of the thesis' objectives, the introduction will be finalized by an overview of the thesis' outline.

1.1 Motivation

Most of the currently available driver assistance system features are implemented based on active sensor elements. Active sensor systems are systems that emit an energy signal and measure the back reflected amount of energy to gather information about the environment. Difficulties with these systems arise, once another system of similar technology is present and causes interferences. Furthermore active systems are much more expensive to produce even in high number series productions. Therefore researchers are focusing on passive sensor systems e.g. monocular or binocular vision systems.

Current research goes toward intelligent systems which extract relevant information from the environment and are able to interpret the information to form an "understanding" of the scene.

The "understanding" includes recognition of the environment and categorizing the objects which move within the environment. To form a comprehensive "understanding" a basic reference needs to be defined. Most driver assistance systems are therefore based on a road model. This brings lane detection to significant importance.

A robust and reliable lane detection method is therefore essential for the success of every driver assistance system. Successful and fast improving driver assistance systems can prevent injury and death of people and reduce the immense costs cased by car accidents.

1.2 Objectives

This thesis has two major goals. The first is to give a comprehensive introduction to Bayesian state estimation and its various implementation methods, which are generally called Particle filters. This is achieved by giving an overview of Bayesian theory and introducing these ideas to the problem of state estimation. Equipped with this knowledge, the Particle filter is introduced and the connections between the implementation steps and the Bayesian state estimation are pointed out.

The second major goal is to implement a robust lane detection and tracking system which is designed for highway and major urban road environments. The algorithm is implemented using the Particle filter as tracking mechanism, so that the knowledge gained about Particle filtering is directly applied to a complex estimation problem.

For testing the algorithm, four different automotive sequences, provided by the German car manufacturer Daimler AG, are used. As a secondary goal, a measurement unit is designed to record further automotive sequences in the Perth city area.

1.3 Thesis Outline

The following chapter 2 gives a detailed analysis of the LDT problem, first as a general formulation, then describing the specific requirements and finally by summarizing current research activities in a literature survey.

In chapter 3 the mathematical background of Particle filtering, based on Bayesian theory and the Bayesian state estimation, is introduced.

The estimation theory is followed by the design of the LDT algorithm in chapter 4. In this chapter all necessary steps of the Particle filter are designed in respect to the LTD algorithm. Furthermore, some major implementation issues concerning the tracking quality are discussed.

In chapter 5 the actual implementation of the LDT algorithm in C using Intel's open source computer vision library OpenCV is described.

The implementation is followed by an accuracy and performance evaluation in chapter 6. The analysis includes several modules and gives the reader an impression of the general algorithm function.

In chapter 7 an overall conclusion is given and several suggestions for improving the Particle filter and LDT investigation are given.

2. Problem Analysis

The underlying idea of every information processing system is to have a comprehensive ”understanding” of the given problem. Once the ”understanding” is given, the system can decide to react to the situation in whatever way seems to be appropriate.

The ”understanding” is therefore implemented in terms of a mathematical model on which some system information can be matched on. For vision systems this can be described as perceiving an image of a real world scene, then extracting relevant information and finally matching this information to a 3D-model.

This chapter will give an understanding of the general problems one is faced with when designing a lane detection and tracking system based on vision information processing. First an overview of the general problem is presented which is then followed by a brief set of requirements for the designed system. Finally a literature survey will complete the reader's understanding of the problem and current approaches to deal with it.

2.1 Lane Detection and Tracking Systems

The automotive driver's scene is exposed to a wide range of different environmental conditions which are individually characterized by a wide range of variations. Furthermore, the scene is composed of a huge variety of different objects where all together they interact in the same environment. All these factors complicate the task of perceiving the environment and matching these information to the mathematical model.

Variations which might occur in automotive scenes are:

- climate and weather conditions
  (snowy, rainy, sunny, foggy, dusty)
- lighting conditions
  (direct sunlight, shadowing, no sunlight)
- lane markings style
  (circular reflectors, solid/segmented, no markings, marking color)

- complex lane markings
  (markings at cross sections, temporary lane markings)
- road surface variations
  (light/dark pavement, wet pavement, dirty pavement)
- environmental structures
  (landscape, forest, urban, interurban, highway)

In the survey from the year 2006, McCall et al. [McTr06] point out that the problem of lane detection and tracking (LDT) is structured by four basic steps:

- Defining a road model which serves as a mathematical model to describe the reality. Common road models are e.g. polynomial curves or splines on a flat plane.
- Extracting relevant road features from the input data. Common feature extraction techniques are e.g. derivation filters, steerable filters, matched filters.
- The extracted features are then segmented by the post processing step so that, based on the segmentation, the model parameters can be determined. Common segmentation techniques are e.g. Hough transform, randomized Hough transform, RANSAC fitting algorithm.
- More complex systems then use higher level techniques to track the lane and vehicle position. Common tracking techniques are e.g. Kalman filtering, Particle filtering.

All four steps need to be adjusted to each other to work efficiently together. Step 2 and 3 are mainly responsible for removing noise from the signal by means of low level image processing. In step 4 the noise reduced information is then considered in a higher level processing step which further removes noise but also forms the ”understanding” of the automotive scene.

In the following section the requirements for this thesis are detailed, this is then followed by a wide literature survey to give a brief introduction to current research efforts.

2.2 Requirements

Since all possible automotive scenes cover a vast range of different situations, it is necessary to limit the system requirements to a definite set of automotive situations.

General requirements are:

1. minimum frame rate is 30 fps
2. minimum observation range is 20 m

The system response rate for driver assistance systems is typically in the range of 30 fps. This is an important time requirement that needs to be met to achieve both, fast interaction for safety purpose, and smooth result visualization.

The system design is focusing on highway and fast urban infrastructures. This means, low curvature roads are expected, furthermore, relevant road features will be different marking types (solid, segmented) and road-curb borders. The system should also be able to handle lane changes and partially occluded roads.

2.3 Literature Survey

Lane detection and tracking systems are distinguished in three main categories. First, there are these systems which process the data and apply a certain model only in image space. They do not introduce a camera model to describe the relation between real world and image space. These applications are easy and straightforward to implement on low performance systems. However, they do not provide a comprehensive "understanding" of their surroundings. Two examples are given in the following paragraphs.

Gong et al. [GWZX+08] introduced a lane detection algorithm designed especially for high speed highway situations. They limited the general ROI to a certain area of the image where the lane is expected to be. They used a median filtering for smoothing and a Sobel filter for lane marking extraction. The features are then fitted to lines in the image space by improved Hough transforming.

Saudi et al. [STHS08] proposed a lane detection method for indoor environment. The lane features are extracted using a Sobel filter. The magnitude image is then binarized by an adaptive threshold method. The features are then fitted to lines in image space by randomized Hough transform. They use a simple tracking mechanism in assuming the subsequent line having a slope within 20 percent of the last one.

The second type of system introduces a camera model to match the features given in image space to a road model given in world space. Due to the transformation from image to world space, these systems are more computation expensive than the systems mentioned above. However, they provide a model located in world space which is more convenient as interface to other system components.

For example Samadzadegan et al. [SaST06] proposed a lane detection system designed for Highway and other marked roads. In order to reduce the needed computation effort they apply a three-layer image pyramid to iteratively determine the lane markings. For each layer, they use a Sobel filter to extract the road features. In the first layer a randomized Hough transform is used to determine width and heading direction, then in second and third layer they use a genetic algorithm and the camera model to fit a parabolic road model which is assumed to be on a flat plane to the extracted features. They also use a simple tracking mechanism by applying *a priori* knowledge by setting the ROI of the first pyramid layer according the results of the last frame.

Aly [Aly08] presented a lane detection algorithm for urban scenes. By introducing a camera model, an inverse perspective mapped image is generated from which then the features are extracted. For lane marking extraction they use a static two dimensional Gaussian filter. The horizontal direction is a smoothing Gaussian which is intended to reduce the noise, the vertical direction is a second-derivative of Gaussian which is matched to the expected line width to extract the line feature. Afterwards the image is adaptive thresholded. To fit the extracted features to the spline model he uses a series of three different algorithms. First, the lines are roughly detected by a simplified Hough transform, that is just a vertical sum of the filtered image which results in a histogram like structure. In that histogram all local maximums are expected to be lanes in the original image. Then a RANSAC-line fitting algorithm first refines these found lines. Afterward a RANSAC-spline fitting algorithm using third degree Bezier splines is applied to get the final result.

Lipski et al. [LSBL+08] presented a multi-camera based lane detection system for clearly marked roads. They merged three different camera images together by means of inverse perspective mapping. To map the transformed image to real world coordinates they used an inertia navigation system which is corrected by GPS coordinates. To extract the lane marking features they apply these three different assumptions to a local area: local contrast must exceed a certain threshold; the two most weighted grey values of a local histogram must have a minimum difference; and the bright pixels must have an evident shape and orientation. For feature fitting they used a piecewise linear defined lane model based on a flat plane. They assume all four lines (lane marks of own and adjacent lane) are lying in parallel. To fit the parallel lines to the extracted features an initial position is either deduced from previous images or by estimating. Then the features are fitted by a RANSAC fitting algorithm. Feature tracking is done by adding the features of the last frame of the same segment to the feature image of the current frame. This is a cumulative feature adding approach.

The approach proposed by Lipski et al. tends already towards the third type of LDT systems which is distinguished by introducing a camera model to match the extracted features to a model in world space and also using vehicle motion information to estimate the subsequent position of the lane. Dickmanns et al. [DiMy92] pioneered these type of system already in 1992, when they proposed a 3D road model with eight parameters and a Kalman filter to estimate their values in time-domain. Since the Dickmanns et al. publication, the information processing hardware systems have made major improvements in performance so that various systems have been designed focusing on the task to equip modern vehicles with vision information processing systems to give a comprehensive estimate about the environment.

McCall et. al. [McTr06] provide a wide survey on LDT systems. Additionally, they proposed the "video-based lane estimation and Tracking" (VioLET) system which is designed for different highway environments. They tested the system on a 65 km long route through San Diego. For road feature detection, they focus on lane markings and road texture in a great variety, that is solid lines, segmented lines and circular reflectors and different road textures. Lane markings are extracted by means of steerable filters for the range up to 20 meters in which they give significant results. They are variable in angle space which enables either to tune on a single angle respond for solid line detection or to tune on full-angle space respond for circular reflector detection. For higher range than that they use adaptive template matching to measure the road curvature. To eliminate perspective effects, both feature extraction methods are applied to inverse perspective mapped images. This allows use of a single sized kernel for steerable filtering and a straight forward template matching method without perspective distortions. Then a multi-lane parabolic road model is fitted to the extracted features in the inverse perspective mapped image by minimizing the squared error. The lane position is tracked by an original Kalman filter. To improve the estimation robustness, before finding the update measurements, outlying features are removed in two stages. First, only the filter candidates within the vicinity of the lanes are used in updating the lanes. Second, for each lane, the first and second moments of the point candidates are computed, hence the lines heading direction is identified. Lines with different direction as expected are removed. Furthermore, circular reflectors are identified by single points on the inverse perspective mapped image, where their position is determined by the vehicle's movement. Spots which do not follow these dynamics are removed. The update

measurement estimates lane position, angle and width by means of Hough transform and the statistic values first and second moment.

As in McCall's et al. survey shown, Kalman filtering techniques have been used and improved in a wide range of applications particularly for highway environments. As will be shown with proceeding this literature review, the relevance of Particle filters in this research area increased tremendously during recent years. Reason for this is, that in automotive scenes a wide range of uncertainty is present due to different whether conditions, road conditions or lane marking quality. Systems using Kalman filtering rely on techniques like data association, robust line fitting and averaging to deal with these uncertainties and to reduce the effects of outliers. In contrast, Particle filters provide a method to deal with uncertainties in a comprehensive and abstract way. It is possible to exploit information such as "this pixel is more likely to belong to the road than to the background", which would not be possible with Kalman filtering techniques.

In 2001, Southall et al. [SoTa01] introduced the Particle filter as a stochastic road shape estimator to LDT systems. Their system is designed for different highway scenes under varying imaging conditions. To extract lane markings a two-stage algorithm is used. First a matched filter of triangular shape is used to detect lane markings, then line candidates are verified by evaluating the intensity value which must reach a certain threshold. The system is described by a six-dimensional state space, the state variables are lateral position; heading direction; first and second order curvature; lane width; and the camera's pitch angle. The state hypotheses are evaluated in image space, therefore a transformation between world space and image space is done in the form of applying a camera model. This step is essential for every estimation process that describes a movement in world space and uses observations given in image space for evaluation. To improve the Particle filter performance, they introduced *importance sampler*, which includes some current measurement to adjust the prior. They used a randomized Hough transform to get a measurement for the vehicle's heading direction, lane width, pitch angle and lateral position. They furthermore introduced *partitioned sampling* which divides the state space in groups of subspaces. Therefore they separated the parameters which can be measured by the Hough transform from the curvature parameters. This improves the performance since the effective state order is reduced and less particles are needed.

Where Southall et al. only used lane markings as road likelihood, Apostoloff et al. [ApZe03] introduced a fusion scheme to combine lane markings, road edges and the color of the road to achieve a higher robustness.

Danescu et. al. [DNMT08] proposed a stereo vision-based probabilistic lane tracker using Particle filters. Stereo vision is used to detect the pitch angle between camera and road surface as well to extract lane features. They use two feature maps, they are: curb features extracted by stereo vision and lane markings extracted by dark-light-dark transition detection. The particle evaluation scheme undergoes a physical boundary check, where the lane width and position are limited to a specific range. The latter one probably reduces the particle filter to find neighboring lanes. To evaluate the current lane estimate, they use a comparison scheme of average weight of predicted particle set and the average weight of totally random particles, introduced by the initialization step.

Franke and Loose et al. [FrLK07] go one step further and propose a lane recognition algorithm for rural roads without any lane markings, demonstrated as lane recognition algorithms designed for rural roads without lane markings. Similar to Apostoloff's et al. approach, they used multiple cues which are fused together within the Particle filter framework. However their approach is slightly different to all earlier mentioned systems, since they try to detect the complete road area instead of separately detecting the lanes. This can be seen as a basic road detection on which a lane separation mechanism can follow.

Franke and Loose et al. [LoFS09] further improved their system by adding measurements of a Doppler radar and depth information of a stereo vision set-up as evidence grid to the particle filter framework. The Doppler radar provides distance information of the environment or an object as well as its relative velocity which is used to identify whether the measurement belongs to the environment or an object within the environment. Furthermore they introduced a Kalman filter step to compute and track the Particle filter's variance Q of the evolution model.

By the steady development of increasingly faster high performance information processing systems, the idea of applying Particle filters for state space estimation in real-time embedded systems is shifting towards the range of reality. The next chapter will introduce the concept of Particle filtering starting with its very essence: The Bayesian Theory.

3. Probability Theory and State Estimation

In this chapter, the Bayesian theory is applied to the problem of lane detection and tracking utilizing vision systems and state estimation methods.

The first section briefly reviews the history of the Bayesian theory, in order to provide the reader with the knowledge necessary to understand the idea of Bayesian estimators.

The Bayesian state estimation is then introduced in the following section. The Bayesian theory is first formulated in the context of state estimation, then a mathematical deduction of the final formula of the recursive Bayesian estimator is given.

Finally, the theory of the Bayesian estimator is implemented as a Monte Carlo Simulation approach - the so called Particle filter. A general description of the algorithm is given, followed by an in depth mathematical description.

3.1 Origins of Bayesian Theory

This section serves to introduce the essential idea of Bayesian theory and the terminologies used in this chapter. The argument is based on an introductory article about Bayesian methods by E. T. Jaynes [Jayn86].

Probability theory deals with the problem of reasoning in situations where only incomplete knowledge is available. Documented records point back to 500 BC when Herodotus was dealing with the decision policy of Persian kings.

In 1713, the principal idea of probability theory was described mathematically by James Bernoulli in his work "Ars Conjectandi".

To represent incomplete knowledge, Bernoulli introduced a set of basic "equally possible" cases denoted by $\{x_1, x_2, ..., x_N\}$ which are called events, or propositions. The set defines the so called *sample space* that is denoted as H_0 (or by other authors as Ω, S (for space) or U (for universe)). As an example, for rolling two dices $N = 6^2 = 36$ and $H_0 = \{\{1,1\},\{1,2\},\ldots,\{2,1\},\{2,2\},\ldots,\{6,6\}\}$.

By introducing a proposition of interest A, that is defined to be true on a subset $H(A)$ of some points M of H_0, the probability of A being the realization or outcome of a trial can be defined as

$$P(A) = M/N, \tag{3.1}$$

where M is referred to as *multiplicity* of A and N as *cardinality* of H_0.

Plausibly, the reasoning used in decision making processes is based on the perceived probabilities of the outcomes of possible choices. The sample space H_0 does not necessarily need to be equal to the natural sample set of the real world, H_N, however H_0 can serve as a first estimate of H_N and can be updated with further knowledge added.

Bernoulli also proved the mathematical connection between probability and frequency. He showed that considering n independent observations, if A is found to be true m times, the frequency of A true, given by $f(A) = m/n$, becomes very close to the probability $P(A) = M/N$ for large n. The observed frequency $f(A)$ is a reasonable estimate for the probability $P(A)$ in the natural sample space H_N and as such gives the possibility of updating the initial sample space H_0. These first ideas are a good start to describe the reasoning process in terms of mathematics, however Bernoulli died before he got get any further.

In 1763, Thomas Bayes' published article gives an different approach to Bernoulli's unfinished problem. Whereas Bernoulli stated the probability of observing event A true n times out of m trials is given by N and M. Bayes inversed the point of view and stated the probability that M has various values for N is given by n and m.

The terminologies *probability* and *likelihood* are distinguished based on these two different points of view. Probability allows us to predict a particular outcome based on known parameters, whereas likelihood allows us to estimate unknown parameters based on observations.

In 1774, Laplace generalized Bayes' principle in greater clarity and deduced what is nowadays called "Bayes' Theorem".

Denoting the propositions A and B, where AB stands for the proposition "both A and B are true". To represent joint probability let the symbol $P(A|B)$ stand for "the probability that A is true, given that B is true". Then the basic product rule is

$$P(AB) = P(A|B)\,P(B). \tag{3.2}$$

Since AB and BA are the same proposition, by interchanging A and B on the right hand side and equalizing both terms, Bayes' Theorem is expressed as

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}. \tag{3.3}$$

The importance of Bayes' theorem lies in its use as a mathematical description of learning processes in general. Therefore, it is most usefully expressed in terms of *probability distribution functions* (pdfs) denoted by lower case p

$$p(A|B) = \frac{p(B|A)\,p(A)}{p(B)}, \tag{3.4}$$

here $p(A)$ is the *prior probability distribution* for proposition A it represents the pre-gained knowledge about A. $p(A|B)$ is its *posterior probability distribution* it represents the knowledge [we have] about A after updating it with new information B. $p(B|A)$ is the likelihood function of obtaining the information B given that A is correct. $p(B)$ is called *evidence* and is merely considered as a normalization term to fulfill the condition of $\int p(A|B)\,dp = 1$.

The next step is to introduce the idea of Bayesian theory to state estimation problems. Therefore, the next section presents the principle and the mathematical deduction of the Bayesian state estimator.

3.2 Bayesian State Estimation

The general idea of state estimation is to estimate the current state of a given system, based on a mathematical description of the system's dynamic behavior (system model), its state history, some known input signals, and some noisy observations of the current system state. The term *state* stands for those parameters that represent the system's internal condition or status.

A typical description of a linear system is given by $\mathbf{s}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B} + \mathbf{w}_k$, where $\mathbf{s}_k \in \mathbb{R}^n$ is the state at time k, $\mathbf{A}$ is an $n \times n$ matrix, $\mathbf{B}$ is an $n \times 1$ vector and $\mathbf{w}$ is a $n \times 1$ random noise vector drawn from a Gaussian distribution.

The more general, not necessarily linear form of a system description is given by the equations

$$\begin{aligned} \mathbf{s}_k &= \mathbf{f}_k(\mathbf{s}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}), \\ \mathbf{y}_k &= \mathbf{h}_k(\mathbf{s}_k, \mathbf{u}_k, \mathbf{v}_k), \end{aligned} \tag{3.5}$$

where $\mathbf{s}_k \in \mathbb{R}^n$ is the system state, and $\mathbf{y}_k$ the noisy observation at time k. The function $\mathbf{f}_k(\cdot)$ is the time-varying, nonlinear system equation which is also referred to as *state evolution model* with process noise $\mathbf{w}_k$. The function $\mathbf{h}_k(\cdot)$ is the time-varying, nonlinear observation model with observation noise $\mathbf{v}_k$ which is assumed to be independent from the process noise. $\mathbf{u}_k$ represents some known input signals.

With Bayes' theory in mind it is evident to consider the problem of state estimation in terms of reasoning as best as possible "finding the best state estimate" in situations "at time k" where only incomplete knowledge "input signals, noisy observation and ambiguous state history" is available. Expressing this in form of Bayes' theorem one can write

$$\underbrace{p(s_k|Y_k)}_{\text{posterior}} = \frac{\overbrace{p(Y_k|s_k)}^{\text{likelihood}} \cdot \overbrace{p(s_k)}^{\text{prior}}}{\underbrace{p(Y_k)}_{\text{evidence}}}, \tag{3.6}$$

where $p(s_k)$ is the prior probability distribution of the state vector, $p(s_k|Y_k)$ is the state's posterior probability distribution, $p(Y_k|s_k)$ is the likelihood of the observation and $p(Y_k)$ is the evidence of the observation, and Y_k is the set of observations $\{y_1, y_2, \ldots, y_k\}$.

The *prior probability distribution* of the state vector s_k represents pre-knowledge about the current state gained from the state history, some input signals and previous state observations. This is merely expressing a preference for the system state toward one or several particular states. It is also called *proposal distribution*.

The *posterior probability distribution* of the state vector s_k represents the knowledge about the current state after updating the prior by the current state observation. Both prior and posterior must be understood as distributions of the probability that the state vector s_k has a particular value s_k^i over the state space $\mathbb{R}^n$.

The *likelihood* of obtaining the observation, given that s_k is the true state, represents the incomplete information about the true current state. Thus it is not actually representing the truth of the current state, but merely a preference for the system state toward the true current state.

The *evidence* of the observation is, as already stated, just to be understood as a normalization term to fulfill the condition of $\int p(s_k|Y_k)\,dp = 1$.

Reading equation 3.6 one might say "the probability that $s_k = s_k^i$ is correct given the current observation is equal to the likelihood of obtaining the observation given that $s_k = s_k^i$ is correct, multiplied by the probability that $s_k = s_k^i$ is correct, divided by the probability of the observation."

The following two subsections show the mathematical deduction of the recursive Bayesian estimator with the intension to give a deeper understanding of its implementation.

3.2.1 Deduction of the Bayesian Estimator

The following paragraph shows, how the pdf of s_k conditioned on the set of observations $\{y_1, y_2, \dots, y_k\}$ can be calculated. It is based on the deduction given by Dan Simon [Simo06].

For the sake of simplicity the following two key points are assumed:

1. the system is a first order Markov process, that is, the current state depends only on the previous state

$$p(s_k|s_{k-1}, s_{k-2}, \dots, s_0) = p(s_k|s_{k-1})$$

2. the observations are conditionally independent, that is, the current observation depends only on the current state

$$p(y_k|s_k, \dots) = p(y_k|s_k)$$

The conditioned pdf is given by applying Bayes' theorem

$$p(s_k|Y_k) = \frac{p(Y_k|s_k)\,p(s_k)}{p(Y_k)} \quad \text{with} \quad Y_k = \{y_1, y_2, \dots, y_k\}. \tag{3.7}$$

By again applying Bayes' rule on the last term, $p(s_k) = p(s_k|Y_{k-1})\,p(Y_{k-1})/p(Y_{k-1}|s_k)$, and splitting up the observation set Y_k into $\{y_k, Y_{k-1}\}$, the conditional pdf can be rewritten as

$$p(s_k|Y_k) = \frac{p(y_k, Y_{k-1}|s_k)}{p(y_k, Y_{k-1})}\,\frac{p(s_k|Y_{k-1})\,p(Y_{k-1})}{p(Y_{k-1}|s_k)}. \tag{3.8}$$

For the sake of eliminating the condition of the observation set $\{y_k, Y_{k-1}\}$ on s_k and $p(s_k|Y_{k-1})$, the definition of conditional probability is applied: $p(A|B) = p(A, B)/p(B)$.

$$p(s_k|Y_k) = \frac{p(s_k, y_k, Y_{k-1})}{p(s_k)\,p(y_k, Y_{k-1})}\,\frac{p(s_k, Y_{k-1})\,p(Y_{k-1})}{p(Y_{k-1})\,p(Y_{k-1}|s_k)}. \tag{3.9}$$

To reduce the complexity, the fraction is expanded by $p(s_k, y_k)$

$$p(s_k|Y_k) = \frac{p(s_k, y_k, Y_{k-1})}{p(s_k)\,p(y_k, Y_{k-1})}\,\frac{p(s_k, Y_{k-1})\,p(Y_{k-1})}{p(Y_{k-1})\,p(Y_{k-1}|s_k)}\,\frac{p(s_k, y_k)}{p(s_k, y_k)}. \tag{3.10}$$

Using the ratios of various joint pdfs, marginals pdfs in the above equation. The following ratios are used:

- $p(s_k, y_k, Y_{k-1})/p(s_k, y_k) = p[Y_{k-1}|(s_k, y_k)]$
- $p(s_k, Y_{k-1})/p(Y_{k-1}) = p(s_k|Y_{k-1})$
- $p(s_k, y_k)/p(s_k) = p(y_k|s_k)$
- $p(Y_{k-1})/p(y_k, Y_{k-1}) = 1/p(y_k|Y_{k-1})$

to obtain

$$p(s_k|Y_k) = \frac{p[Y_{k-1}|(s_k, y_k)]\, p(s_k|Y_{k-1})\, p(y_k|s_k)}{p(y_k|Y_{k-1})\, p(Y_{k-1}|s_k)} \tag{3.11}$$

In considering assumption 2 of conditionally independent observations (since y_k is a function of x_k), the term $p[Y_{k-1}|(s_k, y_k)] = p(Y_{k-1}|s_k)$ cancels in the above equation. This yields the final equation of the conditional pdf of s_k

$$p(s_k|Y_k) = \frac{p(y_k|s_k)\, p(s_k|Y_{k-1})}{p(y_k|Y_{k-1})}, \tag{3.12}$$

Since the first observation is obtained at $k = 1$ the pdf of $\mathbf{s}_0$ is unconditioned and assumed to be known. The pdf of $\mathbf{s}_0$ can be written as

$$p(\mathbf{s}_0) = p(\mathbf{s}_0|\mathbf{Y}_0) \tag{3.13}$$

where $\mathbf{Y}_0$ is the set of no observations. Once the conditional pdf of s is computed, the state estimate $\hat{\mathbf{s}}$ can be established using whatever method seems to be appropriate.

From here, the recursive Bayesian estimator is given by

$$p(s_k|Y_k) = \frac{p(y_k|s_k)\, p(s_k|Y_{k-1})}{\int p(y_k|s_k)\, p(s_k|Y_{k-1})\, ds_k} \tag{3.14}$$

where:

prior: $p(s_k|Y_{k-1}) = \int p(s_k|s_{k-1})\, p(s_{k-1}|Y_{k-1})\, ds_{k-1}$

likelihood: $p(y_k|s_k)$ is defined by the observation model

evidence: $p(y_k|Y_k) = \int p(y_k|s_k)\, p(s_k|Y_{k-1})\, ds_k$

$p(s_k|s_{k-1})$ is given by the system equation $f_k(\cdot)$ and $p(s_{k-1}|Y_{k-1})$ will be given at runtime.

3.3 Particle Filtering

The term *Particle filter* refers to several different *sequential Monte Carlo methods* using sets of samples, the so called *particles*, to represent distribution functions of the Bayesian equations. Different methods have been proposed, including *Bootstrap filtering*, *the condensation algorithm*, *Particle filtering*, *interacting particle approximations* and *survival of the fittest*.

This section gives a brief description of the condensation algorithm used in the implementation. It starts with a general idea of how the probability distributions are represented in Particle filters, then the algorithm steps are detailed. A clear mathematical deduction is given in the following subsection.

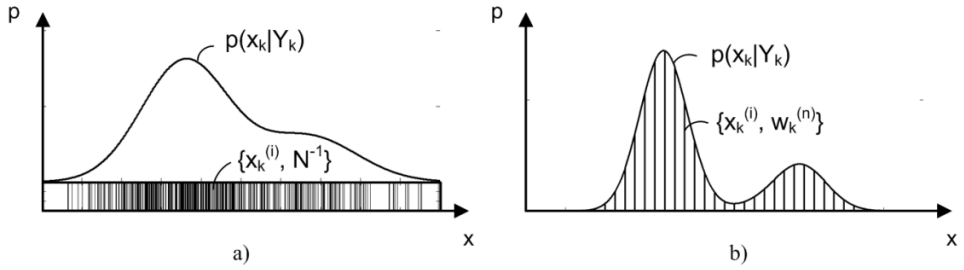

Figure 3.1: *distribution representation by distributed particles; a) particles of equal weight; b) particles of unequal weight*

Particle filters are intended to solve the equations given by the Bayesian estimator numerically rather than analytically. Therefore, a given distribution, e.g. $p(x_k|Y_k)$, is mapped on a set of weighted, independent distributed samples. This is illustrated in Figure 3.1. The mapping of the distribution on the samples can be formulated as

$$x_k^{(i)} \longleftarrow\!\shortmid\, p(x_k|Y_k). \tag{3.15}$$

Any estimate integral in the following form can be approximated by discrete sums of the weighted samples by

$$E[f(x_k)] = \int f(x_k)\, p(x_k|Y_k)\, dx_k \approx \sum_{i=1}^{N} \tilde{w}_k(x_k^{(i)})\, f(x_k^{(i)}), \tag{3.16}$$

where $f(x_k)$ can be any function of x_k, N is the number of particles and $\tilde{w}_k$ is the normalized particle weight.

The Algorithm

Starting from here, the final goal of the Particle filter is to compute the posterior distribution at time k given by the Bayesian estimator equation

$$p(s_k|Y_k) = \frac{p(y_k|s_k)\, p(s_k|Y_{k-1})}{\int p(y_k|s_k)\, p(s_k|Y_{k-1})\, ds_k}.$$

Step 1: Importance Sampling

Therefore, first, the *prior* distribution needs to be calculated. If $s_{k-1}^{(i)}$ is drawn from the true *posterior* distribution at time $k-1$ then the *prior* distribution is approximated by

$$\begin{aligned} p(s_k|Y_{k-1}) &= \int \underbrace{p(s_k|s_{k-1})}_{\mathbf{f}[p(s_{k-1}),\mathbf{u}_{k-1},\mathbf{v}_{k-1}]} \quad p(s_{k-1}|Y_{k-1})\, ds_k \\ &\approx \sum_{i=1}^{N} \tilde{w}_k(s_{k-1}^{(i)})\, \mathbf{f}(s_{k-1}^{(i)}, \mathbf{u}_{k-1}, \mathbf{v}_{k-1}), \end{aligned} \tag{3.17}$$

where $\mathbf{f}(\cdot)$ is the state evolution model and $s_{k-1}^{(i)} \longleftarrow p(s_{k-1}|Y_{k-1})$. When assuming that the particles have been resampled and the weight are all equal, this simplifies to

$$p(s_k|Y_{k-1}) \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{f}(s_{k-1}^{(i)}, \mathbf{u}_{k-1}, \mathbf{v}_{k-1}). \tag{3.18}$$

If the true *posterior* distribution is not available, the first step is to sample from a *proposal* distribution which is in the simplest case the transition density $p(s_k|s_{k-1})$:

$$s_k^{(i)} \longleftarrow p(s_k|s_{k-1}^{(i)}) \tag{3.19}$$

A more complex model introduces the past observations into the *proposal* distribution, that is $p(s_k|s_{k-1}, Y_k)$. However for the sake of simplicity the used algorithm only applies the *transition* density.

Step 2: Particle Evaluation

In the second step, the *prior* distribution is multiplied with the observation likelihood function. This is approximated as follows

$$p(y_k|s_k)\, p(s_k|Y_{k-1}) \approx p(y_k|s_k^{(i)}) \quad \text{where} \quad s_k^{(i)} \longleftarrow p(s_k|Y_{k-1}). \tag{3.20}$$

The second step therefore computes the particle weight based on the observation likelihood. In other words, a measure is given how good the particle matches to the observation. This is also refereed to as update step. The weights (likelihood of each individual particle) are given by

$$w_k^{(n)} = p(y_k|s_k^{(i)}) = P(v_k^{(i)} = \tilde{y}_k^* - \mathbf{h}_k(s_k^{(i)})), \tag{3.21}$$

where $\tilde{y}_k^*$ is a specific measurement and $\mathbf{h}_k$ is the observation model.

Evaluating the importance weights is the most crucial step of Particle filters. To evaluate the weights a common space needs to be mapped at from both state space and observation space, and a proper distance measure needs to be defined. A detailed explanation of this is given in section 4.5.

Step 3: Normalizing Importance Weights

The normalization factor, given by $\int p(y_k|s_k)\, p(s_k|Y_{k-1})\, ds_k$ is easy to calculate, since all the necessary information is contained in the importance weights. Therefore, it is sufficient to normalize the weights on the sum of the set of all particle weights, which is given by

$$\tilde{w}_k^{(i)} = \frac{w_k^{(n)}}{\sum_{n=1}^{N} w_k^{(i)}} \tag{3.22}$$

Step 4: Resampling Particles

The algorithm up to this point is called *sequential importance sampling*. The problem with this algorithm as it stands, is that after some iterations have passed, the it becomes unstable due to discrepancy between the weights. This is referred to as *weight degeneracy*.

To overcome this issue it is crucial to perform the resampling step, which involves, removing the low weight particles and multiplying the high weight particles. Thus, N unequally weighted particles are mapped into a new set of N equally weighted samples when still preserving the actual probability distribution.

$$\left\{s_k^{(i)}, \tilde{w}_k^{(i)}\right\} \longrightarrow \left\{s_k^{(m)}, N^{-1}\right\} \tag{3.23}$$

Several resampling algorithms have been proposed, a detailed description of these follows in section 3.3.2.

3.3.1 Deduction of Sequential Importance Sampling (SIS)

This section describes the underlying theory of the Particle filter implementation. It is based on the deduction given in [MDDFW01].

The integral given by the equations of the recursive Bayesian estimator are difficult to solve analytically for nonlinear equations with non-Gaussian noise sources. However, sequential Monte Carlo simulation provides a numerical solution to these problems.

To compute these integrals, the distribution is first mapped on independent distributed samples.

$$\hat{p}(s_k|Y_k) = \frac{1}{N}\sum_{n=1}^{N}\delta(s_k - s_k^{(i)}) \quad \text{with} \quad s_k^{(n)} \longleftarrow\!\shortmid\, p(s_k|Y_k) \tag{3.24}$$

Therefore, any estimate of the form of $E[f(s_k)] = \int f(s_k)\, p(s_k|Y_k)\, ds_k$ can be approximated by

$$E[f(s_k)] \approx \sum_{n=1}^{N} f(s_k^{(i)}). \tag{3.25}$$

Since it is often impossible to sample directly from the true posterior density it is reasonable to sample from a known *proposal distribution* $q(s_k|Y_k)$. The following substitution is applied

$$\begin{aligned}
E[f(s_k)] &= \int f(s_k)\,\frac{p(s_k|Y_k)}{q(s_k|Y_k)}\,q(s_k|Y_k)\,ds_k \\
&= \int f(s_k)\,\frac{p(Y_k|s_k)\,p(s_k)}{p(Y_k)\,q(s_k|Y_k)}\,q(s_k|Y_k)\,ds_k \\
&= \int f(s_k)\,\frac{w_k(s_k)}{p(Y_k)}\,q(s_k|Y_k)\,ds_k \\
&= \frac{\int f(s_k)\,w_k(s_k)\,q(s_k|Y_k)\,ds_k}{\int p(Y_k|s_k)\,p(s_k)\,\frac{q(s_k|Y_k)}{q(s_k|Y_k)}\,ds_k} \\
&= \frac{\int f(s_k)\,w_k(s_k)\,q(s_k|Y_k)\,ds_k}{\int w_k(s_k)\,q(s_k|Y_k)\,ds_k} \quad \text{with} \quad w_k(s_k) = \frac{p(Y_k|s_k)\,p(s_k)}{q(s_k|Y_k)}
\end{aligned} \tag{3.26}$$

Finally, by drawing samples from the proposal distribution $q(s_k|Y_k)$, expectations of interest can be approximated by

$$\begin{aligned} E[f(s_k)] &= \int f(s_k)\, p(s_k|Y_k)\, ds_k \approx \sum_{i=1}^{N} \tilde{w}_k(s_k^{(i)})\, f(s_k^{(i)}) \\ \text{with} \quad \tilde{w}_k(s_k^{(i)}) &= \frac{w_k(s_k^{(i)})}{\sum_{i=1}^{N} w_k(s_k^{(i)})} \text{ and } w_k(s_k) = \frac{p(Y_k|s_k)\, p(s_k)}{q(s_k|Y_k)} \end{aligned} \tag{3.27}$$

Using the state space assumptions, the importance weights can be estimated recursively [DDFMR00] by

$$w_k = w_{k-1}\, \frac{p(y_k|s_k)\, p(s_k|s_{k-1})}{q(s_k|S_{k-1}, Y_k)}. \tag{3.28}$$

When using the transition density as proposal distribution, $p(s_k|s_{k-1})$ cancels out and the weight is calculated by

$$w_k = w_{k-1}\, p(y_k|s_k). \tag{3.29}$$

3.3.2 Extension to Sampling Importance Resampling (SIR)

The problem with using SIS is that the particle weights decrease with increasing time. This phenomena is referred to as *weight degeneracy* and was first pointed out by Gordon et al. [GoSS93].

Weight degeneracy stems from the very first step in the earlier deduction: drawing N samples from a probability distribution at a definite time instant to represent the distribution numerically

$$s_k^{(i)} \longleftarrow\!\shortmid\, p(s_k).$$

This set of samples represents the distribution in its specific shape at time instant k. By using this sample set to calculate, or evolve other distributions, all samples will still have their origin at time k. Therefore, to have a time-dynamic particle evolution it is essential to sample again to represent the distribution at the following time instant $k+1$.

Given a weighted particle set

$$p_N(s_k) = \sum_{i=1}^{N} \tilde{w}_k^{(i)} \delta(s - s_k^{(i)}), \tag{3.30}$$

the resampling step can be formally written as

$$\hat{p}_N(s_k) = \sum_{j=1}^{N} \frac{1}{N} \delta(s - s_k^{*(j)}) = \sum_{i=1}^{N} \frac{n_i}{N} \delta(s - s_k^{(i)}), \tag{3.31}$$

where n_i is the number of particles in the new set $\{s_k^{*(j)}\}$, which are copied from one particle $s_k^{(i)}$ of the old set. In terms of consistency it must be proved that the resampled density converges to the original density for $N \to \infty$. That is, to fulfill

$$E\left[\left(\int g(s)\, p_N(s)\, ds - \int g(s)\, \hat{p}_N(s)\, ds\right)^2\right] \longrightarrow 0 \quad \text{for} \quad N \to \infty \tag{3.32}$$

for any function $g(\cdot)$ [DoDFG01], [CrDo02]. There are many different resampling methods, four basic ones are commonly used, they are:

Multinomial Resampling

Multinomial resampling was introduced by Gordon et al. [GoSS93] as the first method of overcoming weight degeneracy. The method is as follows:

- N uniform random numbers $u_j \in [0, 1)$ are generated
- u_j is used to select $s_k^{*(i)}$ as follows

$$s_k^{*(j)} = s_k(F^{-1}(u_j)) \quad \text{for} \quad j = \{1, 2, \ldots, N\}$$

where F^{-1} stands for the inverse of the cumulative probability distribution of the normalized particle weights.

Stratified Resampling

Stratified resampling was introduced by Kitagawa [Kita96] with the intention of reducing the variance between the original and resampled density. The method is as follows:

- N ordered random numbers $u_j = \frac{(j-1)+\tilde{u}_j}{N}$ with $\tilde{u}_j \in [0, 1)$ are generated
- u_j is used to select $s_k^{*(i)}$ in the same way as in multinomial resampling

Systematic Resampling

Systematic resampling was first mentioned by Witley [Whit94] and later introduced to Particle filters by Carpenter et al. [CaCF99]. The method is as follows:

- N ordered uniform distributed numbers $u_j = \frac{(j-1)+\tilde{u}}{N}$ with $\tilde{u} \in [0, 1)$ are generated
- u_j is used to select $s_k^{*(i)}$ in the same way as in multinomial resampling

Residual Resampling

Residual resampling was introduced by Liu et al. [LiCh98]. The method is as follows:

- $N_i = \lfloor N\tilde{w}_k^{(i)} \rfloor$ duplicates of $s_k^{(i)}$ are copied to the new set for $i = \{1, 2, \ldots, N\}$
- the still empty space (because of floor-norming) is filled up by using one of the earlier methods.

A comprehensive comparison of the basic resampling methods is given in [HoSG06] and [DCPP05]. Based on these publications, it was decided to use the stratified resampling method because of its good performance and minimized variance.

4. Particle Filter Design

So far, the Bayesian estimator was introduced and its implementation, the Particle filter, was described in general. In this chapter, based on the previous theory, the lane detection and tracking (LDT) algorithm is designed in respect to the single steps of the Particle filter.

First, a model that represents the automotive scene, refereed to as *road model*, is described. Then, a model to estimate the subsequent road model state, based on the vehicle's movement and its current state, called *state evolution model*, is given. This is then followed by the *observation model* which describes the observation of the automotive scene.

These three sections provide the basic tools of the LDT algorithm. Possible orders of applying these tools is analyzed and the optimal order is then defined as the *data flow* structure. Based on the data flow structure, a method to compare the estimated road model state and the actual observation of the automotive scene is given. It is refereed to as *particle evaluation*. These steps combined together form the Particle filter as it is described in the previous chapter.

Finally, a major issue of the Particle filtering, called *sample impoverishment* is discussed in the context of LDT systems in detail.

4.1 State Space

The fundamental part of state estimation is the state space $\mathbb{R}^n$ in which the system operates. The state space is spanned by those parameters that represent the system's status. For LDT systems these parameters are given by the road model which describes the road curvature relative to the vehicle position and bearing as well as its camera set-up.

A great variety of road models for LDT systems have been proposed. The best choice of the road model depends on the type of system and intended environment in which the system will be used.

Road models for highway and rural road applications are commonly based on linear [ApZe03, GWZX+08, STHS08], parabolic [SaST06], or cubic curvatures [SoTa01, McTr06, FrLK07, DNMT08, LoFS09].

Models designed for urban environment require higher flexibility and thus are usually based on piecewise linear curvature [LSBL+08] or curvatures based on different spline types [Aly08].

Most of these models assume a flat road surface which will be referred to as *flat ground plane*. A few systems have also introduced a vertical curvature rather than assuming a flat ground plane [DiMy92, NSGD+04] these models fit better to environments with high height variations.

The road model used in this system extends the model introduced by Southall [SoTa01] and is described in the following paragraphs in detail.

Road Shape

The road is assumed to lie on a flat ground plane and its shape is modeled by a polynomial of 3^{rd} order. The lane's center line is given by

$$Y_c(z) = Y_0 + tan(\epsilon_w)Z + \frac{C_0}{2}Z^2 + \frac{C_1}{6}Z^3, \tag{4.1}$$

where Y_0 is the lateral displacement between the lane center and the vehicles position, $tan(\epsilon_w)$ is the relative vehicle bearing, C_0 and C_1 are the 2^{nd} and 3^{rd} order coefficients describing the road's curvature.

Relative Pitch Angle and Lane Width

The assumption of a fixed ground plane yields a very rigid and fault-prone road model since in real world the road surface also has vertical curvature. However, this assumption simplifies the camera modeling that is described in section 4.3.1. Importantly, the divergence between reality and model can be minimized by introducing the vertical curvature to the state space. This is done to a 1^{st} order approximation by introducing the relative pitch angle α_w, which is the angle between the assumed fixed ground plane and the real ground plane, to the state space. The spatial set-ups, both horizontal and vertical, are given in Figure 4.1.

In this implementation, the variation of the pitch angle is detected by a brute force approach. This means that in the evolution process, there is no contribution which gives an estimate of the next angle value. Instead, a certain region in the state space

is searched. This requires a higher number of additional particles and therefore a higher computational effort. However, it resulted in good performance, as shown in section 6.2.2 and is a good start.

Note that, in contrast the system proposed by Southall et al. [SoTa01] detects the camera movement via visual discrepancy in the image which allows to reduce the number of particles. Danescu et al. [DNMT08] investigates a complete different approach and uses stereo vision information to compute the pitch angle in advance. In this way the pitch angle does not need to be introduced to the state space.

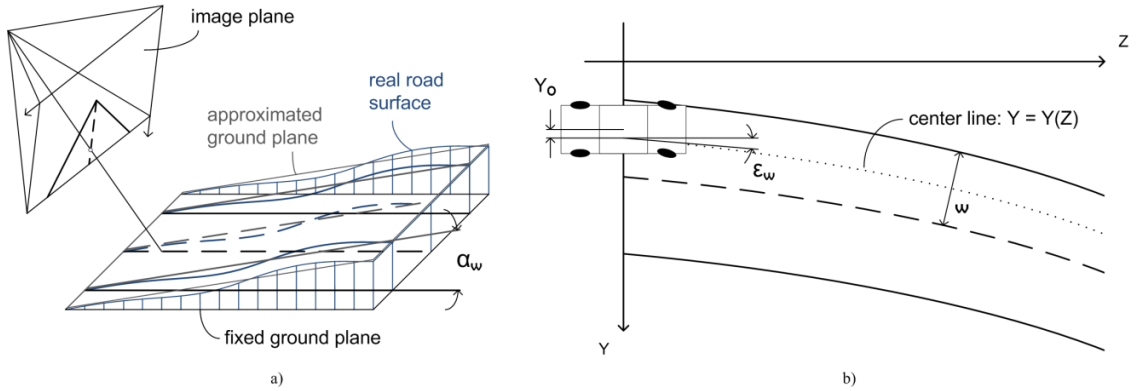

Figure 4.1: *a) road scene with vertical curvature and* 1^{st} *order approximation, b) road model with horizontal curvature*

Considering the pitch angle and introducing the lane width W, the left and right lane boundaries are approximated[1] by

$$\begin{aligned} Y_l &= Y_c - \frac{W}{2} - tan(\alpha_w)Z, \\ Y_r &= Y_c + \frac{W}{2} + tan(\alpha_w)Z. \end{aligned} \tag{4.2}$$

The state space spanned by the parameters described so far is conceptually the same as given by Southall et al., the state vector is

$$\mathbf{s}_s = \begin{bmatrix} Y_0 & \epsilon_w & C_0 & C_1 & W & \alpha_w \end{bmatrix}^T. \tag{4.3}$$

Stabilizer Lines

Unfortunately, although this model works well for highway scenes, it lacks robustness in urban environments, characterized by frequent occlusions and weak road features. It is therefore necessary to improve the system's robustness in urban situations by adapting the model to the urban environment.

Usually, the lane is bordered by several edge structures which lie in parallel to the lane. These might be lane markings of one or more additional lanes, or just the infrastructure of the surroundings. By introducing additional *stabilizer lines*, one on the left and one on the right side of the lane, the parallel edge structures are taken into account, resulting in improved robustness. The additional lines are defined by

[1] It is an approximation because due to perspective transform the term $tan(\alpha_w)$ is not exactly representing the relative pitch angle but serves as a good approximate.

the distance that is relative to the lane width, from the left or right lane border respectively, that is

$$\begin{aligned} Y_{sl} &= Y_l - W \cdot D_l, \\ Y_{sr} &= Y_c + W \cdot D_r, \end{aligned} \tag{4.4}$$

where the stabilizer line distance is given as ratios of the lane width D_l and D_r.

The State Vector

Finally, the state space is defined by the parameters: $Y_0, \epsilon_w, C_0, C_1, W, D_l, D_r$ and α_w. To improve the calculation performance, the actual state space is spanned by the terms as they are processed, for example instead of the angles ϵ_w and α_w their tangents are used. The state vector is then given by

$$\mathbf{s} = \begin{bmatrix} Y_0 & tan(\epsilon_w) & C_0 & C_1 & \frac{W}{2} & D_l & D_r & tan(\alpha_w) \end{bmatrix}^T. \tag{4.5}$$

4.2 State Evolution Model

Step 1 of the Particle filter as described in section 3.3 requires the transition density $p(s_k|s_{k-1})$ which is calculated by the previous posterior particle distribution $p(s_{k-1}|Y_{k-1})$ and a model which describes the transition of the system state from time index $k-1$ to k given some known input signals. This model is called *state evolution model* and is formally written as

$$\mathbf{s}_k = \mathbf{f}_k(\mathbf{s}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}).$$

In this application, the system input vector $\mathbf{u}$ consists of the vehicle's velocity and yaw rate, these are so called *motion parameters*. The following paragraph describes how the state transition is determined by the vehicle's motion behavior.

Taking into account the vehicle's position in world coordinates and its motion parameters as time-discrete quantities, the position change within two subsequent frames and system states respectively is approximated by

$$\begin{aligned} \Delta Z_k &= v_k \cdot \Delta t \cdot cos\left(\epsilon_{w,k} + \frac{\Delta\epsilon_{w,k}}{2}\right) \\ \Delta Y_k &= v_k \cdot \Delta t \cdot sin\left(\epsilon_{w,k} + \frac{\Delta\epsilon_{w,k}}{2}\right) \end{aligned} \tag{4.6}$$

where Δt is the time step, $\Delta\epsilon_{w,k} = \psi_{w,k} \cdot \Delta t$ is the time-discrete change of the vehicle's heading direction and v_k is the time-discrete velocity. The state evolution model can be rewritten as

$$\mathbf{s}_k = \mathbf{f}_k(\mathbf{s}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) = \mathbf{A}_{k-1}(\mathbf{u}_{k-1}) \cdot \mathbf{s}_{k-1} + \mathbf{B}_{k-1}(\mathbf{u}_{k-1}) + \mathbf{w}_{k-1}. \tag{4.7}$$

By combining the equations 4.1 and 4.6, where $\mathbf{u}_k = \begin{bmatrix} v_k & \psi_{w,k} \end{bmatrix}$, the transition matrices $\mathbf{A}_{k-1}(\mathbf{u}_{k-1})$ and $\mathbf{B}_{k-1}(\mathbf{u}_{k-1})$ are given by

$$\mathbf{A}_{k-1}(\mathbf{u}_{k-1}) = \begin{bmatrix} \mathbf{A}_{M,k} & \mathbf{0}_{4x4} \\ \mathbf{0}_{4x4} & \mathbf{I}_{4x4} \end{bmatrix} \quad \text{and} \quad \mathbf{B}_{k-1}(\mathbf{u}_{k-1}) = \begin{bmatrix} 0 & \psi_{w,k} \cdot \Delta t & \mathbf{0}_5^T \end{bmatrix}^T \tag{4.8}$$

$$\text{where } \mathbf{A}_{M,k} = \begin{bmatrix} 1 & \Delta Z & \frac{\Delta Z^2}{2} & \frac{\Delta Z^3}{6} \\ 0 & 1 & \Delta Z & \frac{\Delta Z^2}{2} \\ 0 & 0 & 1 & \Delta Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

and $\mathbf{w}_k$ represents uncertainties in the evolution model. The uncertainties are introduced by erroneous measurements of the vehicles motion parameters and the time-discretization of the model itself.

4.3 Observation Model

So far, the state space to describe the real world and the evolution of the state based on the motion parameters has been given. From here, it is essential to compare the estimated system state with some observations of reality. The relation between the system state and the observation is given by the observation model.

For example, a system with a straightforward relation between the measured quantity $\mathbf{y}_k$ and the system's state variable $\mathbf{s}_k$, the relation can be expressed by

$$\mathbf{y}_k = \mathbf{h}(\mathbf{s}_k, \mathbf{v}_k)$$

where $\mathbf{h}(\cdot)$ is a mathematical description of the observation model, which is characterized by measure uncertainties $\mathbf{v}_k$ given as noise vector with zero mean Gaussian distribution and covariance $\mathbf{R}$. Considering a specific measurement $\mathbf{y}_k^*$ as observation of the reality, the difference between proposed system state $\mathbf{s}_k$ and the reality is then given by

$$v = \mathbf{y}_k^* - \mathbf{y}_k = \mathbf{y}_k^* - \mathbf{h}(\mathbf{s}_k, \mathbf{v}_k).$$

For systems with complex measuring processes and data preprocessing, such as applications in computer vision, a more convenient method is to split the model in several parts, which then can be combined in whatever way seems to be appropriate.

The key point is that a measure of the difference between observation in world space and the hypothesis in state space can be calculated.

Camera Model

The camera model $\mathbf{h}_c(\cdot)$ describes the process of acquiring an image. A detailed model description is given in section 4.3.1. The image $\mathbf{y}_k^*$, which is considered as specific measurement, is given by

$$\mathbf{y}_k^* = \mathbf{h}_c(\text{real world}, \mathbf{v}_{k,c}), \tag{4.10}$$

where $\mathbf{v}_{k,c}$ represents disturbances of the imaging system introduced by lens distortions, sensitivity variances of the light sensitive element, spatial quantization due to finite element segments and finally by reduction of information from $\mathbb{R}^3$ to $\mathbb{R}^2$.

From here two different data processing steps can be applied: feature extraction and remapping of the image in the world space. The sequential order of these two steps is left open at this point, and is treated in section 4.4.

Feature Extraction

The image $\mathbf{y}_k$, given either in image space or world space, is processed by different filter algorithms, given in section 4.3.2. The resulting image is then called *feature map*. This is formally expressed by

$$\tilde{\mathbf{y}}_k = \mathbf{h}_f(\mathbf{y}_k, \mathbf{v}_{k,f}), \tag{4.11}$$

where $\mathbf{v}_{k,f}$ represents uncertainty of the filters for feature extraction. The reason for this is that feature extraction is based on an operation in two dimensional space and thus is not perfectly suited for acquiring features which originate in three dimensional space. For example, features which do not have any meaning in three dimensions might be extracted and some relevant features might be left out due to thresholding.

Inverse Perspective Mapping

The image $\mathbf{y}_k$, given either as unprocessed or preprocessed data, is projected to world space for further image processing or evaluation purpose. The inverse perspective mapping is described in section 4.3.3 and formally expressed by

$$\mathbf{Y}_k = \mathbf{h}_p(\mathbf{y}_k, \mathbf{v}_{k,p}), \tag{4.12}$$

where $\mathbf{v}_{k,p}$ represents uncertainty of the projection in the world space. The reason for this is that the projection of the image in the world space is a reconstruction of the real world scene, hence a back projection of $\mathbb{R}^2$ in $\mathbb{R}^3$. The missing information needs to be assumed, thus generating projection artifacts. Due to quantization of the image data, the data needs to be interpolated, which arises with further uncertainties.

State Vector Projection

At this point, the observation $\mathbf{y}_k^*$ is given either in image space or in world space, therefore, finally, the state hypothesis $\mathbf{s}_k$ must be projected in the same space so that a difference measure can be computed. The projection is given by

$$\tilde{\mathbf{y}}_{k,s} = \mathbf{h}_s(\mathbf{s}_k, \mathbf{v}_{k,s}), \tag{4.13}$$

where $\mathbf{v}_{k,s}$ represents uncertainty of the projection of the state hypothesis in the evaluation space. This uncertainty is introduced by quantization of the evaluation space.

The following subsections describe the model functions in detail. A comprehensive analysis of the data flow is then given afterwards in section 4.4.

4.3.1 Camera Model

A camera's perspective geometry can be described sufficiently using the *pinhole model*. The pinhole model is an idealized model, which limits the incident light per image pixel to a single light beam. The abstract description of the *imaging function* is, as already stated, given by

$$\mathbf{y}_k^* = \mathbf{h}_c(\text{real world}, \mathbf{v}_{k,c}).$$

The simplicity of this model requires that non-linear distortions, which occur due to imperfections of lenses, sensor element and quantization are treated by introducing a variance $\mathbf{v}_{k,r}$ to the observation model.

Figure 4.2 shows the principle model set-up with its *image plane* I and *optical center* O. All light rays that are captured by the camera will be focused at the optical center. The distance between optical center and image plane is called *focal length* f.

Assuming an orthonormal system of coordinates in the image plane, centered at the *principle point* C, a three-dimensional orthonormal system of coordinates, called the *camera coordinate system*, can be defined. The axis X_C and Y_C are parallel to the image plane and the third axis Z_C lies parallel to the principle axis with the optical center as its origin.

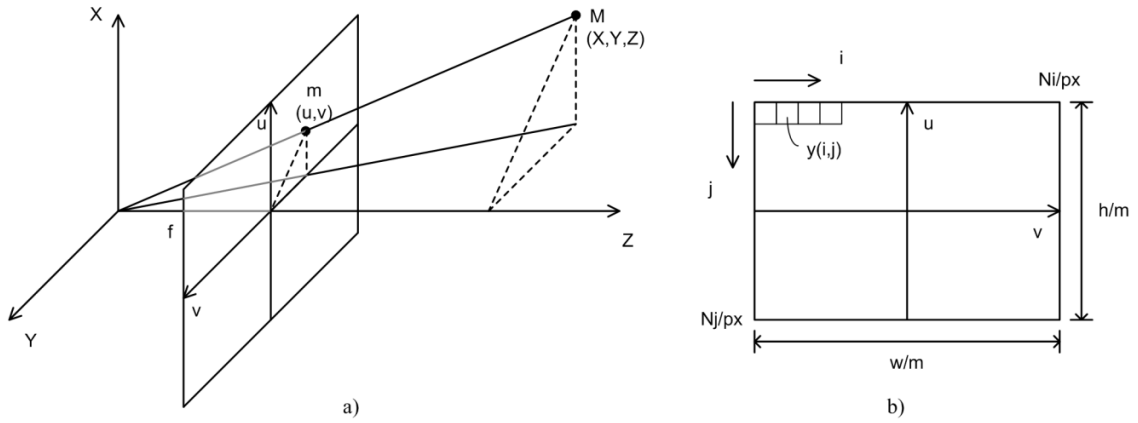

Figure 4.2: *pin hole camera set-up (a) and details of sensor element (b)*

Taking a point M, $[X_C, Y_C, X_C]^T$, in the camera coordinate system, its light ray will go through the optical center and pierces the image plane at point m, $[u, v]^T$. The relation between the coordinates of M and m is given by Thales theorem:

$$\frac{f}{Z_C} = \frac{u}{X_C} = \frac{v}{Y_C} \tag{4.14}$$

which yields

$$u = f \cdot \frac{X_C}{Z_C}, \qquad v = f \cdot \frac{Y_C}{Z_C}. \tag{4.15}$$

However, the relation between the projection m and its object M comes with a certain ambiguity. This is, because any point $[\lambda X_C, \lambda Y_C, \lambda X_C]^T$ along the optical ray could project onto m. This ambiguity is best described by considering $[\lambda X_C, \lambda Y_C, \lambda X_C]^T$ and projection m to be projective coordinates[2] of the optical ray in object and image space.

$$m_p = \begin{bmatrix} u_p \\ v_p \\ \psi \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{K}_0} \begin{bmatrix} \Psi X_{C,p} \\ \Psi Y_{C,p} \\ \Psi Z_{C,p} \\ \Psi \end{bmatrix} = \mathbf{K}_0 M_p \tag{4.16}$$

The Euclidean coordinates are related to projective coordinates by: $u = u_p/\psi$, $v = v_p/\psi$, $X_C = X_{C,p}/\Psi$, $Y_C = Y_{C,p}/\Psi$, $Z_C = Z_{C,p}/\Psi$ with $\Psi = 1$.

In general the image coordinate system is defined by the photo sensitive element and its pixel resolution (which might be different for each direction):

$$j = -m_u u + m_u t_u, \qquad i = m_v v + m_v t_v \tag{4.17}$$

[2]The relation between Euclidean and perspective space is detailed in appendix A.1.

where m_u and m_v are the pixel resolution in pixel/meter units and t_u and t_v are the actual displacement of the principal axis to the image center in meter units. The sign of the m_u factor is necessary since the direction of the pixel index is opposite to that of the actual coordinate system of the sensitive area.

Furthermore, it is desired to relate the object coordinates to a freely defined coordinate system which is relative to the camera coordinate system. Therefore, the general form of the projection matrix is:

$$\mathbf{K} = \underbrace{\begin{bmatrix} \alpha_u & s & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{K}_i} \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix}}_{\mathbf{K}_e} \tag{4.18}$$

where

- $\mathbf{K}_i$ describes the characteristics of the imaging system. Its five entries are called *intrinsic parameters*. The focal length f is represented by α_u and α_v in *pixel* units so that $\alpha_u = -m_u f$ and $\alpha_v = m_v f$ respectively. The displacement in each direction is also expressed in *pixel* units by $u_0 = m_u t_u$ and $v_0 = m_v t_v$ likewise. The skew parameter s is usually zero except in some particular imaging systems.
- $\mathbf{K}_e$ describes the relation between the chosen world coordinate system and the camera coordinate system. Its entries are called *extrinsic parameters*. It is built up on a translation vector $\mathbf{t}$ and the rotation matrix $\mathbf{R}(\alpha, \beta, \gamma)$ providing three dimensions of rotation.

All in all, the projection m in the image plane of an object M, given in real world coordinates, can be calculated by

$$m_p = \mathbf{K} M_p, \qquad m = \frac{m_p}{m_{p,3}}. \tag{4.19}$$

The Homography

Capturing an image is a projective transformation, $\mathbb{R}^3 \rightarrow \mathbb{R}^2$, that is, information is reduced from three-dimensional to two-dimensional space, and, as such, is not directly invertible. The reconstruction of the captured scene using its image, $\mathbb{R}^2 \rightarrow \mathbb{R}^3$ is therefore ambiguous and needs additional spatial information or assumptions.

If no further information (for instance sensor data or depth information from stereo vision) is available, a commonly used technique is to assume a flat ground plane. In this case the reconstruction is reduced to a 2D-to-2D correspondence between the ground plane Π and the image plane I. The ground plane represents the road surface and is aligned with the world coordinate system as illustrated in Figure 4.3. The image plane can have any possible position in the world coordinate system.

The 2D-to-2D correspondence is called *homography* of $\mathbb{P}^2$ and is described by a 3x3 matrix $\mathbf{H}$. Generally speaking, a homography is any projective transformation $\mathbf{H}$ which is linear and invertible, that is a (n+1) x (n+1) non-singular matrix for $\mathbb{P}^n$.

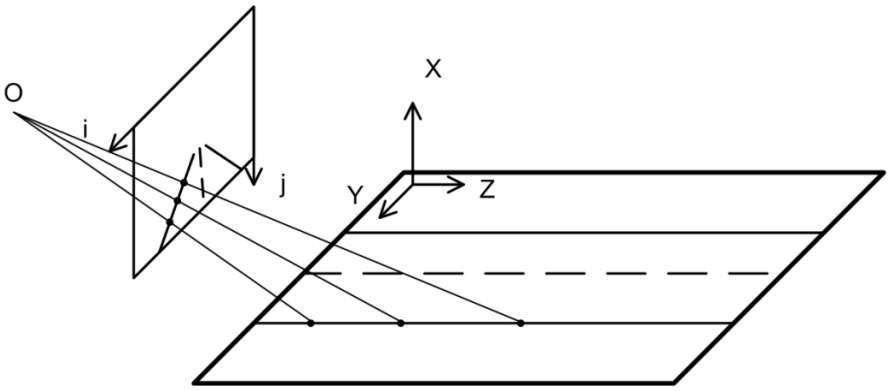

Figure 4.3: *homography between ground plane in space and image plane*

Choosing the world coordinate system so that the first two axes define the ground plane, the homography is defined by the projection matrix K and can be viewed as transformation between two spaces $\mathbb{P}^2$.

$$\underbrace{\begin{bmatrix} u_p \\ v_p \\ \psi \end{bmatrix}}_{m'_p} = \mathbf{K} \underbrace{\begin{bmatrix} 0 \\ Y_p \\ Z_p \\ \Psi \end{bmatrix}}_{M_p} = \underbrace{\begin{bmatrix} K_{12} & K_{13} & K_{14} \\ K_{22} & K_{23} & K_{24} \\ K_{32} & K_{23} & K_{34} \end{bmatrix}}_{\mathbf{H}_w} \underbrace{\begin{bmatrix} Y_p \\ Z_p \\ \Psi \end{bmatrix}}_{m_p} \tag{4.20}$$

Finally, the projection m' of the point m in ground plane, given in real world coordinates, can be calculated by

$$m'_p = \mathbf{H}_w m_p, \qquad m' = \frac{m'_p}{m'_{p,3}}, \qquad m' = \begin{bmatrix} i \\ j \end{bmatrix}. \tag{4.21}$$

4.3.2 Feature Extraction

After acquiring the image, the relevant information about the automotive scene needs to be extracted. The information must then be given in a way that allows to compute a probabilistic difference measure between observation and state variables.

The relevant information in automotive scenes is therefore identified in the next paragraph, followed by a detailed description of different feature extraction methods.

Firstly, the road area needs to be separated from the background, where background in this context is defined as every spatial area not belonging to the road area. Road features are points, lines or areas of interest that help to separate the road from its background. The following types of road features have been identified:

Lane markings are usually designed to have a high contrast. That is, there is a high intensity gradient at the intersection between the road surface and lane marking. Markings occur in various shapes, including solid lines, dashed lines, double lines and Bott's dots, as well as have variable width, dash distance and dash length.

Road-curb borders are structural features in $\mathbb{R}^3$ that often occur in urban environments. These are difficult to extract without depth measurement system such as stereo vision. However, they are detectable due to intensity gradients generated by light-shadowing effects, but the certainty is lower than that of lane markings.

Road-trimming borders are structural features that often occur on rural roads. The road-trimming border produces a gradient intensity due to different road surface color and background color. Furthermore the background texture usually has a higher local entropy than the road surface.

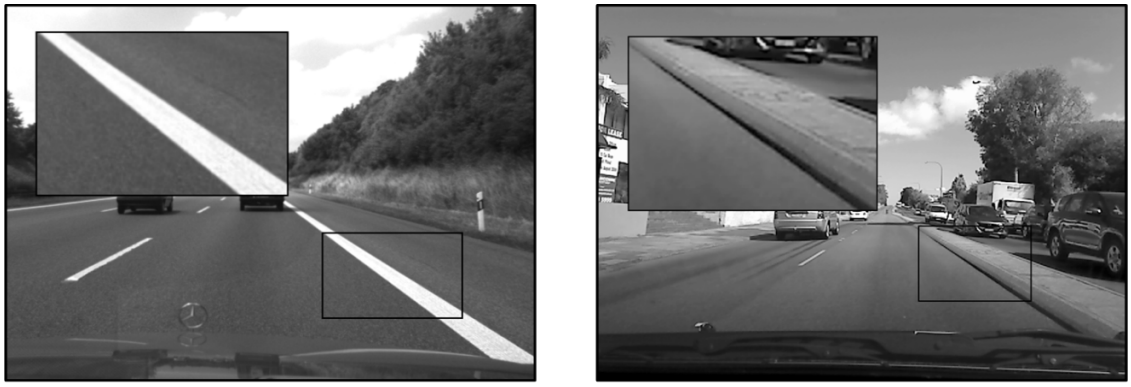

Figure 4.4: *example of typical road features, a) from Daimler sequence No. 2 - lane marking, b) from UWA sequence No. 1 - road-curb border*

This work focuses on highway and major urban road environments, which means that lane markings and road-curb borders need to be detected. Since this thesis focuses on Particle filtering, the feature extraction problem is not treated in a great deal of depth. Sobel filtering yielded reasonably good results, so the algorithm mainly relies on the gradient magnitude and direction information. Unfortunately, one consequence of using this approach was that the algorithm lacks robustness in situations with weak road features and a lot of clutter present.

Intensity Gradient Filtering

As stated above, lane markings and road-curb borders generate intensity gradients in the image of the road scene. To extract these intensity gradients it is convenient to apply the two-dimensional, gradient operator to the intensity image $\mathbf{y}$, that is

$$\nabla \mathbf{y}(j,i) = \frac{\partial \mathbf{y}}{\partial j} e_j + \frac{\partial \mathbf{y}}{\partial i} e_i = \mathbf{y}_j e_j + \mathbf{y}_i e_i \tag{4.22}$$

where e_j and e_i are the unit vectors in j and i direction.

The gradient magnitude gives a measure of the likelihood of observing a road feature which will be further discussed in section 4.5.1. For now though, the gradient magnitude image is given by

$$M(j,i) = \sqrt{\mathbf{y}_j^2 + \mathbf{y}_i^2}. \tag{4.23}$$

The gradient direction gives an estimate of the spatial direction of the believed feature and is given by

$$\Theta(j,i) = arctan\left(\frac{\mathbf{y}_j}{\mathbf{y}_i}\right). \tag{4.24}$$

Since the image is given in discretized form, the above needs to be approximated by finite differences, that is

$$\frac{\partial \mathbf{y}}{\partial j}[j,i] = \frac{\mathbf{y}[j+1,i] - \mathbf{y}[j-1,i]}{\Delta j}, \tag{4.25}$$

$$\frac{\partial \mathbf{y}}{\partial i}[j,i] = \frac{\mathbf{y}[j,i+1] - \mathbf{y}[j,i-1]}{\Delta i}, \tag{4.26}$$

where Δj and Δi is chosen to be equal 1 pixel unit.

These gradients are usually aligned to form a certain curvature, generally called edges. To extract the edge information, several linear aligned pixels are summed to calculate the intensity gradient.

To improve robustness against image noise a weighted average smoothing is usually performed in the orthogonal direction. Common edge filter kernels are *Perwitt* or *Sobel*. Figure 4.5 illustrates the Sobel filter applied on an automotive scene.

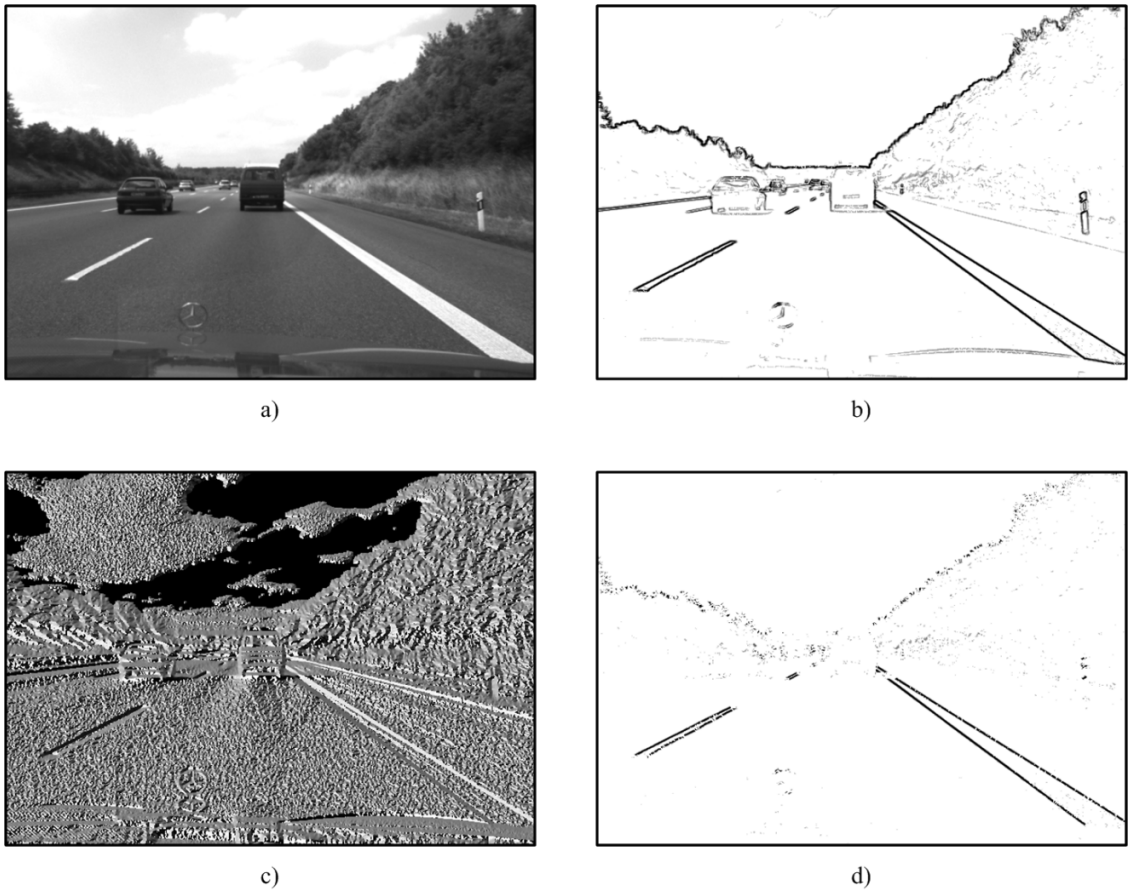

Figure 4.5: *example of intensity gradient filtering using Sobel operator; a) intensity image from Daimler sequence No. 2; b) gradient magnitude image (inverted); c) gradient direction image; d) filtered magnitude image (inverted) based on direction of lane hypothesis*

Entropy Filtering

Entropy is a statistical measure of randomness, and in image processing, entropy quantifies the information content of an image. The entropy of an intensity distribution $p = \{p_1, p_2, p_3, \ldots, p_n\}$, also referred to as *image histogram*, is defined as

$$H = -\sum_i p_i \, log(p_i), \tag{4.27}$$

where *log* is the logarithm to the base 2.

The entropy measure can also be given for localized area in the image. If these areas are defined as squares of odd size, every pixel can be assigned a certain entropy value which corresponds to the information content of its neighboring pixels.

For areas with homogeneous pixel intensities, as e.g. road surfaces, a low information content is expected. On the other hand, areas with high randomness, for example meadow area, the information content is expected to be high. Figure 4.6 illustrates the entropy filter applied on an automotive scene.

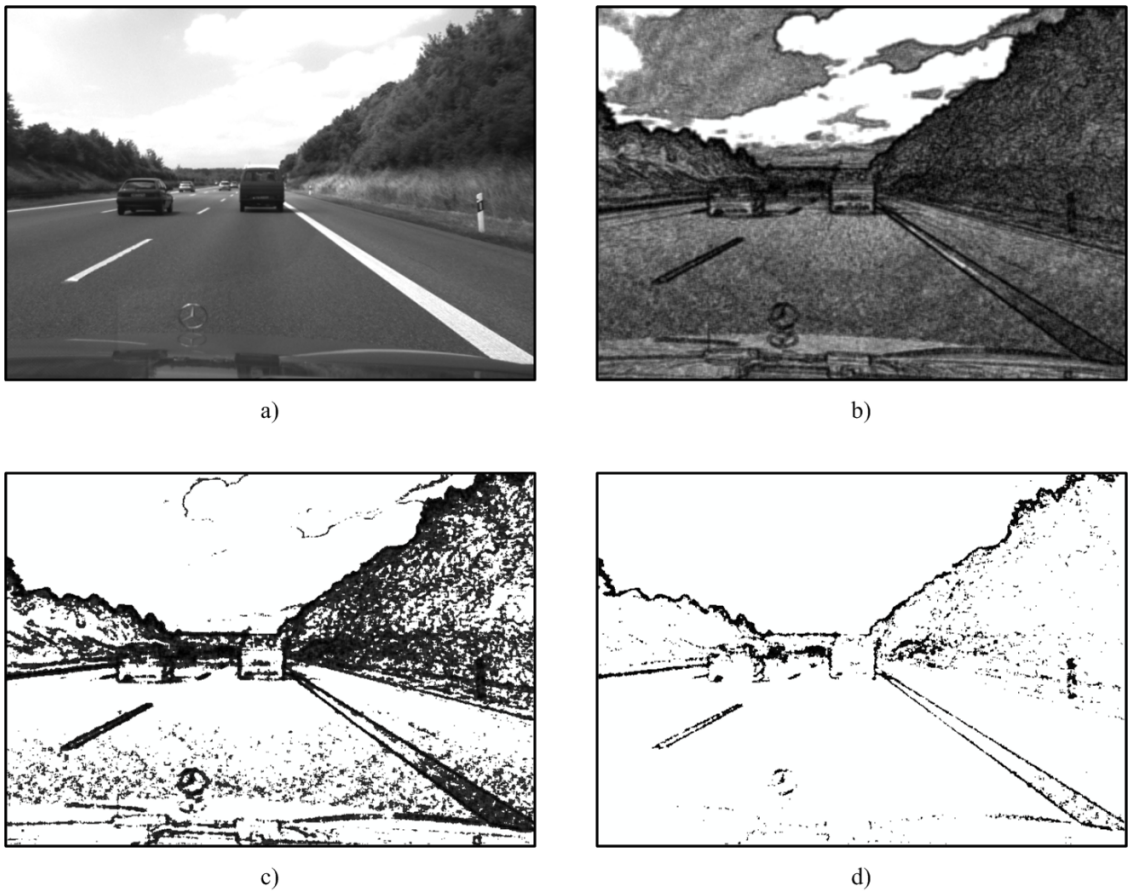

Figure 4.6: *example of local entropy filtering; a) intensity image from Daimler sequence No. 2; b) local entropy image (inverted) with kernel size 5x5; c) thresholded local entropy image (inverted) up to 75 percent; d) thresholded local entropy image (inverted) up to 90 percent*

4.3.3 Inverse Perspective Mapping

The transformation from image space to world space is called *inverse perspective mapping* (IPM). As already state, the transform is formally written as

$$\tilde{\mathbf{y}}_{k,m} = \mathbf{h}_m(\mathbf{y}_k, \mathbf{v}_{k,m}).$$

IPM performs the reconstruction of the real world scene based on the camera model. In general this relation is then given by a perspective transform, that is the inverse transform of the imaging function described in section 4.3.1. When a flat ground plane is assumed, this relation is given by the homography of the two planes.

The method of transforming the image data back to world space was first introduced to lane detection tasks by Broggi et al. [Brog95].

As stated in section 4.3.1, the coordinate relation between world space and image space is defined by the homography matrix $\mathbf{H}_w$. To map the image in world space the indirect form of the transformation is used. That means, the coordinates in world space are swept through, and for each element the corresponding pixel coordinate is calculated and the pixel value copied. The coordinates are given by

$$m_p = \mathbf{H}_w^{-1} m'_p, \qquad m = \frac{m_p}{m_{p,3}}, \qquad m = \begin{bmatrix} Y \\ Z \end{bmatrix}. \tag{4.28}$$

which can also be expressed as

$$Y = \frac{i\mathbf{H}^{-1}_{w,11} + j\mathbf{H}^{-1}_{w,12} + \mathbf{H}^{-1}_{w,13}}{i\mathbf{H}^{-1}_{w,31} + j\mathbf{H}^{-1}_{w,32} + \mathbf{H}^{-1}_{w,33}}, \quad Z = \frac{i\mathbf{H}^{-1}_{w,21} + j\mathbf{H}^{-1}_{w,22} + \mathbf{H}^{-1}_{w,23}}{i\mathbf{H}^{-1}_{w,31} + j\mathbf{H}^{-1}_{w,32} + \mathbf{H}^{-1}_{w,33}}, \tag{4.29}$$

where Z and Y are the coordinates in world space and i and j are the coordinates in image space.

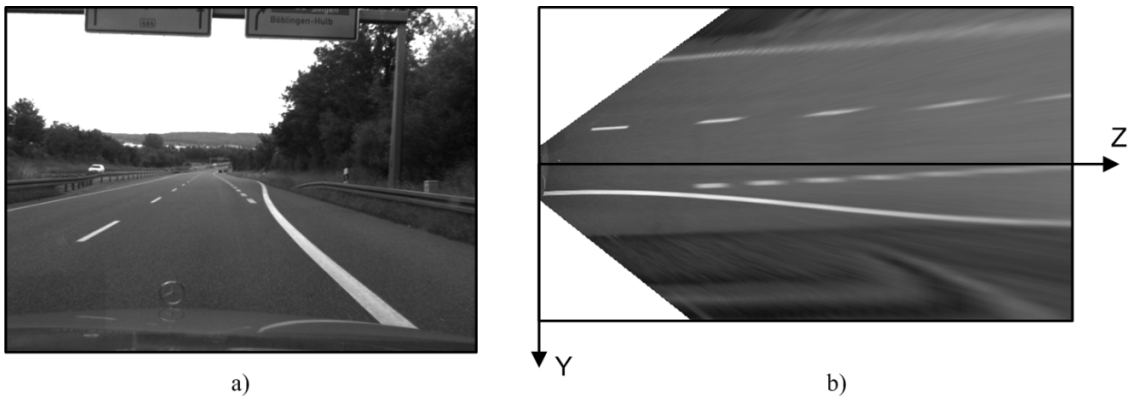

Figure 4.7: *example of inverse perspective mapped image; a) intensity image from Daimler sequence No. 1; b) re-mapped image*

4.3.4 State Vector Projection

Different state hypotheses need to be evaluated to calculate the likelihood function $p(y_k|s_k)$. Therefore, the state vector must be projected either in image space or in world space, depending on the space type of the given data.

The projection in world space is straightforward since the state vector provides parameters for the road model which is already located in world space. The projection $\tilde{\mathbf{y}}_{k,s} = \mathbf{h}_s(\mathbf{s}_k, \mathbf{v}_{k,s})$ is given for the left and right lane boundaries by

$$\begin{aligned} Y_l &= Y_c - W - tan(\alpha_w)Z, \\ Y_r &= Y_c + W + tan(\alpha_w)Z, \end{aligned} \tag{4.30}$$

and for the stabilizer lines by

$$\begin{aligned} Y_{sl} &= Y_c - W \cdot D_l \\ Y_{sr} &= Y_c + W \cdot D_r, \end{aligned} \tag{4.31}$$

where the center line is given by $Y_c = Y_0 + tan(\epsilon_w)Z + \frac{C_0}{2}Z^2 + \frac{C_1}{6}Z^3$.

The projection in image space just needs one more step, the transformation of each point according to the camera model. This is determined by the homography matrix H. The projection $\tilde{\mathbf{y}}_{k,s} = \mathbf{h}_s(\mathbf{s}_k, \mathbf{v}_{k,s})$ is then given for each line by

$$m'_p = Hm_p, \qquad m' = \frac{m'_p}{m'_{p,3}}, \qquad m' = \begin{bmatrix} i \\ j \end{bmatrix}, \tag{4.32}$$

which can also be expressed as

$$i = \frac{YH_{11} + ZH_{12} + H_{13}}{YH_{31} + ZH_{32} + H_{33}}, \quad j = \frac{YH_{21} + ZH_{22} + H_{23}}{YH_{31} + ZH_{32} + H_{33}}. \tag{4.33}$$

The projection of the state vector on data that contains any kind of road features, given either in image space or in world space, will be referred to as projection on a *feature map*. This is formally written as

$$F(z, y) \longleftarrow s_k, \tag{4.34}$$

A detailed analysis of possible data flow structures follows in the next section. Then the order in which the above methods are applied is defined.

4.4 Data Flow

For the sake of clarification, a brief review of the system's spaces will be given. Then different approaches to implement each step of the particle filter in different spaces will be considered.

World Space refers to an Euclidean space $\mathbb{R}^3$ that addresses the real world scene in which the vehicle is traveling.

State Space refers to a parameter space $\mathbb{R}^8$ that addresses the road model that is matched to the real world scene in either world coordinates or world map coordinates.

Image Space refers to a finite, discrete space $\mathbb{N}^2$ that addresses a projective image of the real world scene in which the vehicle is traveling. Data given in image space is referred to as *image*.

World Map Space refers to a finite, discrete space $\mathbb{N}^2$ that addresses a discretized ground plane of the real world scene in which the vehicle is traveling. Data given in world map space is referred to as *map*.

Four different data flow structures have been considered and analyzed in terms implementation performance and algorithm quality. An illustration of the different approaches is given in Figure 4.8. The focus lies in variations of the space in which the feature extraction and evaluation takes place. The approaches are:

Structure 1

1. capturing scene
2. feature extraction in image space
3. transforming feature images to world map space
4. evolving system state which refers to world map space
5. projecting hypothesis on feature maps
6. evaluating particles in world map space
7. calculating expected gradient direction value per hypothesis during evaluation (additional coordinate transformations)

Benefits: straightforward, no artifacts in gradient maps, accurate model matching

Drawbacks: high cost for transformation, feature extraction is faced perspective effects

Structure 2

1. capturing scene
2. transforming scene image straight away to world map space
3. feature extraction in world map space
4. evolving system state which refers to world map space

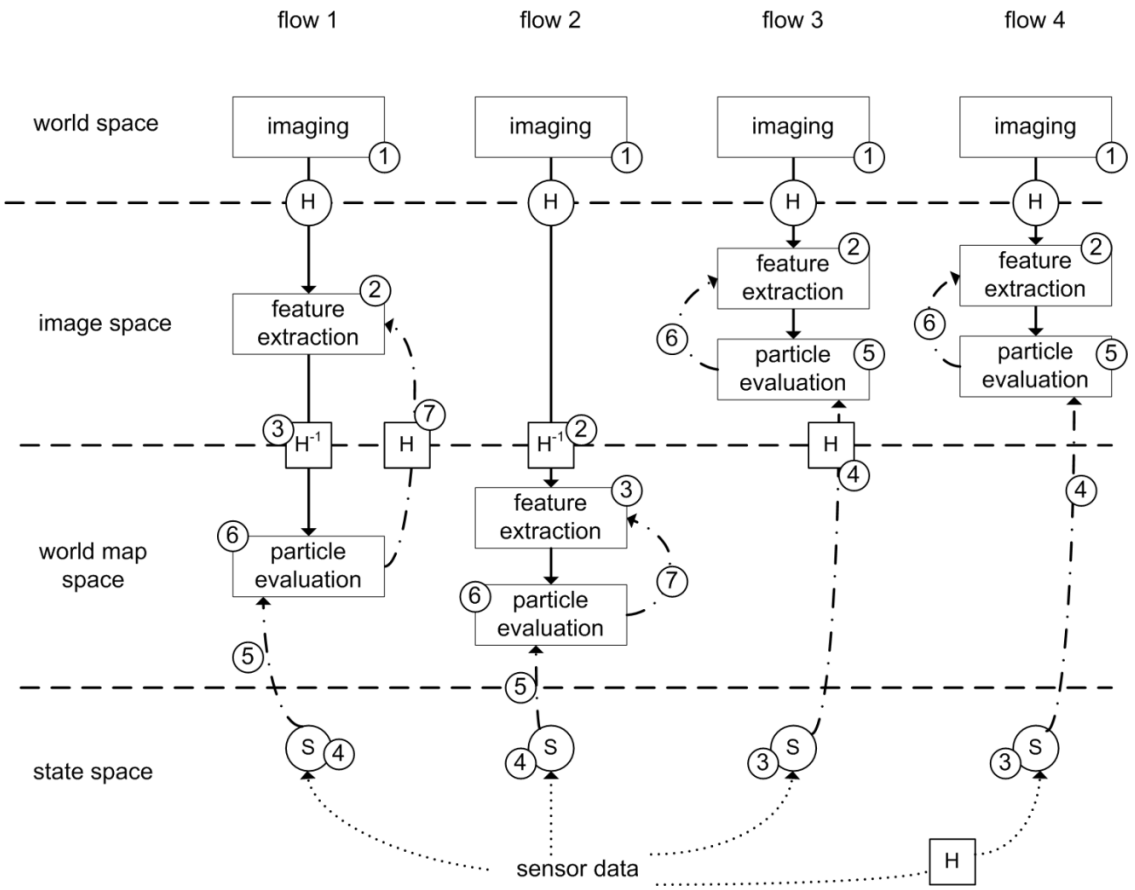

Figure 4.8: *four data flow structures*

5. projecting hypothesis on feature maps
6. evaluating particles in world map space
7. calculating expected gradient direction value per hypothesis during evaluation (no additional coordinate transformations)

Benefits: straight forward, no additional transformation for acquiring gradient direction, accurate model matching, feature extraction with almost no perspective effects

Drawbacks: artifacts in feature map are generated due to interpolation during image transform

Structure 3

1. capturing scene
2. feature extraction in image space
3. evolving system state which refers to world space
4. projecting hypothesis on feature images via coordinate transformation
5. evaluating particles in image space
6. calculating expected gradient direction value per hypothesis during evaluation (additional coordinate transformations)

Benefits: no artifacts in gradient maps, accurate model matching

Drawbacks: high cost for transformation, feature extraction needs to deal with perspective effects

Structure 4

1. capturing scene
2. feature extraction in image space
3. evolving system state which refers to image space
4. projecting hypothesis on feature images
5. evaluating particles in image space
6. calculating expected gradient direction value per hypothesis during evaluation

Benefits: no artifacts in gradient maps, accurate model matching, minimum number of coordinate transforms possible

Drawbacks: feature extraction is faced perspective effects, motion parameters must be transformed to image space, different road model which is based in image space does not reflect the real world, hence it is an abstract approximation

4.4.1 Summary

Structure 1, 2 and 3 have the great advantage that the road model is originated in real world coordinates and thus fits better the reality than a model in image space. However, Structure 4 provides a minimum number of coordinate transformations, which are quite expensive to process.

Structure 2, in contrast to the others, provides a feature extraction in world map space with reduced perspective effects. However artifacts generated by the IPM influence the feature extraction in world map space.

Structure 1 and 3 have about the same amount of coordinate transformations depending on world map size and number of particles, however structure 1 is easier to implement and, because the world map is static, also faster to implement.

All in all, structure 4 would be the fastest but least accurate implementation, whereas a combination of 1 and 2 would be the most accurate and flexible structure. Since the particle filter and feature extraction can be implemented in parallel structure easily, structure 1 and partially considerations of structure 2 were chosen for this project.

4.5 Particle Evaluation

This first paragraph explains the term *particle evaluation* by the help of a brief recall from section 3.3. It is then followed by a detailed description of its implementation for LDT algorithms.

The final aim is to compute the posterior distribution given by the recursive Bayesian estimator

$$p(s_k|Y_k) = \frac{p(y_k|s_k)\, p(s_k|Y_{k-1})}{\int p(y_k|s_k)\, p(s_k|Y_{k-1})\, ds_k}.$$

Since the true $p(s_k|Y_{k-1})$ is not available, it is substituted by the *proposal distribution* $p(s_k|s_{k-1})$ which is approximated by the particle distribution in state space. The particle distribution of time k is given by applying the evolution model to the particle distribution of time $k-1$, this is formally written as

$$\mathbf{s}_k^{(n)} = f(\mathbf{s}_{k-1}^{(n)}, \mathbf{u}_{k-1}, \mathbf{v}_{k-1}^{(n)}), \tag{4.35}$$

where $\mathbf{v}_{k-1}^{(n)}$ is a random vector drawn from a zero-mean Gaussian distribution with covariance Q. Therefore, one can write the following

$$p(s_k|Y_k) = \frac{p(y_k|s_k)\, p(s_k|s_{k-1})}{\int p(y_k|s_k)\, p(s_k|s_{k-1})\, ds_k} \equiv \frac{p(y_k|s_k^{(n)})}{\int p(y_k|s_k^{(n)})\, ds_k}. \tag{4.36}$$

The nominator term of the recursive Bayesian estimator is therefore approximated by

$$p(y_k|s_k) \cdot p(s_k|Y_{k-1}) \equiv p(y_k|s_k^{(n)}) = w_k^{(n)}, \tag{4.37}$$

where $w_k^{(n)}$ is the particle weight which reflects the relative likelihood of the particle representing the true lane parameters. This likelihood is determined by the difference between the observation and the lane hypothesis. The computation of the particle weight is called *particle evaluation*.

The relative likelihood of each particle is formulated in more detail by

$$w_k^{(n)} = P(y_k = \tilde{y}_k^*|s_k = s_k^{(n)}) = P(v_k^{(n)} = \tilde{y}_k^* - h_s(s_k^{(n)})), \tag{4.38}$$

where $v_k^{(n)}$ is the difference measure for each particle, $\tilde{y}_k^*$ is a specific observation given in either image space or world space and $h_s(s_k^{(n)})$ is the lane hypothesis transformed to either image space or world space.

The difference measure is a crucial design step, because it is the interconnection between the real world measurement and the system model. Ideally, both the simplicity of the system model, and its fit to the real world problem should be maximized. For this reason, it must be carefully chosen with respect to the particular application.

The relative particle likelihood is then given by

$$w_k^{(n)} \sim exp\left[-\frac{v_k^{(n)\,2}}{2\,\sigma^2}\right], \tag{4.39}$$

where σ^2 is the variance of the difference measure, which represents all uncertainties that are introduced by the observation model.

A more general form of the above is given by the product of several statistically independent observations, that is

$$w_k^{(n)} = \prod_{m=1}^{M} w_{k,m}^{(n)} = \prod_{m=1}^{M} P_m(y_{k,m} = \tilde{y}_{k,m}^* | s_k = s_k^{(n)}), \tag{4.40}$$

where P_m denotes the likelihood, given observation $m \in [1, 2, \ldots, M]$. The distance measures are individually designed for each feature map.

The first step of weight evaluation, independent of the type of feature data, is the projection of the lane hypothesis on the data according to

$$F(z, y) \longleftarrow s_k.$$

After the projection, the difference measures for every projected line, or the area spanned by the lines, are computed and finally logically linked together. Different methods of doing this will be discussed in the next sections.

Since all difference measures relate to a single particle at a single time, the time index k and the particle index (n) will be left out.

4.5.1 Line Difference Measure

The difference measure for the projected lines can be computed in different ways which are described in the following subsections. Before that, a description of how the feature map values are related to the likelihood of obtaining a road feature is described in the following paragraph.

The feature map elements $F[z, y] \in [0, 1, \ldots, 255]$ represent the likelihood of obtaining an actual road feature with respect to an expectation value μ_F.

For example, an element of the gradient magnitude map $F_M[z, y]$ would be most likely to be a road feature if its value is "255" [FrJo00, SKAD07, LoFS09]. This is based on the premise that lane markings have a intensity value of "255" and the road surface of "0". Therefore, the expectation value of the gradient magnitude μ_{F_M} is

$$\mu_{F_M} = 255.$$

Since the road markings are not necessarily in saturation, a representative variance $\sigma_{F_M}^2$ needs to be found. A different approach would be to adjust μ_{F_M} to a value that is more likely to be observed, or even to calculate the value dynamically to adapt the system to changing light conditions.

The same principle holds for the gradient direction map $F_\Theta[z, y]$. In this case, however, the expected value μ_{F_Θ} depends on the lane hypothesis. This relation is described in section 4.5.3.

Average Likelihood

To compute the difference measure v given by each line $(y_l, y_r, y_{sl}, y_{sr})$, the pixel values of the projected line are summed and normalized to the number of pixels. This method was proposed by several authors [SKAD07, SoTa01, DNMT08], and is formally written as

$$v_{line} = \frac{1}{N_z} \sum_{z=1}^{N_z} \left(F\left[z, y_{line}(z)\right] - \mu_F \right), \tag{4.41}$$

Average Likelihood of Connected Elements

Since road features usually appear as segmented pixel elements, it is necessary to apply an evaluation scheme that considers these structural aspects, and takes into account that each element must have neighbors in the direction of the line hypothesis.

The proposed evaluation scheme therefore introduces a logical AND-conjunction between several feature points. However, since any kind of thresholding is undesirable, the AND-conjunction is generalized by multiplication. The connected element selective difference measurement is given by

$$v_{line} = \frac{1}{N_z} \sum_{z=1}^{N_z} \left(\sqrt[M]{\prod_{m=0}^{M-1} F\left[z, y_{line}(z-m)\right]} - \mu_F \right), \tag{4.42}$$

where M is the number of elements which are expected to be connected and determines the size of minimum segment length.

This method is in principle similar to edge filtering, but provides a larger range for small features. Additionally, it can be used for feature maps which do not contain any direction information.

Average Likelihood with Equal Distribution

Structural analysis of road boundaries reveal that the road features are spread over the whole measuring range along the z-axis, whereas strong features introduced by occlusions are only present at parts of the z-axis.

By introducing this information the difference measure should gain further robustness against partial road occlusions like shadowing or vehicles.

Average Likelihood with Periodicity

The previous idea can be generalized by the introduction of Fourier analysis. The lane features that occur usually have some periodic behavior. Considering the Fourier spectrum of $F\left[z, y(z)\right]$, the different road features might be identified by: a) continuous lane marking, by constant spectral intensity corresponding to the likelihood, b) dashed lane marking, by frequency component with intensity corresponding to half the likelihood, assuming dashes are as long as gaps. More difficult cases are also determinable. c) Occlusions, by high frequency or low frequency components

The problem of the last two proposed methods is the detecting phase. During that phase, usually only small parts of the lane are detected and the rest is slowly approached. Unfortunately, no further investigation of these ideas could follow due to lack of time.

4.5.2 Conjunction of Several Lines

Once the cumulative likelihood is calculated for each line separately using either of the above methods, the relative hypothesis likelihood is calculated by conjugating the line likelihoods. This can be done in either of two ways:

The first, an OR-conjunction, yields a conjunction that is tolerant of missing data

$$v = \frac{1}{N} \sum_{line} v_{line}$$

The second, an AND-conjunction, yields a very rigid conjunction toward present data

$$v = \sqrt[N]{\prod_{line} v_{line}}$$

where $line \in \{l, r, sl, sr\}$ and N is the number of lines used.

The lane boundary lines should always be present together so that v_l and v_r are AND-conjugated. However, during the initialization phase it turned out to be convenient to allow an OR-conjugate so that the particles can converge to an estimate by "following a trace" which might be high likelihoods of one of the boundary lines. Therefore, the likelihood of both boundary lines are conjugated by

$$v_c = 0.9 \cdot \sqrt{v_l v_r} + 0.05 \cdot (v_l + v_r) \tag{4.43}$$

The left and right stabilizer lines do not need to be present at the same time, they should just increase the hypothesis likelihood if they are present. Therefore, the difference measures are OR-conjugated:

$$v_s = 0.5 \cdot (v_{sl} + v_{sr}) \tag{4.44}$$

The overall difference measure and the relative likelihood of the gradient intensity map is given by

$$v_{F_M} = v_c + v_s \quad \text{and} \quad w_{F_M} \sim exp\left[-\frac{v_{F_M}^2}{2\,\sigma_{F_M}^2}\right]. \tag{4.45}$$

4.5.3 Expectation Value of Gradient Direction

The gradient direction represents a direction estimate of the believed road feature. It is given by the feature map $F_\Theta \in [0, 1, \ldots, 255]$ representing the angles from 0° to 360°.

The expected direction value μ_{F_Θ} is simply given by the first derivative of the lane curvature, that is

$$\frac{dy}{dz} = tan(\epsilon) + c_0 z + \frac{c_1}{3} z^2, \tag{4.46}$$

where the expected angular value is given by

$$\mu_{F_\Theta} = arctan\left(\frac{dy}{dz}\right) = arctan\left(tan(\epsilon) + c_0 z + \frac{c_1}{3} z^2\right). \tag{4.47}$$

considering only linear road curvatures as a first order approximation it reduces to

$$\mu_{F_\Theta} = arctan\left(tan(\epsilon)\right) = \epsilon. \tag{4.48}$$

The expectation value given above can be used if the feature extraction was performed in world map space. For feature maps resulting from feature extraction in image space, the direction information must be transformed to image space.

Due to perspective effects, the coordinate transform needs to be performed for each single line. A straightforward method is to use two points to represent the line, transform the points to image space and calculate the angle estimate, based on the slope in image space, that is

$$\mu_{F_\Theta} = arctan\left(\frac{i_2 - i_1}{-(j_2 - j_1)}\right). \tag{4.49}$$

4.6 Sample Impoverishment

Sample impoverishment is an implementation issue that occurs when the region of state space in which the observation likelihood $p(y_k|s_k)$ has significant values does not overlap with the prior distribution $p(s_k|Y_{k-1})$. In this case, only a few distinct *a priori* particles become *a posteriori* particles. This eventually causes all particles to collapse to a single point in state space.

The same holds for multimodal likelihood functions. When taking first a unimodal function and let the particles distribute accordingly, after a few iterations the situation is as given in Figure 4.9 a). Assuming the likelihood is changing towards a bimodal distribution, even after several iterations, no particles will evolve into the second mode because of sample impoverishment. This is shown in Figure 4.9 b).

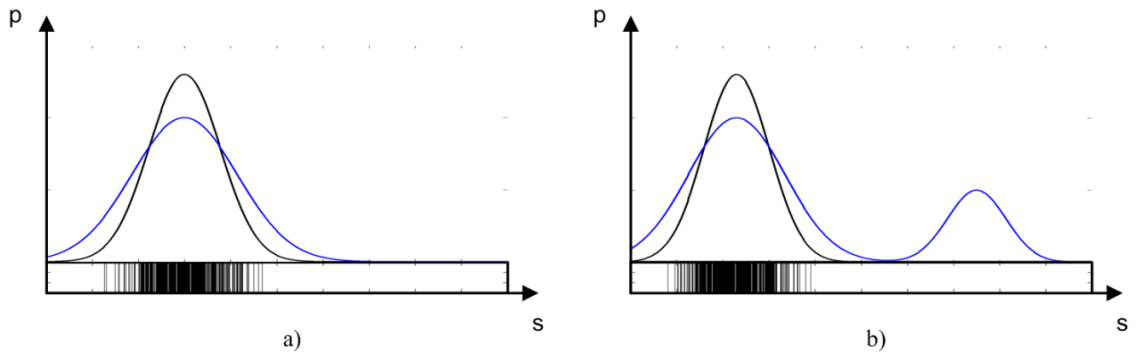

Figure 4.9: *sample impoverishment in one-dimensional state space where the black curve is the prior distribution $p(s_k|Y_{k-1})$ and the blue curve the observation likelihood $p(y_k|s_k)$, the bar underneath represents the particle distribution; a) unimodal distribution; b) multimodal distribution*

Sample impoverishment is frequently present in LDT systems, for example when an additional lane begins or in situations when a wrong mode was tracked and the real lane is actually in a different place in the state space. To handle these situations, it has been proposed to include initial particles which are introduced in every iteration [SoTa01, NSGD+04, DNMT08].

Unfortunately, analyzing the Particle filter in terms of multimodal likelihood functions reveals a structural problem, because the Bayesian estimator is a *maximum a posteriori* (MAP) estimator. This means that even when initial particles are introduced to every iteration, the original Particle filter will always converge to a unimodal distribution, which represents the strongest lane hypothesis. This behavior is explained in the following thought experiment and will be referred to as *multimodal sample impoverishment*.

Thought Experiment

- Given is a discrete state space with ten states denoted by n and
- a discrete multimodal likelihood function with likelihood value 4 in state three and 2 in state six as given in Figure...
- the particle set consists of 100 particles each with weight $\tilde{w}^{(n)} = 1/100$ and an equal distribution over the state space.

after iteration 1: Initially, there are 10 particles in each state, with $w^{(3)} = 4$, $w^{(6)} = 2$ and 80 particles in the remaining states with $w^{(n)} = 0$.

After resampling $2/3 \cdot 100 \approx 66$ equally weighted particles will be generated in state three, and $1/3 \cdot 100 \approx 33$ in state six.

after iteration 2: Now, there are 66 particles in state three with each $w^{(3)} = 4$ and 33 particles in state six with each $w^{(6)} = 2$. Sum of all weights will be $\sum w^{(n)} = 66 \cdot 4 + 33 \cdot 2 = 330$.

After resampling $66 \cdot 4/330 = 80$ equally weighted particles will be generated in state three, and $33 \cdot 2/330 = 20$ in state six.

It is evident that within a few iterations, mode two, which has lower likelihood than mode one, will vanish from the posterior distribution. Figure 4.10 illustrates the experiment again.

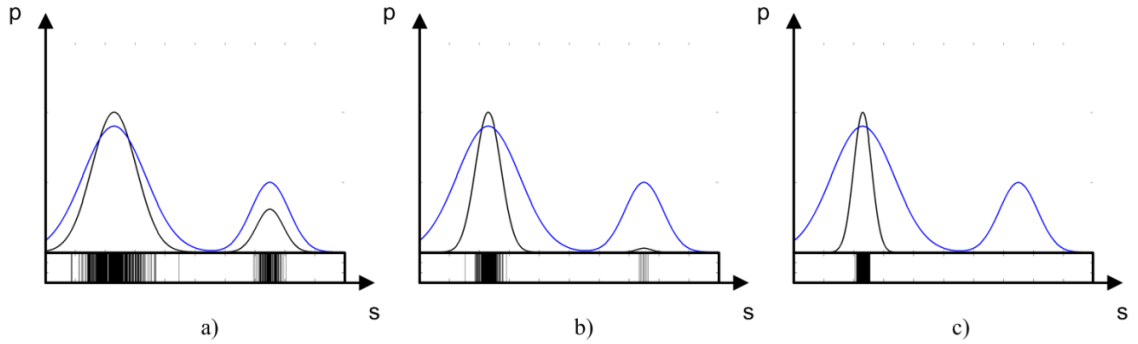

Figure 4.10: *sample impoverishment in one-dimensional state space where the black curve is the prior distribution $p(s_k|Y_{k-1})$ and the blue curve the observation likelihood $p(y_k|s_k)$, the bar underneath represents the particle distribution; a) after 2 iterations; b) after 6 iterations; c) after 12 iterations*

The thought experiment demonstrates why multimodal sample impoverishment occurs for unequal multimodal likelihood functions. The reason it is present even for equal likelihood functions is because usually the first hypotheses do not directly hit the best estimates, but converge slowly to that position in state space. Therefore it is very unlikely that an equal weight for all hypotheses of each mode will be maintained.

Multimodal sample impoverishment yields two major shortcomings regarding LDT systems. Firstly, any information about neighboring lanes is lost, and secondly, usually the lane with strongest lane features will be tracked.

The latter issue can be handled by introducing narrow physical constraints during the weight evaluation. This has been done by some authors [DNMT08], however it is merely a hack since the actual idea of having a comprehensive understanding of the scene is still lost. There are other approaches to deal with this problem, which are discussed in the following sections.

These methods have been evaluated by simulating a three lane structure with ideal lane markings. Figure 4.11 shows the lane set-up in world map space and its result with multimodal sample impoverishment.

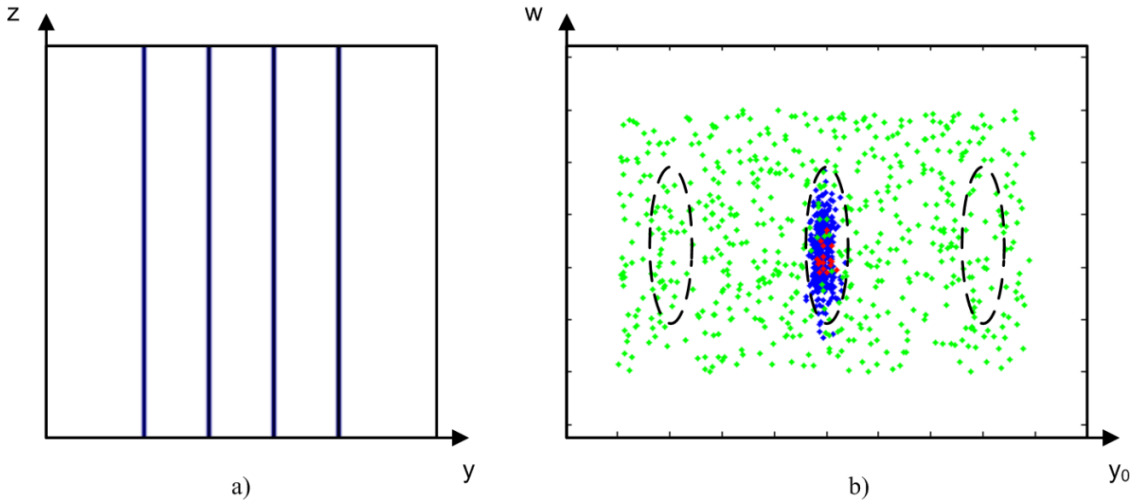

Figure 4.11: *sample impoverishment simulation: original algorithm; a) lane structure in world map space; b) state space subset (y_0 and w), green particles are initialization samples, blue particles are prior samples and red particles are posterior samples, dashed ellipses mark the area of expected local maxima*

4.6.1 Auxiliary Resampling

One attempt to overcome the problem of multimodal sample impoverishment is to reduce the weight of particles in highly populated regions of the state space. This can be done by introducing an auxiliary parameter to the weight normalization step so that the weights are regularized towards the average weight [PiSh99, RiAG04]. This is formally written as

$$\tilde{w}^{(n)} = \frac{(\alpha - 1)w^{(n)} + \overline{w}}{\alpha} \tag{4.50}$$

where $\overline{w}$ is the sample mean of all $w^{(n)}$ likelihoods. The auxiliary parameter α controls the regularization, that is if $\alpha \to \infty$ then the regularized likelihoods $\tilde{w}^{(n)}$ are equal to the original likelihoods $w^{(n)}$. If $\alpha = 1$ then all regularized likelihoods are equal to the mean likelihood $\overline{w}$.

However, this turns out to be only minimally effective in terms of recovering the multimodal distribution. Despite the fact that auxiliary resampling introduces high randomness for low α values, the particles only occasionally cluster around multiple modes depending on the initialization state. They always end up with a single mode distribution. Furthermore, with decreasing α towards 1, the more iterations are necessary until clustering can be observed.

4.6.2 Importance Sampling

A different approach is to introduce further information during sampling. That is, instead of using the transition density $p(s_k|s_{k-1})$ to sample from, one modifies the proposal distribution by adding the information gained from the stabilizer lines, which are indicators where additional lane candidates might occur.

Based on the lateral location and curvature of the stabilizer lines, importance samples are introduced to the proposal distribution. However, due to projective effects,

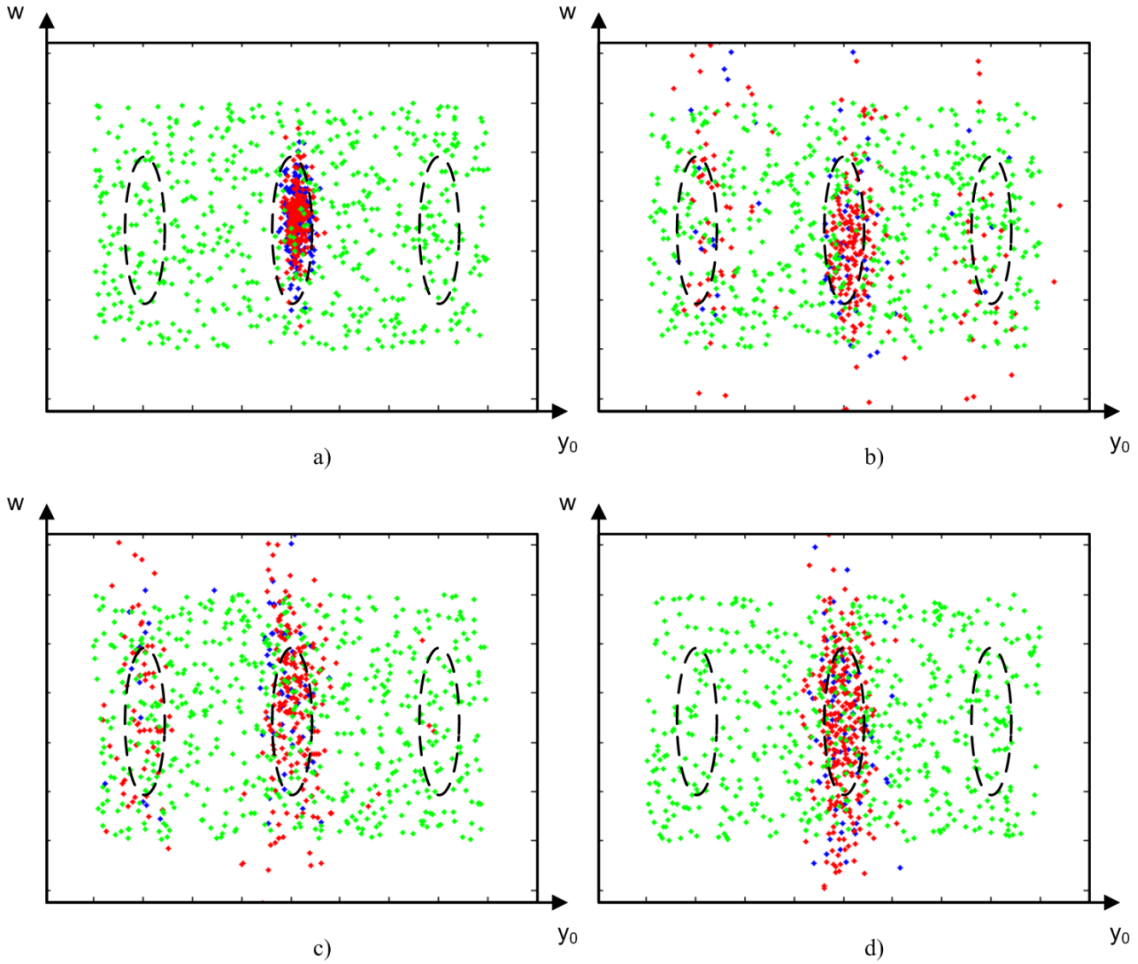

Figure 4.12: *sample impoverishment simulation: auxiliary resampling; a) after 15 iterations with* $\alpha = 1.5$*; b) after 30 iterations with* $\alpha = 1.1$*; c) after 50 iterations with* $\alpha = 1.1$*; d) after 100 iterations with* $\alpha = 1.05$

the importance samples have not matched the road features as well as the current hypothesis does. Hence the weights of the importance samples are too low to compete with the samples of the current hypothesis.

The next logical step is to combine importance sampling and auxiliary resampling. This was attempted, but did not yield any satisfactory results for the same reasons as given above.

4.6.3 Constrained Residual Resampling

If the situation is considered from a different point of view, it can be seen that the problem is that there are an excessive number of particles in one single mode. From here, the idea arises to artificially limit the number of *a posteriori* particles which are generated by a single *a prior* particle. An easy way of implementing this is to use residual resampling, where the number of generated particles is limited. This is referred to as *constrained residual resampling*. The residual particles are then introduced as initial particles. This approach is similar to the auxiliary resampling, but results in less randomness, because the initial particles are not part of the posterior distribution.

Simulations revealed the following behavior when varying the limiting value:

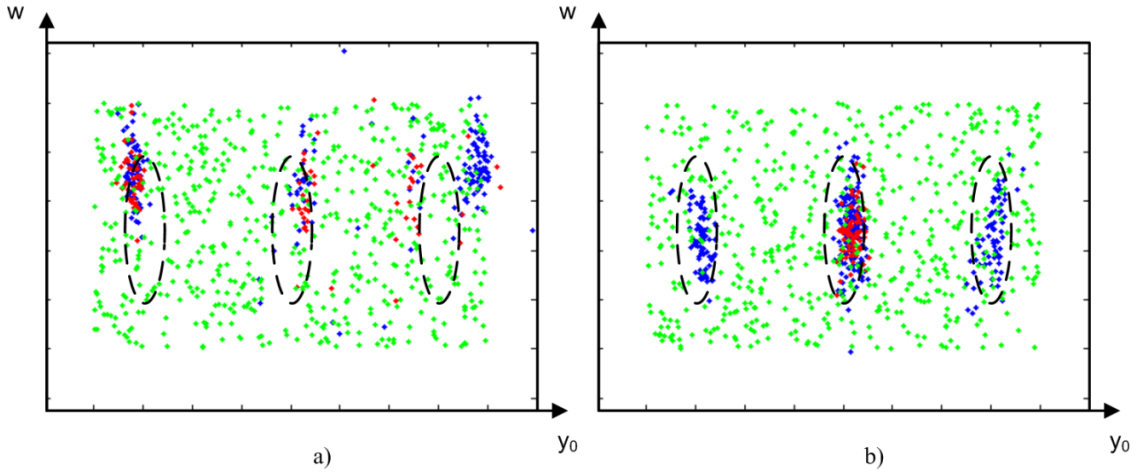

Figure 4.13: *sample impoverishment simulation: neighbor lane importance sampling; a) after 5 iterations; b) after 13 iterations*

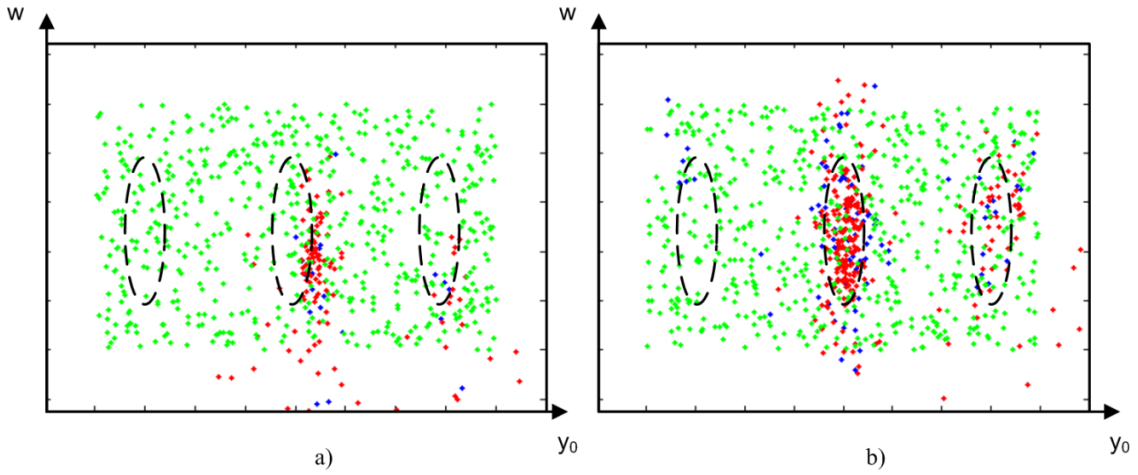

Figure 4.14: *sample impoverishment simulation: neighbor lane importance sampling and auxiliary resampling with $\alpha = 1.1$; a) after 20 iterations clustering is observed; b) after 40 iterations stabilizing estimate*

limit↑ limit-less behavior with unimodal distribution

limit↓ higher number of initial particles and still unimodal distribution

A change towards a slight multi modal distribution could be shown when varying the observation variance R:

R↑ stable behavior but unimodal distribution

R↓ unstable behavior but multimodal distribution

The reason for this is that the limitation is applied to a single *a prior* particle. However, considering that a mode is represented by not one, but several particles, and the fact that the state space is continuous, the overall number of particles of that mode is not limited at all. Therefore, one also needs to consider the particle density to be able to limit the particle population per mode. This insight leads to clustered resampling which is discussed in the next section.

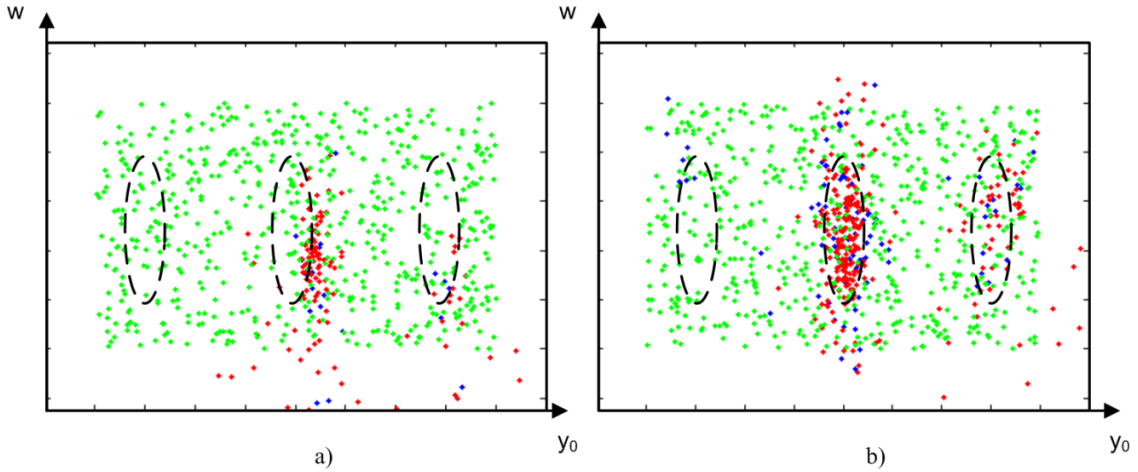

Figure 4.15: *sample impoverishment simulation: neighbor lane importance sampling and auxiliary resampling with $\alpha = 1.1$; a) after 20 iterations clustering is observed; b) after 40 iterations stabilizing estimate*

4.6.4 Clustered Resampling

Clustered resampling is a more computationally expensive approach, which enables present modes to be treated separately. This is achieved by limiting the number of particles per mode so that other modes can also arise.

The modes are determined by a simple clustering algorithm e.g. hierarchical clustering. Then the particle weights can be modified according to their cluster group.

4.6.5 Grid based Density Constrained Resampling

Another approach to limit the number of particles per mode is to introduce a limitation on the particle density. This could be done by lying a grid over the state space and constraining the resampling process to a maximal number of particles per grid element. Therefore, residual resampling seems to be a straightforward method of implementing the density constrained resampling.

Unfortunately, the last two proposed methods couldn't be implemented due to lack of time. However, they seem to be promising methods of achieving multimodal particle filter implementation, and would be investigating further.

5. Implementation

In this chapter, several implemented modules are discussed. Starting with the lane detection and tracking algorithm based on Particle filtering, the overall algorithm structure is shown and particular implementation issues are discussed.

Finally the embedded vehicle measurement system is explained. This system was implemented to record image data including measurements of the vehicles velocity and yaw rate.

5.1 Image Space, World Map Space and State Space

Before describing each algorithm module in detail, a brief note about the implemented data spaces is given in this section.

First the data structure of image space and world map space is stated, then the relation between world map space and world space is given. Finally, the relevance of this relation for implementation of the state space is discussed.

Both, data in image space and world map space are stored in Intel's `IplImage` data structure. The matrix indices are illustrated in Figure 5.1. It should be noted that the indices start with 0 and go up to $N-1$ in contrast to MATLAB structures where indices start with 1 and go up to N.

The relation between world coordinates and world map coordinates is then given by

$$\begin{bmatrix} y \\ z \end{bmatrix} = f\left(\begin{bmatrix} Y \\ Z \end{bmatrix}\right) = \left(\begin{bmatrix} Y \\ Z \end{bmatrix} - \begin{bmatrix} a_y \\ a_z \end{bmatrix}\right) \cdot \begin{bmatrix} b_y & 0 \\ 0 & b_z \end{bmatrix}, \tag{5.1}$$

where a_z and a_y are the offset of the world map coordinate system relative to the world coordinate system in meter, b_z and b_y are the world map resolution in meter/pixel. These parameters are fixed to specific values that showed good results for both accuracy and performance given in appendix ... .

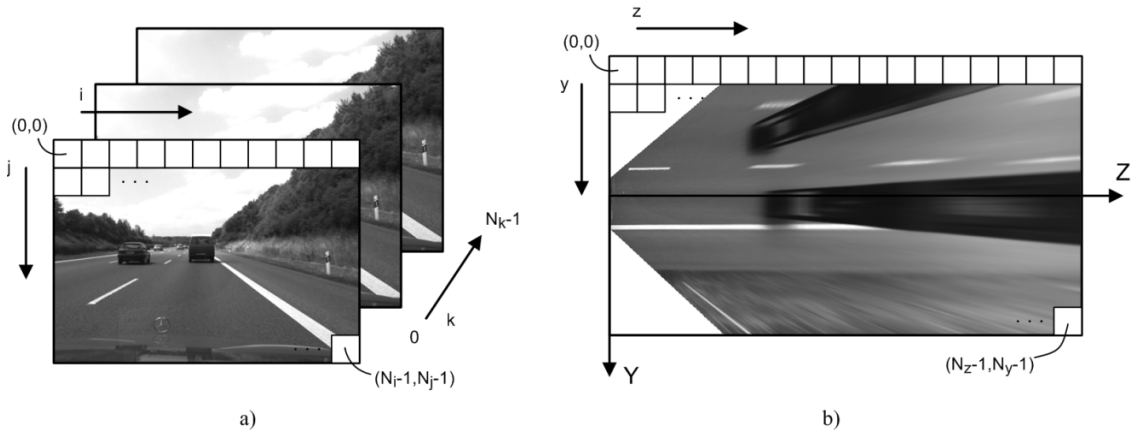

Figure 5.1: *a) image space, where k is the discrete time index, i and j are the indices denoting the pixels in image space. b) world map space, where z and y the pixels in world map space and Z and Y are the axes of the world coordinate system.*

The state space given in chapter 4 refers to the road model described in world space. Since the particle evaluation is computed in world map space, the road model must be projected in world map space, too. This projection is given by equation 5.1.

To speed up the particle evaluation module, this projection step is eliminated by defining the state space referring to the road model described in world map space instead of world space. Recall from chapter 4, the state vector describing the road model in world space is

$$\mathbf{s}_w = \left[\begin{array}{cccccccc} Y_0 & tan(\epsilon_w) & C_0 & C_1 & \frac{W}{2} & D_l & D_r & tan(\alpha_w) \end{array} \right]^T,$$

using the road model parameters for world map space, the implemented state vector is given by

$$\mathbf{s} = \left[\begin{array}{cccccccc} y_0 & tan(\epsilon) & c_0 & c_1 & \frac{w}{2} & d_l & d_r & tan(\alpha) \end{array} \right]^T. \tag{5.2}$$

5.2 Lane Detection and Tracking Algorithm

The LDT algorithm is implemented in C++, utilizing Intel's open source computer vision library OpenCV. The development suite for this project was Microsoft Visual Studio 2008 Professional.

The implementation is principally identical to the structure given in the design chapter 4. There are some deviations, though, to improve the system performance.

It shout be noted, that this implementation, as it is given, does not use any parallel computation methods. However, the Particle filter is very well suited to be implemented on highly parallel structures. These parts are marked in Figure 5.2 as 3D structures.

Initialization

Regarding performance reasons, all data which is used in calculation expensive algorithm parts, such as particle evolution and particle evaluation, is pre-allocated and passed to the processing function by reference.

Therefore different data structures need to be allocated and initialized before the LDT algorithm can be started.

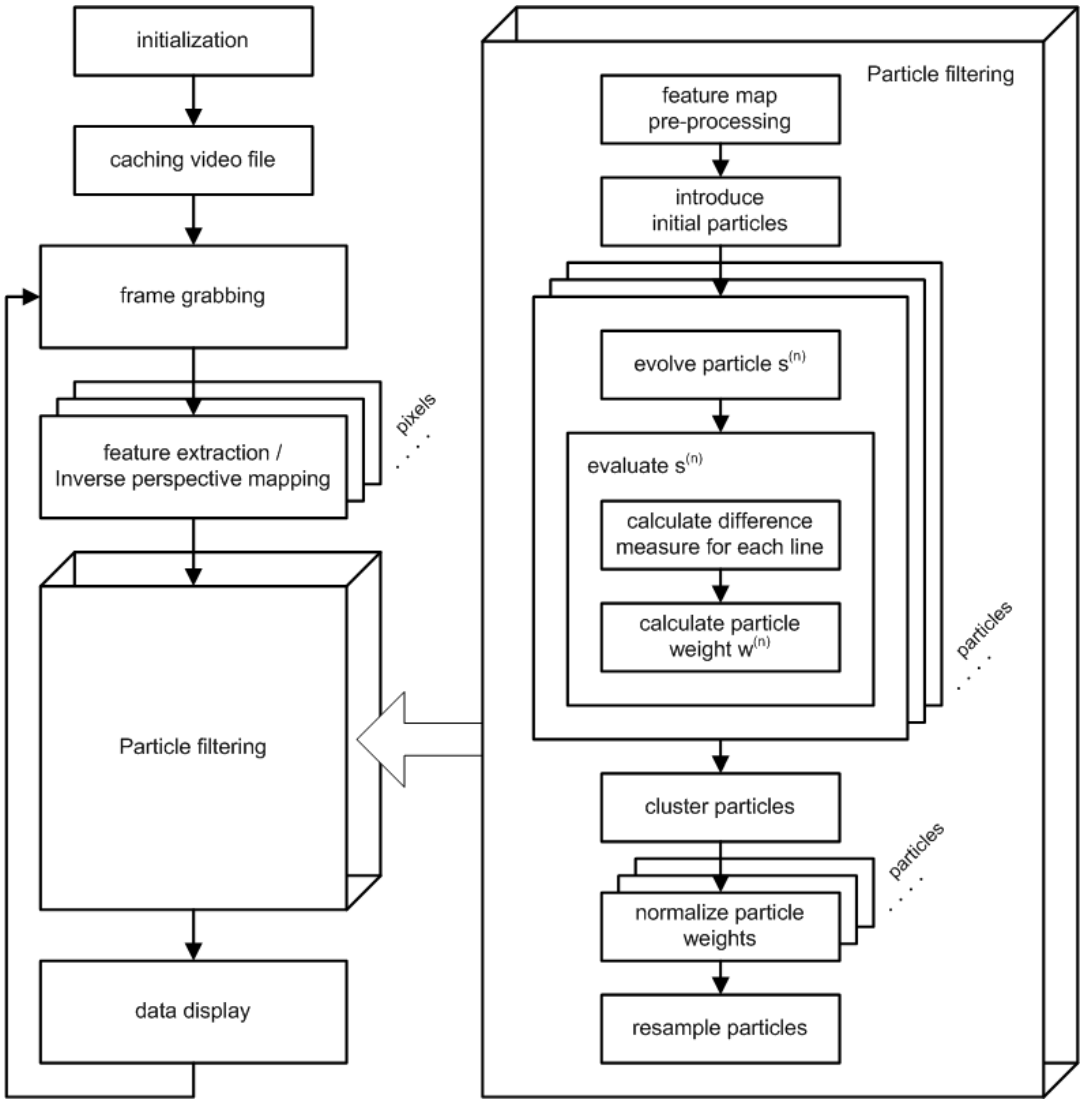

Figure 5.2: *flow chart of lane detection and tracking algorithm*

Caching Video File

The automotive videos are given in the form of single images in the PGM-file format. The file header contains time discrete measurements of the vehicle's velocity and yaw rate. To get a higher performance, the video sequence is pre-allocated in the memory.

The file header of the videos given by the German automotive company Daimler AG is slightly different from the file header of the videos recorded by the author and hence specific loading functions are provided.

Listing 5.1: *Daimler AG header information*

```
P5
\# :speed dd.dddddd :steeringAngle dd.dddddd :cycleTime dd.dddddd
\#
640 480
255
```

Listing 5.2: *UWA header information*

```
P5
\#bigEndian
\#[Units are rads, meters and seconds]
\#dt= dd.dddddd
\#Sp= dd.dddddd
\#Ax= dd.dddddd
\#Ay= dd.dddddd
\#Az= dd.dddddd
\#Pi= dd.dddddd
\#Ro= dd.dddddd
\#Yw= dd.dddddd
800 600
255
```

Frame Grabbing

During frame grabbing, first an image pointer is assigned to the particular image in the source image array at time index k. This serves only to preserve overview of the code. Second a copy of the source image is stored in a three dimensional image structure so that extracted features, lane estimate and other information can be displayed in color.

Feature Extraction

Intel's open source computer vision library OpenCV provides a various amount of different feature extraction methods so that the Sobel operator does not have to be implemented from scratch. The function `cvSobel(·)` provides an intensity gradient image in either i- or j-direction for different kernel sizes.

The gradient magnitude image is easily computed given the equation in section 4.3.2. However, the gradient direction must be encoded in a way that the slope of a line hypothesis given in image space can be compared to it.

The slope of a line in image space is defined as $m = \Delta i / -\Delta j$, this gives $m = 0$ for a vertical line and $m = \pm\infty$ for a horizontal line approached from either positive or negative slope respectively. The negative sign results from the fact that in image space the origin is in the left upper corner and therefore the j-axis is in the opposite position from that of common coordinate systems. The line's angle is then given by

$$\theta(m) = atan\left(\frac{\Delta i}{-\Delta j}\right), \tag{5.3}$$

where return values of `atan(·)` is in the range $[-\pi/2, \pi/2]$ so that a continuous transition from horizontal line with $m = -\infty$ via vertical line to horizontal line with $m = \infty$ is assured.

The gradient direction must be encoded based on the same definition of line slope given above. However, one slope has two different gradient direction values. One for a dark-bright transition and one for a bright-dark transition. If the gradient direction is encoded as

$$\Theta(j, i) = [atan2\left(\mathbf{y}_i, \mathbf{y}_j\right) + \pi] \cdot \frac{255}{2\pi}, \tag{5.4}$$

where $\mathbf{y}_i$ is the gradient image in i-direction and $\mathbf{y}_j$ in j-direction and `atan2(y,x)` is a standard c library function that computes the arc tangent of y/x, using the signs of the arguments to compute the quadrant of the return value with return value range $[-\pi, \pi]$, then the line's angle for dark-bright transition is given by

$$\theta_{\nearrow}(m) = \left[atan\left(\frac{\Delta i}{-\Delta j}\right) + \frac{\pi}{2} \right] \cdot \frac{255}{2\pi} \tag{5.5}$$

and the angle for bright-dark transition by

$$\theta_{\searrow}(m) = \left[atan\left(\frac{\Delta i}{-\Delta j}\right) + \frac{3\pi}{2} \right] \cdot \frac{255}{2\pi}. \tag{5.6}$$

These expectation values are needed during the particle evaluation to compare the expected gradient direction value $\theta(m)$ with the actual gradient direction value of the feature map $\Theta(j, i)$.

Inverse Perspective Mapping

OpenCV provides the inverse perspective mapping function `cvWarpPerspective(·)` which computes the perspective mapping using one of the following pixel interpolation methods; nearest neighbor, bilinear, bicubic and pixel area relation, based on a given homography matrix. The coordinate transform is given by[1]

$$\begin{bmatrix} z \\ y \\ \psi \end{bmatrix} = \mathbf{H}^{-1} \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \tag{5.7}$$

which is

$$\texttt{wm}\,[z, y] = \texttt{img}\left[ip\left(\frac{i\mathbf{H}_{11}^{-1} + j\mathbf{H}_{12}^{-1} + \mathbf{H}_{13}^{-1}}{i\mathbf{H}_{31}^{-1} + j\mathbf{H}_{32}^{-1} + \mathbf{H}_{33}^{-1}}\right), ip\left(\frac{i\mathbf{H}_{21}^{-1} + j\mathbf{H}_{22}^{-1} + \mathbf{H}_{23}^{-1}}{i\mathbf{H}_{31}^{-1} + j\mathbf{H}_{32}^{-1} + \mathbf{H}_{33}^{-1}}\right)\right], \tag{5.8}$$

where `wm` is the destination image in world map space, `img` the source image in image space and $ip(\cdot)$ the specified interpolation method[2].

It should be explicitly noted that $\mathbf{H}$ is the homography between the image plane and the world map. It is significantly distinguished from $\mathbf{H}_w$ which is the homography between the image plane and the ground plane given in world coordinates.

Introduce Initial Particles

During the deduction of the Bayesian estimator, it is assumed that the initial density function $p(\mathbf{s}_0)$ is known, see equation 3.13. Since there is no prior knowledge about the position and shape of the lane, this density function is set to a uniform distributed probability over the whole state space. This is implemented by introducing initial particles which are randomly spread through the initial range of the state space.

Furthermore, to avoid sample impoverishment when tracking wrong road features, initial particles are used to continuously search the state space for more significant lane hypothesis. Therefore, 10 percent of the particle set are initial particles introduced at each iteration.

[1] The coordinate system used in MATLAB is different to the one used in OpenCV, in MATLAB this relation is given by $\begin{bmatrix} y & z & \psi \end{bmatrix}^T = \mathbf{H}^{-1} \begin{bmatrix} j & i & 1 \end{bmatrix}^T$, therefore the homography matrix needed to be recalculated, this is given in appendix ... .

[2] Further information about interpolation methods used by `cvWarpPerspective(·)` is given in the OpenCV reference at http://opencv.willowgarage.com/documentation/c/index.html.

Evolve Particles

The particle evolution as well as the particle evaluation is processed directly one after another for each particle in a for-loop. This structure enables to implement both modules in a highly parallel structure. This is denoted by the 3D structure in figure 5.2 and is shown in detail in figure 5.3.

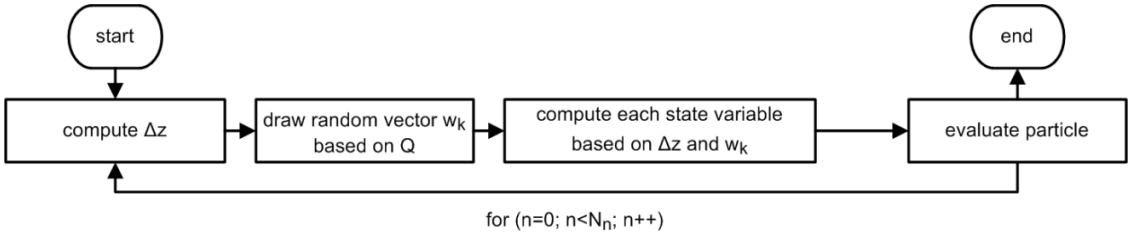

Figure 5.3: *flow chart of particle evolution and evaluation in one for-loop*

Since the state space is implemented referring to world map space, the evolution process given in section 4.2 must be adapted accordingly. Recall from chapter 4, the road model is basically[3] given by

$$Y_c(z) = Y_0 + tan(\epsilon_w)Z + \frac{C_0}{2}Z^2 + \frac{C_1}{6}Z^3,$$

and the position change in Z-direction within two subsequent states is given by

$$\Delta Z_k = v_k \cdot \Delta t \cdot cos\left(\epsilon_{w,k} + \frac{\Delta\epsilon_{w,k}}{2}\right).$$

With the given road model for world map space

$$y_c(z) = y_0 + tan(\epsilon)z + \frac{c_0}{2}z^2 + \frac{c_1}{6}z^3. \tag{5.9}$$

the position change in z-direction is then given by

$$\Delta z_k = (\Delta Z_k - a_z) \cdot b_z.$$

Evaluate Particles

The particle evaluation is implemented according to the methods described in section 4.5. An overview of the evaluation algorithm is given in Figure 5.4 where the average likelihood measure is shown.

First the lane hypothesis is projected according to equation 5.9. Then the line individual difference measure is computed by summing up over z, subtracting the mean either during the summation or after normalizing and normalizing the sum. Finally computing the particle weight according to equations 4.40 and 4.39.

[3]For Y_l, Y_r, Y_{sl} and Y_{sr} holds the same.

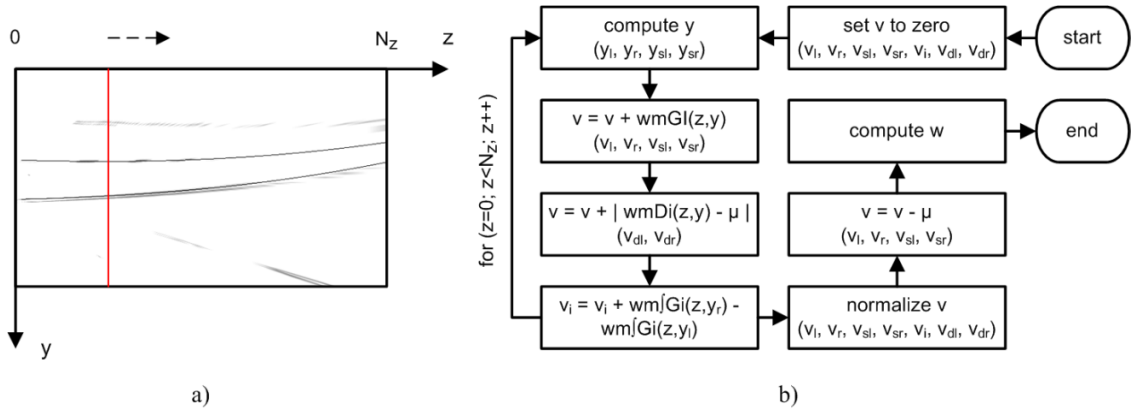

Figure 5.4: *a) gradient magnitude world map with projected lane hypothesis; b) flow chart of evaluation algorithm*

Normalize Weights and Resample Particles

After normalizing the particle weights the particles are resampled with the stratified resampling method described in section 3.3.2.

Compute Lane Estimate and Display Data

Finally the lane estimate is computed using a simplified clustering method. Here the particle with maximum weight is used as center point, all particles with a certain distance to the center or smaller are included in computing a weighted mean, others are left out. Finally the lane estimate and other relevant information is drawn into the output image.

5.2.1 Naming Convention

Since the `IplImage` structure provides different bit-depths, all image structures follow one naming convention which encodes the bit-depth and the image type, this is for structures in `main(·)`:

`typeDescr_BIT`

Since the data is passed to functions by reference, the arguments serve as input, output, both or temporary data used in the function, therefore the naming convention is extended for function arguments to:

`typeIntfDescr_BIT`

Table 5.1: *wildcard definition: variable name*

wildcard	definition	description
`type`	`img`	data in image space
	`wm`	data in world map space
	`tm`	data in travel map space
	`ss`	data in state space
	`mat`	matrix data
`Intf`	`In`	variable is data input
	`Out`	variable is data output
	`IO`	variable is data input and output
`Descr`		short description what the variable is used for
`BIT`	`08U`	unsigned char
	`16S`	signed short
	`32S`	signed int
	`32F`	float
	`64F`	double

Since for some functions a huge number of variables are passed by reference, the order in which the different variables appear is also defined, that is

`ldVerbObject([input],[input/output],[output],[parameters],[temp])`

Table 5.2: *wildcard definition: function name*

wildcard	description
`Verb`	describes what the function is doing
`Object`	is the Object which the function is applied at
`[input]`	list of data input arguments
`[input/output]`	list of data input and output arguments
`[output]`	list of data output arguments
`[parameters]`	list of function parameters
`[temp]`	list of temporary data arguments

5.3 Car Measurement Unit

To record automotive scenes in the area of Perth, a measurement unit for the University of Western Australia's renewable energy vehicle (REV) was designed. The REV which is used is a modified *Getz* from the car manufacturer Hyundai to prove the concept of zero emission vehicles for the future[4].

The measurement unit needs to acquire images of the road scene and the vehicle's motion parameters (velocity and yaw rate). The images are captured by the high resolution web cam *QuickCam Pro 9000* from Logitech, the yaw rate is measured by the low cost intrinsic measurement unit *Atomic IMU 6DoF* from Sparkfun and the velocity was measured directly from the speed sensor signal of the car electronics utilizing the micro controller evaluation kit *AVR Butterfly* from Atmel.

The measurements are pre-processed and combined using a MATLAB script. The principal set-up is illustrated in Figure 5.5. Each single image frame is stored in a PGM-file, where the motion parameters are stored in the PGM-header. A description of the image header is given in appendix ... .

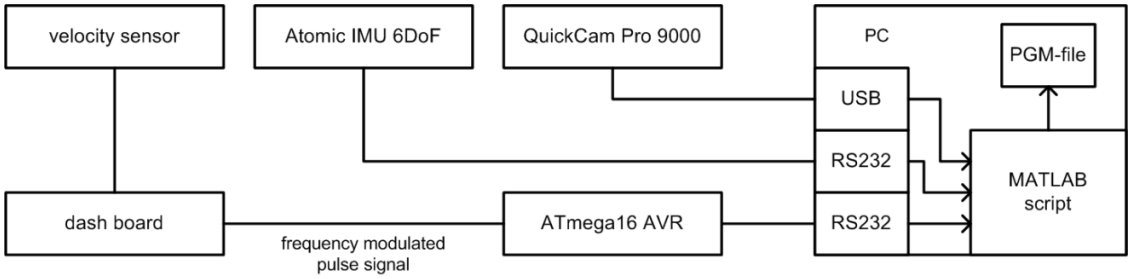

Figure 5.5: *principal set-up of car measurement unit*

Frequency Consideration

The IMU sends the data with a baud rate of 115200 bit/second via serial connection using the following data protocol.

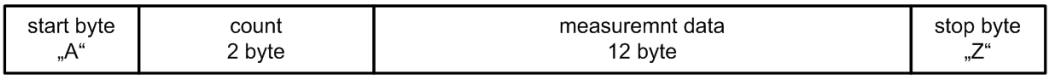

Figure 5.6: *data protocol of IMU serial connection*

Given the number of bits used for one byte defined by the RS232 protocol and the number of bytes sent by the IMU, the maximum frequency is given by

$$f_{max} = \frac{115200 \text{ bit/s}}{(\text{start bit} + 8 \text{ data bit} + \text{stop bit}) \cdot (16 \text{ byte})} = \frac{115200}{10 \cdot 16 \text{ s}} = 720 \text{ Hz}. \quad (5.10)$$

The speed signal is a frequency modulated pulse signal. The relation between frequency and velocity is linear and was determined empirically to be

$$v = 1.413 \cdot f_v. \quad (5.11)$$

[4]Further information on the university's efforts on renewable energy vehicles are given at http://robotics.ee.uwa.edu.au/rev/index.html.

The maximum frequency that needs to be measured is therefore given by the maximum speed (defined to be 120 km/h) and is 85 Hz.

The micro controller unit serves as buffered interface which measures the frequency varying signal and outputs the velocity with a constant frequency of 200 Hz via serial connection. It should be noted that always the latest measurement is sent.

The bottle neck of the measurement unit is given by the image acquisition, therefore all other measurements are buffered in an array. The final motion parameters are computed by the mean of all measurements acquired during one cycle. Figure 5.7 shows the flow chart of the measurement system.

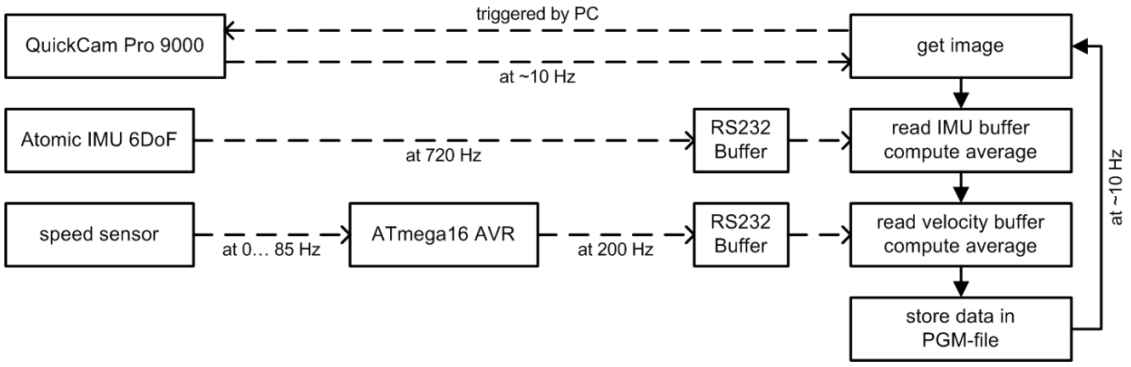

Figure 5.7: *flow chart of MATLAB script for capturing and storing image data and motion parameters*

Finally a dynamic frame rate is given within the range 9 to 11 fps, the actual cycle time is also stored in the image header.

Frequency Measurement

To measure the pulse frequency, a 16-bit timer and an external input interrupt of the ATmega16 are used. The timer is used to measure the time between two edges (half the period time), where the external interrupt is used to detect a rising or falling edge.

Each time a rising or falling edge is detected the timer value is used to compute the frequency, and then set to zero again to measure the next period.

6. Performance and Accuracy Evaluation

In this chapter, the proposed LDT system and its various implementations are analyzed and evaluated in terms of accuracy and performance.

In the first section, the applied error measurement methods are defined. This is followed by a discussion of the most important algorithm issues as well as solutions for overcoming them. In section three and four a comparison of different interpolation methods of the IPM and of two different tracking methods is given. Finally, the performance of the overall system is analysed.

In this context, the term *performance* refers to a measure of the execution time of the algorithm or its modules.

The accuracy evaluation is based on so called *ground truth data*. This data holds the information about position and shape of the actual lane which is given per frame. The system output can then be evaluated in terms of accuracy based on the given ground truth data.

Unfortunately, the ground truth data was not available, so the data needed to be generated manually. This was achieved by tuning the algorithm towards the specific road scene sequence, increasing the number of particles to 3000, and running the sequence 50 times in a loop and computing the average value of each parameter per frame. The result gives an estimate of the ground truth, which was validated manually and shown to match the lane very well.

All automotive sequences used in this chapter were provided by the German car manufacturer Daimler AG and are recorded on German Autobahn in the area of Stuttgart in Germany.

6.1 Measuring Methods

In this section a brief explanation of the used measuring methods for evaluating performance and accuracy is given.

6.1.1 Performance Measuring Method

The execution is measured using the `timeGetTime()` function, provided by Microsoft Windows' Winmm-library, which returns the system run time in milliseconds (accuracy is ca. 15 ms). The measured function is looped 1000 or 10000 times to achieve a higher accuracy.

All measurements are taken on a Thinkpad R61 running MS Windows XP Professional SP3. The residing CPU is an Intel Core 2 Duo, T8300 at 2.4 GHz which has 3GB of RAM to access.

6.1.2 Accuracy Measuring Method

The computed error between ground truth data and estimated lane can be measured in several different ways. A straightforward method would be to compute the distance of the gound truth state vector and the estimate state vector element-wise. The distance vector is given by

$$f_{k,ss} = |s_{k,gt} - s_k|, \tag{6.1}$$

where $s_{k,gt}$ is the state vector given by the ground truth data. This gives reasonably good results for lane departure warning systems, where only the lateral position and the lane width is necessary.

However, in terms of the accuracy of matching the lane estimate to the features over distance, this error measure would give poor information.

A second approach is to project both, ground truth and lane estimate in world map space and compute the average distance of the lateral displacement of every line element. This measure is given by

$$f_{k,wm} = \frac{1}{N} \sum_{z}^{N} |h_s(s_{k,gt}) - h_s(s_k)|, \tag{6.2}$$

where N is the measure distance and $h_s(\cdot)$ is the projection function $F(z, y) \longleftarrow s_k$ as described in section 4.3. This method gives a reasonable error measure when considering an overall match of the lane estimate.

To measure the accuracy against other LDT systems which do not provide any parameters, but just the lane in image space, another error measuring method was implemented. Here, the ground truth and the lane estimate is projected in image space, then the average of the absolute lateral difference of pixels is computed. This is given by

$$f_{k,im} = \frac{1}{N} \sum_{z}^{N} |h_c^{-1}(s_{k,gt}) - h_c^{-1}(s_k)|, \tag{6.3}$$

where N is the number of pixels that are evaluated and h_c^{-1} is the inverse imaging function described in section 4.3.

Which of these error measures are applied in the following sections will be indicated by the measurement set data.

6.2 Design Issues

During the design phase, several problems arose which influenced the design of the algorithm. The following issues influenced the design the most.

6.2.1 Comparison of different Road Model Orders

The order of the road model has a great influence on the systems accuracy. However, the choice of the road model always depends on the application the lane estimate is intended for. In this section polynomial road models with 1^{st}, 2^{nd} and 3^{rd} order are compared.

Figure 6.1 shows the world space distance measure $f_{k,wm}$ for all three model orders. The high values in the first frames are caused by the initialization phase and can be neglected. As can be seen, the deviation of the linear model is rather high (up to 1 meter). The 2^{nd} and 3^{rd} order model are almost the same, which is explained by the presence of only parabolic shapes in this specific automotive sequence.

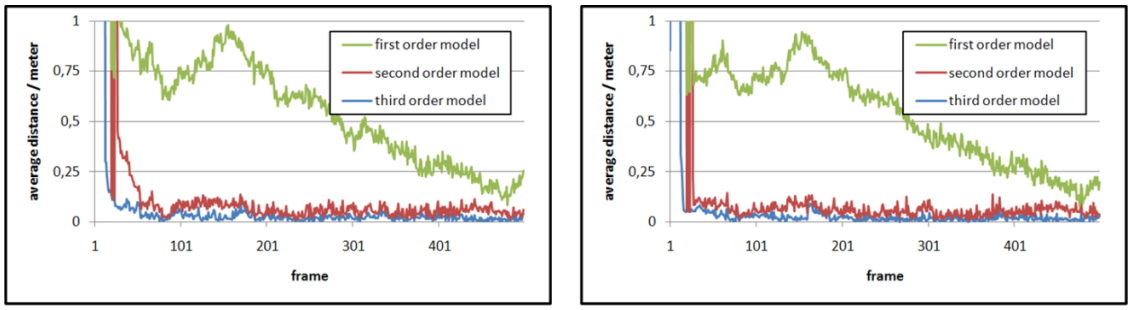

Figure 6.1: *average line distance measured in world space for 1^{st}, 2^{nd} and 3^{rd} road model order; a) left line; b) right line*

Figure 6.2 shows the state space distance vector $f_{k,ss}$ (position and width) for all three model orders. There are only minor differences, which indicates that the linear model could be used for lane departure warning systems without any loss of accuracy. However, higher order models are necessary for other applications - for example, it has been shown by [McTr06] that a lane change assistance system needs a road model of third order to compute the intersection of the vehicles trajectory and the lane.

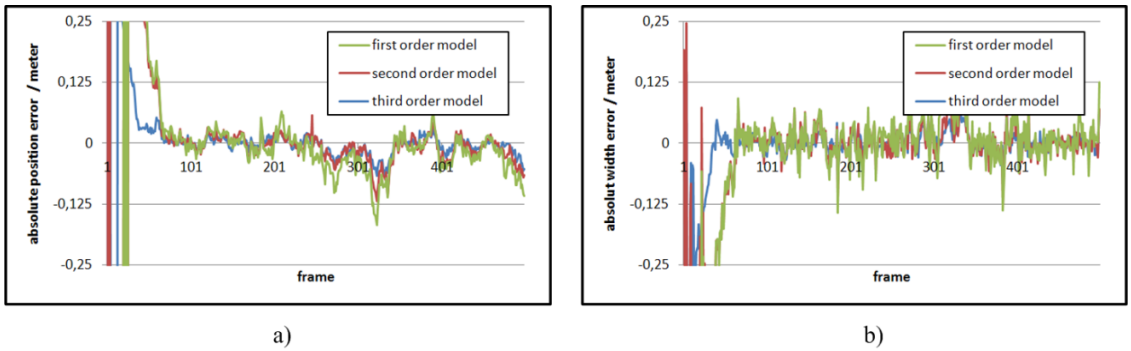

Figure 6.2: *state space distance vector for dimension: position and width for 1^{st}, 2^{nd} and 3^{rd} road model order; a) lateral position; b) width*

This analysis highlights two Aspects of algorithm design. Firstly, the better the road model matches the scene, the better are the results. This means that the choice must take the target application into account. Second, this algorithm can easily be adapted to use lower order models as well. The following frame series shows selected frames, which enables the data given in the diagram to be visualized.

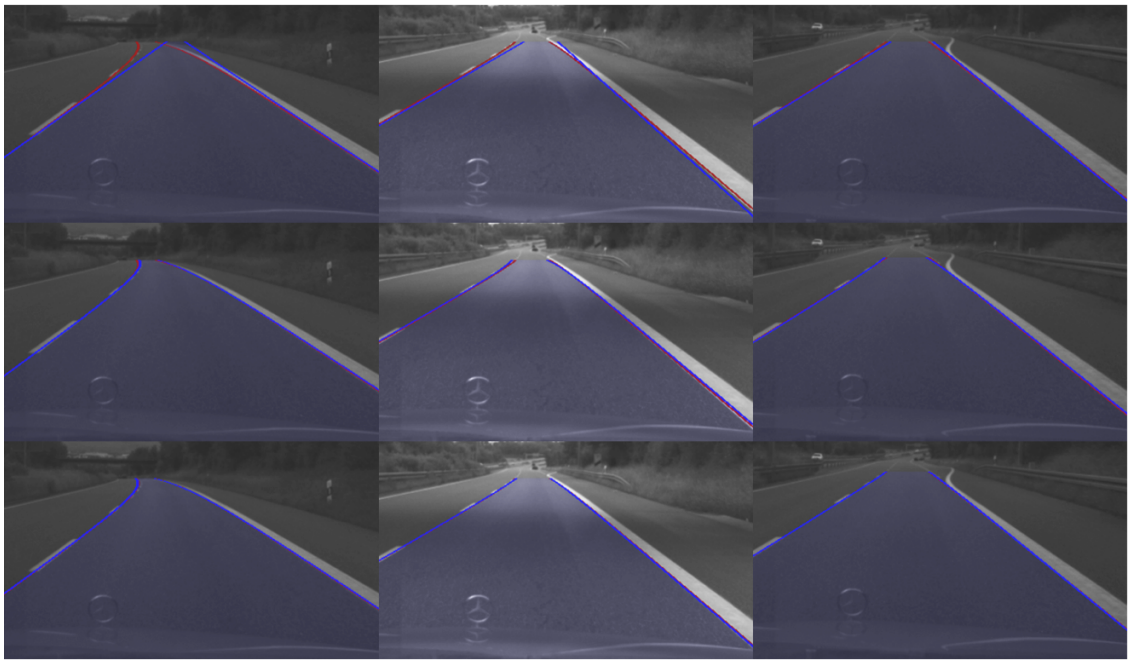

Figure 6.3: *frame series for different road model orders; first row: model of 1^{st} order; second row: model of 2^{nd} order; third row: model of 3^{rd} order; frame numbers from left to right are: 153, 384, 448*

6.2.2 Influence of Pitch Angle Variations

Due to changes in the slope of the road or pitch movement of the car, the angle between the image plane and the assumed ground plane changes continuously. The following analysis shows that taking the pitch angle variation into account is very important. The automotive sequence used here includes a large change in the pitch angle.

Figure 6.4 shows the average distance of left and right lane boundary estimated the algorithm without compensating the pitch angle change. The actual change of the pitch angle is blended over, however, this curve is unit-less and serves just as a reference for the dynamic vehicle behavior.

In the first half of the sequence, the difference is relatively large, and as a result of this, only one line provides a good estimate, the other is ”hanging in the air”. There are quite a few singularities in the curve diagram, these result from jumps from one lane to the neighboring lane. It can also be seen that with declining pitch angle, the system provides more stable and more accurate results. A series of frames is given in Figure 6.5.

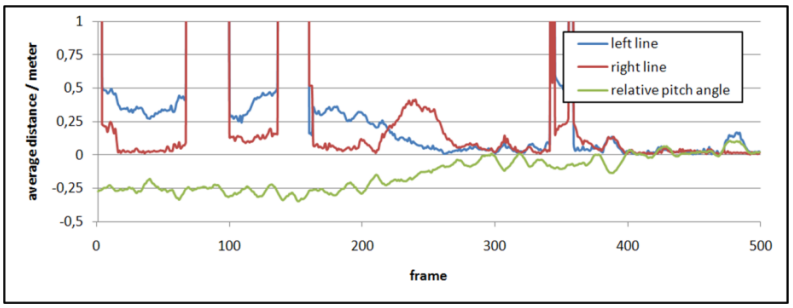

Figure 6.4: *average line distance measured in world space for algorithm without taking pitch angle change into account; blended over unit-less pitch angle*

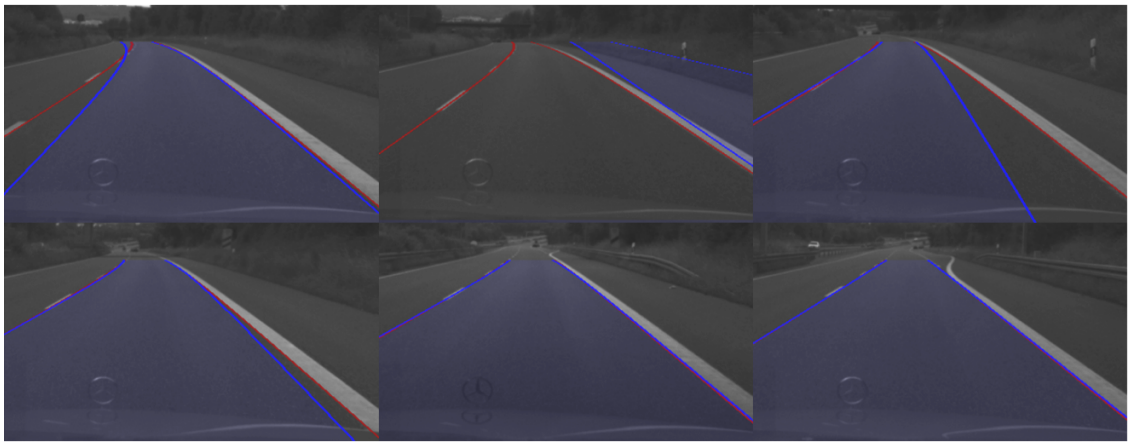

Figure 6.5: *frame series for algorithm without taking pitch angle intp account; frame numbers from left to right and top to bottom are: 50, 150, 262, 300, 400, 450*

6.3 Comparison of IPM Interpolation Methods

Intel's open source library OpenCV offers an inverse perspective mapping (IPM) function which can apply three different interpolation methods, they are: nearest-neighbor interpolation, bilinear interpolation and bicubic interpolation.

Interpolation is needed when the intensity of a pixel in the source image is to be mapped to a pixel in the destination image. The reason for this is that the destination coordinates, computed by the specified transform and source coordinates, do not necessarily match the destination image raster. *Mapping* is a very general terminology and can refer to resizing, affine transform or perspective transform.

In this section the interpolation quality and performance will be analyzed. To analyze the performance, only the image size of the destination image is relevant (due to indirect transform, the destination image will be the space where the coordinates are swept through). Three different destination image sizes are used to perform the evaluation: size 1, 361x400; size 2, 361x600; and size 3, 361x1000. These values have been chosen because 1 and 3 represent the length limit at the minimum of 20 m and at maximum of 50 m, size 2 is the length actually used which is 30 m.

There are two options to combine feature extraction and the IPM algorithm as described in section 4.4. In the first of these, the image is gradient filtered and then transformed to world map space. In the second option, the image is transformed to world map space, then the gradient filter is applied. If the direction information is maintained, this second method is preferable because the direction expectation

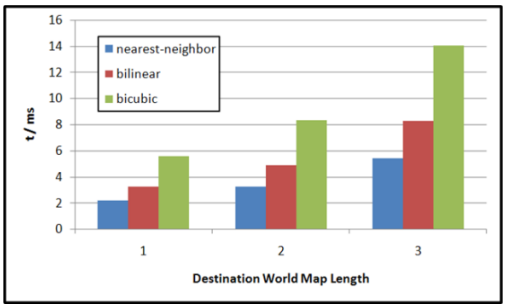

Figure 6.6: *execution time of IPM using three different interpolation methods for three world map lengths; size 1, 20 m; size 2, 30 m; size 3, 50 m*

value is faster to compute for direction information gained in world map space, see section 4.5.3. Three example images are shown in Figure 6.7.

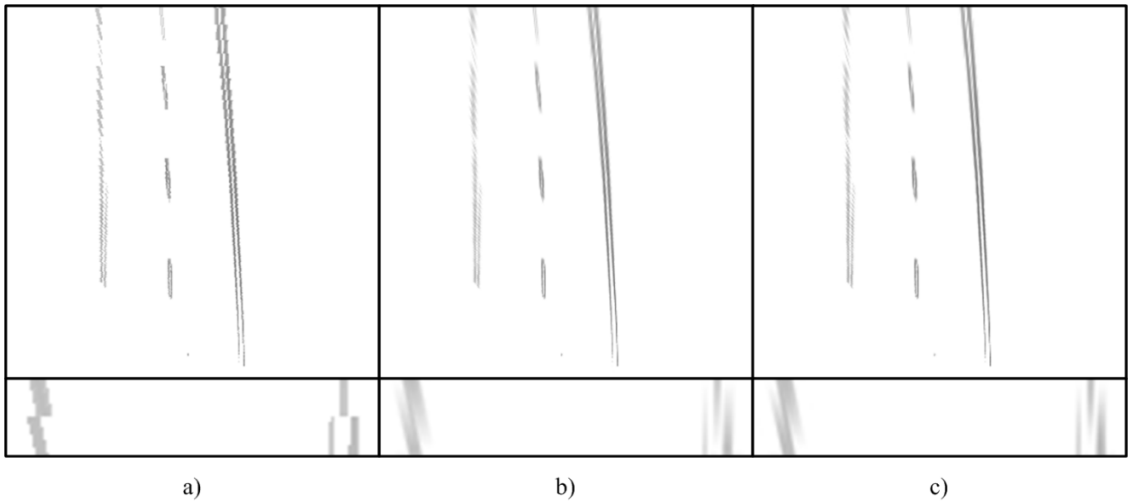

Figure 6.7: *IPM with different interpolation methods (lower image is rescaled by the factor 3); a) nearest-neighbor interpolation; b) bilinear interpolation; c) bicubic interpolation*

The nearest-neighbor interpolation is the fastest but also least accurate method. In comparison, the bilinear interpolation yields a smoother transition and better accuracy. The bicubic interpolation shows only a little improvement compared to bilinear interpolation, and considering the higher computational effort appears to be less attractive. Therefore, the bilinear interpolation appears to be the best compromise between good accuracy and fast execution time.

Considering the results shown in Figure 6.8, it appears that the remapping process introduces edge artifacts. The direction is determinable only with a large error. This means that the Sobel filter is not applicable in world space, however a filter with larger kernel size would probably work.

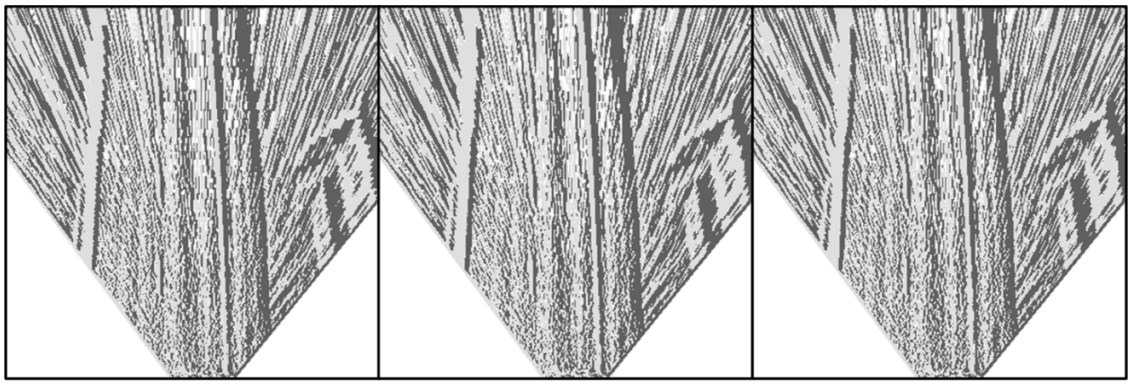

Figure 6.8: *IPM with different interpolation methods and gradient filtering in world map space; a) nearest-neighbor interpolation; b) bilinear interpolation; c) bicubic interpolation*

6.4 Comparison of Tracking Methods

The quality of particle filters is determined by the state evolution model, which links together the information obtained by subsequent frames and filters the state estimate in time-domain. This is called *tracking*.

Particle filter tracking is a non-trivial method which involves measuring motion parameters. Another commonly used, more simple approach is to have an initial guess of the lane's position and then search for the lane around this position [STHS08, SaST06, LSBL+08]. The initial guess is given by the lane estimate of the previous frame.

In Figure 6.9, a comparison of the average distance using both tracking methods is shown for left and right lane boundary. The distance data up to frame 100 can be neglected, since this is caused by the initialization phase. Until frame 450 both tracking methods show good results. This is because the vehicle bearing has relatively small values. Therefore, the lane position of the subsequent frame stays almost the same. However, from frame 450 on, when the bearing angle changes because of a lane changing maneuver, the simple tracking method lacks estimation accuracy, as shown in the diagrams (the error of the left line is larger then that of the right line). The bearing angle is blended over, however, this curve is unit-less and serves just as a reference for the dynamic vehicle behavior.

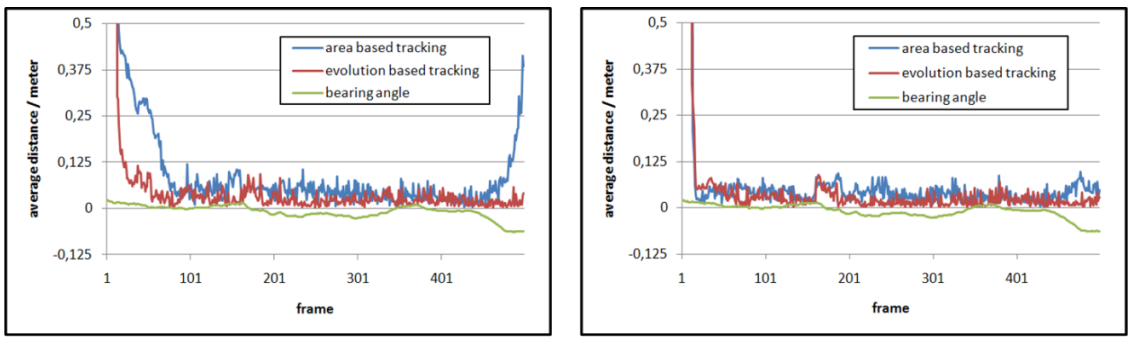

Figure 6.9: *blended over unit-less bearing angle*

Figure 6.10 shows two frames of the algorithm using the area-based tracking method with low accuracy. For comparison, one frame of the algorithm using the evolution-based method is also given. It should be noted that the simple tracking method tracks very satisfactory until frame 450, so there is no need to illustrate the algorithm accuracy for the earlier frames.

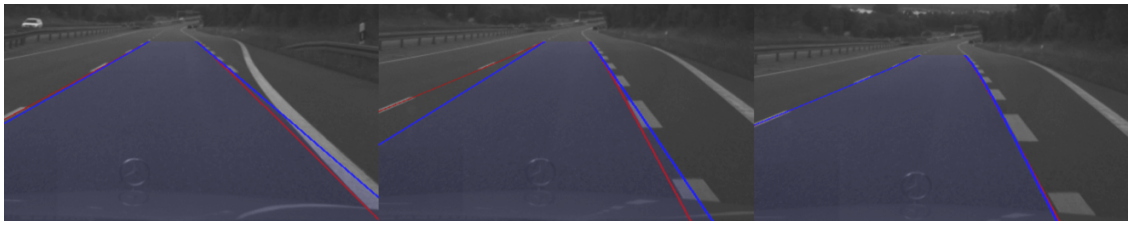

Figure 6.10: *two frames of algorithm with area based tracking (left, frame 484; middle, frame 498) and one frame of the algorithm with evolution based tracking (right, frame 498)*

Figure 6.11 illustrates both tracking methods. The drawn lane represents the estimate based on tracking only. This means the system runs until frame 465, then the update process is stopped and the lane estimate is computed based on the evolution model, or kept the same (area-based tracking) respectively, until frame 475. It can be seen that the area-based method does not follow the lane to the left, whereas the evolution-based method does follow the lane, however with a significant difference at the right line. This is probably caused by measurement errors of the motion parameters. The algorithm deals with these errors by introducing a Gaussian distribution when evolving the particles.

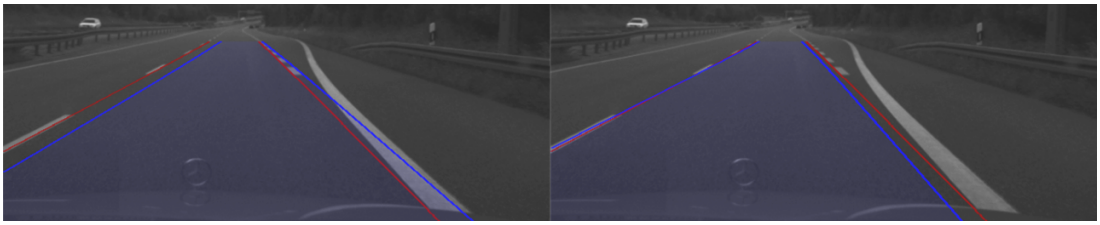

Figure 6.11: *illustration of tracking methods; left, area based tracking, frame 475; right, evolution based tracking, frame 475*

6.5 Overall System Performance

In this section, the overall system performance in terms of accuracy and execution time is presented.

6.5.1 Algorithm Accuracy

Since no ground truth data has been available, and the manually generated data was only available only for the first sequence, the algorithm accuracy is judged by the author's impression, where the percentage value refers to the sequence performance without including the initialization time.

Four frame series, one for each automotive sequence, are given and discussed. This is then followed by an analysis of the overall execution time. Finally, the accuracy and execution time are analyzed with respect to the number of particles used.

All system parameters are set as given in the source code initialization. The number of particles used is 500.

Automotive Sequence 1

This sequence provides an occlusion-free and curved run on a German Autobahn during the day time. The weather is cloudy, providing conditions of equal light distribution . This is an ideal test sequence because there is no shadowing, there are no occlusions and the lane markings are very clear.

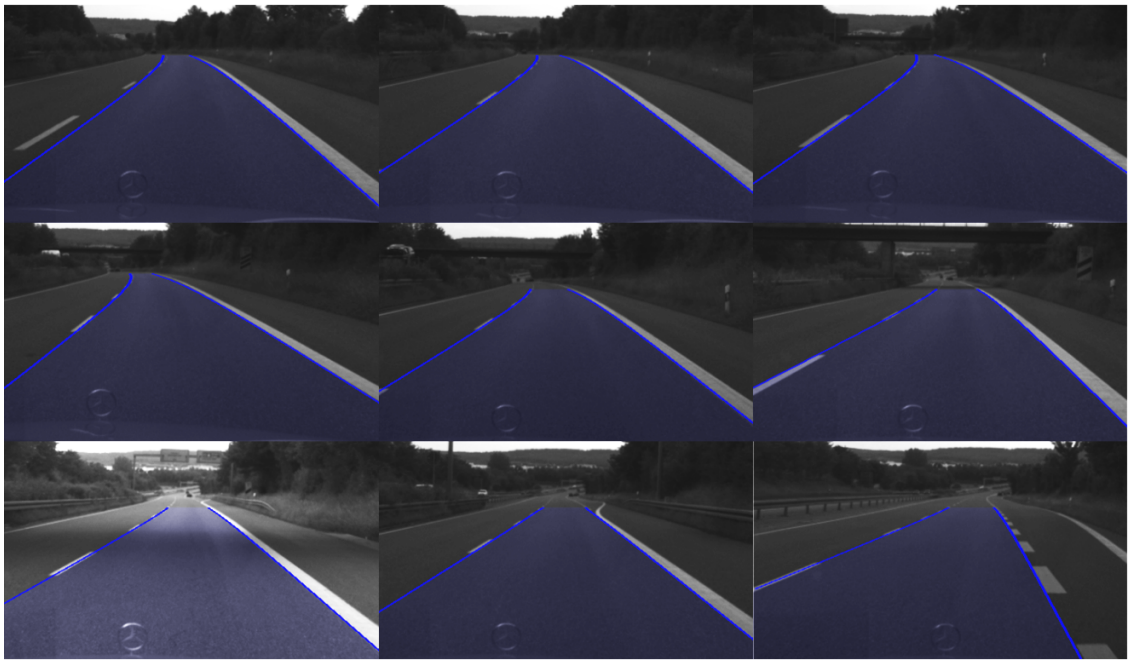

Figure 6.12: *automotive sequence 1; frame numbers from left to right and top to bottom are: 20, 80, 140, 200, 260, 320, 380, 440, 498*

The algorithm performs very accurately on 100 percent of the sequence. The only deviation occurs in the first frames, during the time of initialization, when the left lane boundary shows some deviation from the actual lane marking. The initialization takes quite some time (about 20 to 30 frames), this could be improved by dynamically setting the number of initial particles.

Automotive Sequence 2

This sequence provides a straight run on a German Autobahn at bright day time. There are some occlusions in front and in the neighboring lane. The weather is cloudy, thus providing conditions of equal light distribution.

The algorithm performs well on 70 percent of the sequence. The lane estimate occasionally jumps to one of the neighboring lanes, which demonstrates the necessity of multimodal Particle filtering (frame 480). The particle filter also occasionally tracks the car in front (frame 29). This is because of the trivial feature extraction and could be prevented by using matched filters to extract the lane.

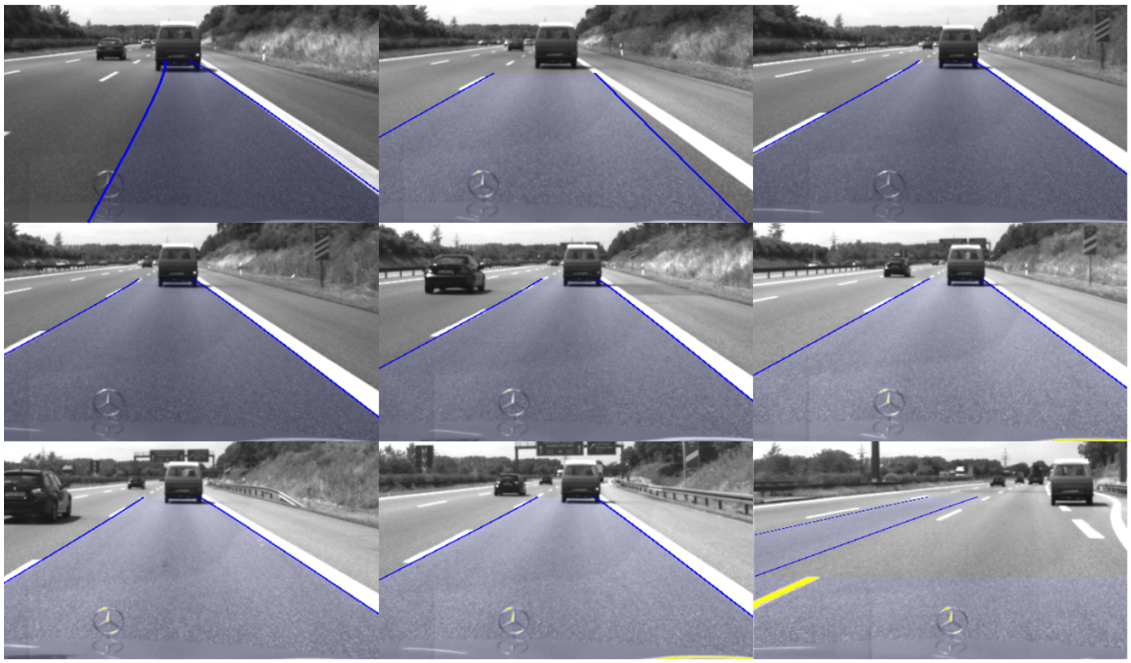

Figure 6.13: *automotive sequence 2; frame numbers from left to right and top to bottom are: 29, 80, 140, 200, 260, 320, 380, 440, 480*

Automotive Sequence 3

This sequence provides a slightly curved run on a German Autobahn during the evening. There are some occlusions in front and in the neighboring lane. In addition, a lane change maneuver is performed.

The algorithm performs well on 90 percent of the sequence. Inaccuracies are introduced when changing the lane, in this situation some extra curvature is introduced (frame 80). This probably points to inaccurate measurement of the motion variables or an error in the evolution model, which would need further investigation.

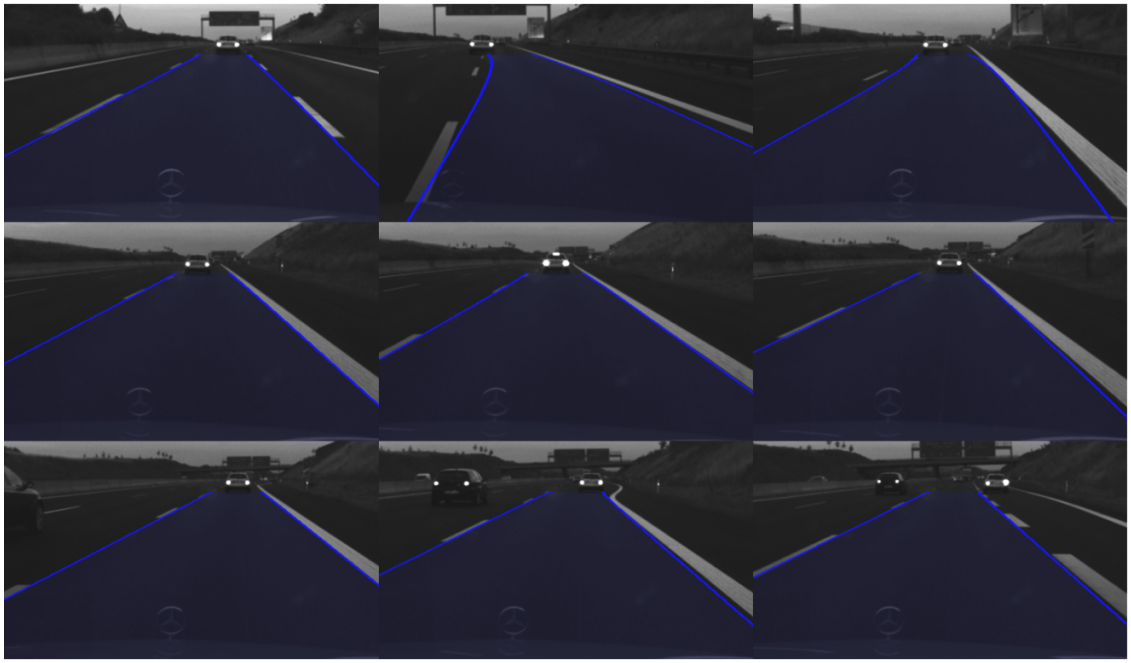

Figure 6.14: *automotive sequence 3; frame numbers from left to right and top to bottom are: 20, 80, 140, 200, 260, 320, 380, 440, 498*

Automotive Sequence 4

This sequence provides a curved run on a German Autobahn at night time. There are some occlusions in front and a blending effect from the adjacent lane. Furthermore, there are some edge structures in the middle of the lane surface. In addition, the driver performs a drift off the lane.

This sequence is the most challenging, because of the bad light conditions and the additional clutter on the road surface. The lane is only detected correctly for about 30 percent of the sequence. The rest of the time the lane jumps to the neighbor lanes (frame 20) and in about 40 percent of the frames it sticks to the clutter line (frames 140, 200, 320).

Figure 6.15: *automotive sequence 4; frame numbers from left to right and top to bottom are: 20, 80, 140, 200, 320, 498*

6.5.2 Algorithm Execution Time

Particle filtering is known as computationally expensive tracking method. The following figures show the execution times of th algorithm using 500 particles to get a good idea of the bottle necks in the implementation and the Particle filter performance using an off the shelf computer. Figure 6.16 shows the major modules that contribute to the total execution time. The achieved frame rate is 20 fps.

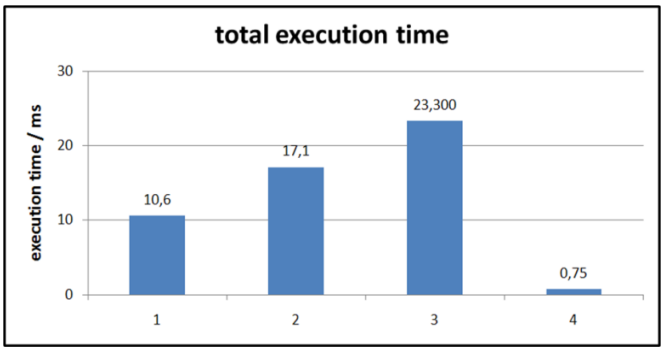

Figure 6.16: *execution times of: 1) preprocessing, 2) feature extraction, 3) Particle filtering, 4) visualization*

The preprocessing takes about 20% of the overall execution time. It consists of two steps, firstly, preparation of the output image, which needs about 1% and secondly, the transformation of two different images to world map space which needs about 19%. The execution times of these steps are shown in Figure 6.17 (right diagram).

The feature extraction takes about 33% of the overall execution time. About 25% is needed for computing the gradient magnitude and direction images. The remaining 8% is needed to threshold filter the images in respect to the direction information. These execution time values are illustrated also in Figure 6.17 (left diagram)

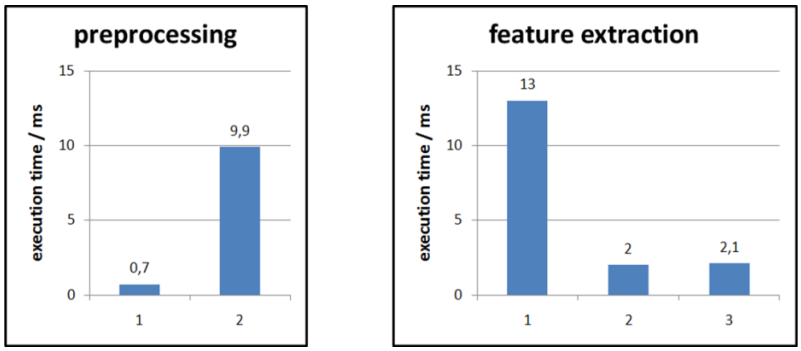

Figure 6.17: *execution times of: 1) preprocessing, 2) feature extraction, 3) Particle filtering, 4) visualization*

The Particle filtering demands the most percentage of the execution time, at about 45%. The particle loop, which is the particle evolution and evaluation, takes most of this, 35%. To compute the integral image, which is needed for the particle evaluation, demands 3.5%. All remaining functions have only a minor contribution of about 7%. The execution time values of the Particle filter are illustrated in Figure 6.18.

In considering these results, it is clear that one needs to find a good trade off between a proper designed difference measure and the maximum execution time which the

evaluation is allowed to have. In this implementation, the trivial method showed good results and was therefore chosen.

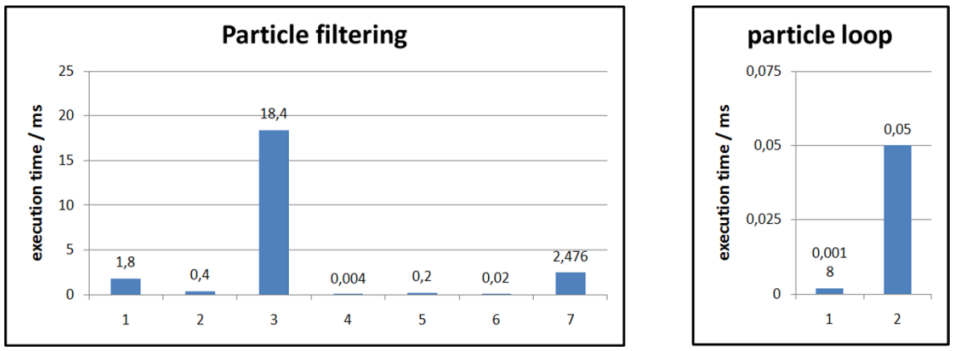

Figure 6.18: *execution times of: 1) preprocessing, 2) feature extraction, 3) Particle filtering, 4) visualization*

All in all, the Particle filter implementation showed good performance results of 20 fps. It is possible to achieve even higher performance by using a LUT for the IPM function or by using multi core functionality. Due to the highly parallel structure of Particle filtering, a 10-times performance increase is expected when using modern GPU hardware.

6.5.3 Variation of Number of Particles

The question of how many particles should be used is also essential. It affects the overall accuracy in terms of tracking the lane, the stability in terms of finding the lane and finally the execution time of the system as discussed in the section above.

To get the best trade off, the system was tested with varying numbers of particles - from 100 to 500 in 100-steps, and additionally 850 and 1000 particles. The average distance of left and right lines as well as the total execution time was averaged over time to get a single accuracy value and performance value respectively. The results are shown in Figure 6.19.

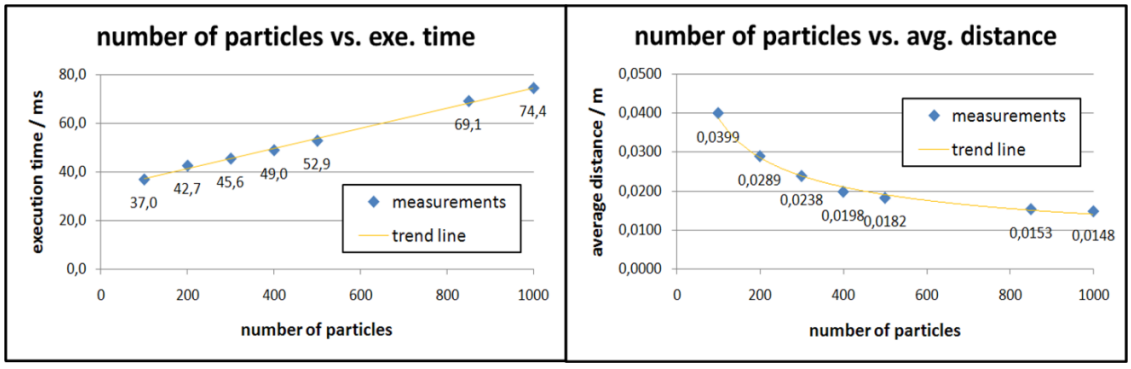

Figure 6.19: *analysis of number of particles in terms of performance and accuracy*

Even though 200 particles gave a sufficient average distance measure, the algorithm was quite instable in terms of detecting the lane. The final conclusion is that 400 to 500 particles give a stable lane detection and an accurate lane tracking performance and were found to be sufficient for the given test scenarios.

7. Conclusion and Outlook

In this project, a lane detection and tracking (LDT) algorithm for highway and major urban road applications was designed and successfully implemented where the system requirements have been met to a high degree.

The algorithm developed here is similar to those proposed in other current research, [SoTa01], [DNMT08] and works robustly for highway scenes with occlusions present. The system design focused especially on investigating the Particle filter as tracking module for LDT systems. For this reason, this thesis presents a comprehensive introduction to Particle filtering and its underlying theoretical background.

The implementation allows the system parameters to be tuned, enabling the filter state to be controlled, and the video sequence to be stopped or restarted during runtime. This provides a tool for investigating and demonstrating the dynamics of the particle filter, which is of particular interest for academic teaching purposes.

A small measurement unit for recording automotive scenes was also designed and implemented. The unit includes hard- and software to capture the automotive images, to measure the vehicle's speed sensor signal and to record measurements from an intrinsic measurement unit.

Two major implementation issues of Particle filtering regarding LDT systems have been identified. Firstly, the definition of the difference measure that is needed to update the estimate by means of measurements must be specifically designed for every application. The second problem is the inability of original Particle filters to track multimodal likelihoods because of structural reasons. Both issues have been investigated and suggestions for solving them have been given.

In designing the algorithm, it was found that even though the Particle filter is able to deal with clutter at a high level, a sophisticated feature extraction method which also reduces clutter at a low level, would improve the robustness significantly.

The algorithm works well for clear lane markings, but lacks robustness in situations of high clutter. The feature extraction is based on the Sobel operator and uses gradient magnitude in combination with the gradient direction information to detect several types of road features. Because of choosing this approach, the algorithm still

has problems handling weak road features with strong clutter present. For this reason, it would make sense for future work to focus on a more sophisticated feature extraction scheme which could handle weak road features better. The various road feature types could be addressed specifically by using matched filters and introducing a Markov state model to control which feature extraction method is to be used.

The feature extraction could be further improved by employing a method that detects whether the feature is traveling together with the road surface or not. Then, road features could be identified by their position in the current and the previous frame.

Further investigation of the difference measure could also increase the overall system performance. Suggestions for future development include: introducing structural aspects to the difference measure as described in the design chapter, and using the EM-Algorithm to increase the weights of dashed lines compared to solid lines.

In this algorithm, stabilizer lines were added as a method of increasing robustness. They did improve the robustness in some situations, but unfortunately, were not flexible enough to adapt to the environment. However, the underlying idea of adapting the model to the environment makes sense, and is definitely the right way to go.

Two suggestions of how to do so are, firstly, to use a single lane model with a multimodal Particle filter design with high observation variance to get a rough estimate of the environment. Based on this information, a more complex road model could be used for tracking. One would need a road model that is able to adapt to the situation, so that lanes can be added or removed.

A second possibility is to use global positioning system (GPS) information in combination with a pre-generated map. The map could be formed into an adaptive model which is elastic and the deformation could be described by a few parameters.

The Particle filters barrier is the high computational effort for complex systems. Because of steadily increasing computational power of high speed information processing systems, this barrier is getting lower and lower and puts Particle filtering in an attractive position for all sorts of estimation problems. This thesis provides a wide range of basic considerations of Particle filtering and offers a basic LDT algorithm which can be extended in many ways.

A. Camera Model

A.1 Euclidean versus Perspective Space

The camera model describes the mathematical relation between coordinates of a three-dimensional point in the real world, and coordinates of its two-dimensional projection onto the image plane. The real world is mathematically described by the *object space* and is denoted as $\mathbb{P}^3$. The image plane is represented by the *image space* which is denoted as $\mathbb{P}^2$.

Both, object and image space are *perspective spaces* $\mathbb{P}^n$ which have, as distinguished from Euclidean spaces $\mathbb{R}^n$, an additional vector element. In other words, a point in a n-dimensional perspective space is represented by a (n+1)-tuples of coordinates. Given coordinates in $\mathbb{R}^n$, projective coordinates can be built by the correspondence

$$[x_1, \cdots, x_n]^T \rightarrow [x_1, \cdots, x_n, 1]^T.$$

The inverse transform, from perspective to Euclidean space, is performed by simply dividing the coordinates by the vector element (n+1):

$$[x_1, \cdots, x_n, x_{n+1}]^T \rightarrow [\frac{x_1}{x_{n+1}}, \cdots, \frac{x_n}{x_{n+1}}]^T.$$

In contrast to Euclidean coordinates, scaling of perspective coordinates by a non-zero factor is not significant, so that $[\lambda u, \lambda v, \lambda]$ and $[u, v, 1]$ are representing the same point in image space.

A.2 Camera Calibration

In general, camera calibration addresses the problem of finding the projection matrix K by using either numerical or analytical methods. In this thesis, camera calibration is reduced to simply finding the homography H of $\mathbb{P}^2$ between the ground plane and the image plane.

The homography matrix has nine, with neglecting the scale factor, only eight entries. Since each point correspondence (m',m) yields two independent equations, four point correspondences are sufficient to calculate the homography matrix.

The equations are:

$$\frac{u_p}{\psi} = \frac{h_{11}X_{W,p} + h_{12}Y_{W,p} + h_{13}\Psi}{h_{31}X_{W,p} + h_{32}Y_{W,p} + h_{33}\Psi}, \qquad \frac{v_p}{\psi} = \frac{h_{21}X_{W,p} + h_{22}Y_{W,p} + h_{23}\Psi}{h_{31}X_{W,p} + h_{32}Y_{W,p} + h_{33}\Psi}, \tag{A.1}$$

which can be rearranged to

$$\begin{aligned} h_{11}\psi X_{W,p} + h_{12}\psi Y_{W,p} + h_{13}\psi\Psi - h_{31}u_p X_{W,p} - h_{32}u_p Y_{W,p} - h_{33}u_p\Psi = 0, \\ h_{21}\psi X_{W,p} + h_{22}\psi Y_{W,p} + h_{23}\psi\Psi - h_{31}v_p X_{W,p} - h_{32}v_p Y_{W,p} - h_{33}v_p\Psi = 0. \end{aligned} \tag{A.2}$$

H is determined uniquely by solving the system of linear equations which is set up from these equations based on four non-collinear point correspondences.

List of Figures

References

[Aly08] M. Aly. Real time detection of lane markers in urban streets. In *Proc. Of Intelligent Vehicles Symposium, Eidenhoven, the Netherlands*, 2008, S. 7–12.

[ApZe03] N. Apostoloff und A. Zelinsky. Robust vision based lane tracking using multiple cues and particle filtering. In *Proc. IEEE intelligent vehicles symposium*, 2003.

[Brog95] A. Broggi. Robust real-time lane and road detection in critical shadow conditions. In *Proceedings IEEE International Symposium on Computer Vision*, 1995, S. 353–358.

[CaCF99] J. Carpenter, P. Clifford und P. Fearnhead. Improved particle filter for nonlinear problems. *IEE Proceedings-Radar, Sonar and Navigation* 146(1), 1999, S. 2–7.

[CrDo02] D. Crisan und A. Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on signal processing* 50(3), 2002, S. 736–746.

[DCPP05] R. Douc, O. Cappé, E. Polytech und F. Palaiseau. Comparison of resampling schemes for particle filtering. In *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, 2005, S. 64–69.

[DDFMR00] A. Doucet, N. De Freitas, K. Murphy und S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*. Citeseer, 2000, S. 176–183.

[DiMy92] E. Dickmanns und B. Mysliwetz. Recursive 3-D road and relative ego-state recognition. *IEEE Transactions on pattern analysis and machine intelligence* 14(2), 1992, S. 199–213.

[DNMT08] R. Danescu, S. Nedevschi, M. Meinecke und T. To. A stereovision-based probabilistic lane tracker for difficult road scenarios. In *2008 IEEE Intelligent Vehicles Symposium*, 2008, S. 536–541.

[DoDFG01] A. Doucet, N. De Freitas und N. Gordon. *Sequential Monte Carlo methods in practice*. Springer Verlag. 2001.

[FrJo00] U. Franke und A. Joos. Real-time stereo vision for urban traffic scene understanding. In *IEEE Conference on Intelligent Vehicles, Dearborn*, 2000.

[FrLK07] U. Franke, H. Loose und C. Knoppel. Lane recognition on country roads. In *2007 IEEE Intelligent Vehicles Symposium*, 2007, S. 99–104.

[GoSS93] N. Gordon, D. Salmond und A. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings*, Band 140, 1993, S. 107–113.

[GWZX+08] J. Gong, A. Wang, Y. Zhai, G. Xiong, P. Zhou und H. Chen. High speed lane recognition under complex road conditions. In *2008 IEEE Intelligent Vehicles Symposium*, 2008, S. 566–570.

[HoSG06] J. Hol, T. Sch
"on und F. Gustafsson. On resampling algorithms for particle filters. In *Nonlinear Statistical Signal Processing Workshop*, 2006, S. 79–82.

[Jayn86] E. Jaynes. Bayesian methods: General background. *Maximum Entropy and Bayesian Methods in Applied Statistics*, 1986, S. 1–25.

[Kita96] G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 1996, S. 1–25.

[LiCh98] J. Liu und R. Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American statistical association* Band 93, 1998, S. 1032–1044.

[LoFS09] H. Loose, U. Franke und C. Stiller. Kalman Particle Filter for Lane Recognition on Rural Roads. 2009.

[LSBL+08] C. Lipski, B. Scholz, K. Berger, C. Linz, T. Stich und M. Magnor. A fast and robust approach to lane marking detection and lane tracking. In *IEEE Southwest Symposium on Image Analysis and Interpretation, 2008. SSIAI 2008*, 2008, S. 57–60.

[McTr06] J. McCall und M. Trivedi. Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. *IEEE Transactions on Intelligent Transportation Systems* 7(1), 2006, S. 20–37.

[MDDFW01] R. Van der Merwe, A. Doucet, N. De Freitas und E. Wan. The unscented particle filter. *Advances in Neural Information Processing Systems*, 2001, S. 584–590.

[NSGD+04] S. Nedevschi, R. Schmidt, T. Graf, R. Danescu, D. Frentiu, T. Marita, F. Oniga und C. Pocol. 3D lane detection system based on stereovision. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, 2004, S. 161–166.

[PiSh99] M. Pitt und N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 1999, S. 590–599.

[RiAG04] B. Ristic, S. Arulampalam und N. Gordon. *Beyond the Kalman filter: Particle filters for tracking applications.* Artech House Publishers. 2004.

[SaST06] F. Samadzadegan, A. Sarafraz und M. Tabibi. Automatic Lane Detection in Image Sequences for Vision-Based Navigation Purposes. In *Proceedings of the ISPRS Commission V Symposium Image Engineering and Vision Metrology.* Citeseer, 2006.

[Simo06] D. Simon. *Optimal state estimation: Kalman, H [infinity] and nonlinear approaches.* Wiley-Interscience. 2006.

[SKAD07] S. Sehestedt, S. Kodagoda, A. Alempijevic und G. Dissanayake. Efficient lane detection and tracking in urban environments. In *3rd European Conference on Mobile Robots (EMCR 07), Freiburg, Germany*, 2007.

[SoTa01] B. Southall und C. Taylor. Stochastic road shape estimation. In *Proc. int. conf. computer vision.* Citeseer, 2001, S. 205–212.

[STHS08] A. Saudi, J. Teo, M. Hijazi und J. Sulaiman. Fast lane detection with Randomized Hough Transform. In *Information Technology, 2008. IT-Sim 2008. International Symposium on*, Band 4, 2008.

[Whit94] D. Whitley. A genetic algorithm tutorial. *Statistics and computing* 4(2), 1994, S. 65–85.