ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG
FACULTY OF APPLIED SCIENCES

Department of Computer Science
Autonomous Intelligent Systems Laboratory
Prof. Dr. Wolfram Burgard

Characterizing Dynamic Objects in 3D Laser Range Data

Master Thesis

Author:	Jürgen Michael Hess
Submitted on:	11. November 2008
Supervisors:	Prof. Dr. Wolfram Burgard Dipl.-Inf. Christian Plagemann M.Sc. Daniel Meyer-Delius

Zusammenfassung

Das Erkennen dynamischer Objekte und die Vorhersage ihrer Bewegung sind zentrale Themen der Robotik. Bewegung wird häufig als die Verschiebung der Position eines Objekts über die Zeit definiert. Diese Definition führt zu einer der grundlegenden Ideen um Bewegung zu erkennen. Wenn ein Objekt in verschiedenen Messungen identifiziert werden kann, kann die Bewegung durch dessen Positionsveränderung bestimmt werden. Bei bisherigen Ansätzen ist zunächst ein Modell des Objekts erforderlich, um ein Objekt zu erkennen. Weiterhin muss das Objekt auch in einer folgenden Messung erkannt und identifiziert werden. Zusätzlich zur Erkennung von Objekten ist es also notwendig, das Problem der Datenassoziation zu lösen. Diese Probleme gaben die Anregung für die Kernidee dieser Arbeit. Anstatt ein Objekt zunächst zu erkennen um dann dessen Bewegung vorherzusagen, soll die Bewegung eines Objekts bereits vor dessen Identifizierung geschätzt werden. Für einige Anwendungen ist es wichtig, die Bewegung verschiedener Objekte schnell und zuverlässig vorherzusagen. Wichtiger als das Objekt zu identifizieren, ist zu wissen, wo, wohin und mit welcher Geschwindigkeit sich etwas bewegt. Diese Arbeit schlägt einen Ansatz zur Bewegungsvorhersage vor, der ausschliesslich lokale Merkmale verwendet. Das Ziel ist, auf Grund lokaler Veränderungen der Umgebung und der daraus gewonnenen Merkmale eine Bewegung vorherzusagen. Die verwendeten Merkmale sollen möglichst unabhängig von der Struktur des zugrundeliegenden bewegten Objekts sein. Basierend auf den lokalen Merkmalen wird die Bewegung mittels Gauss-Prozessen geschätzt. Dieser Ansatz hat die Vorteile, dass kein Modell des bewegten Objekts notwendig ist und es auf Grund der verwendeten Merkmale nicht notwendig ist, die Objekte miteinander zu assoziieren.

Abstract

Recognizing dynamic objects and predicting their motion are central tasks within robotics. Motion is commonly described as the displacement of an object over time. This intuitively gives rise to the idea that if the same object can be identified in two subsequent observations, we can estimate the objects movement from its displacement. To detect an object a model of the object is needed. Recognized once, the object must be identified in subsequent observations to estimate motion. Hence, the problem of data association needs to be solved. These problems motivate the key idea of this thesis. Instead of detecting the objects first and then estimating their motion the idea is to detect motion first. For some relevant applications, the agent needs to be able to detect movement quickly. It is more important to know something is moving, into which direction and at what speed than what kind of object is actually causing the movement. This thesis presents an approach to estimating motion that only relies on local features. The goal is to infer motion by detecting local changes in the environment. We construct features describing these local changes such that they are as independent of the underlying moving object as possible. Motion is inferred from these local features using Gaussian process regression. This approach offers some distinct advantages. Firstly, no model of the underlying dynamic object is required because the local features are similar for different types of objects. Secondly, the problem of associating objects does not need to be explicitely solved.

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Jürgen Hess

Freiburg, den 11. November 2008

Acknowledgment

I want to thank all the people that supported my work, especially my supervisors Prof. Dr. Wolfram Burgard, Christian Plagemann, and Daniel Meyer-Delius for the interesting topic, their support, and ideas. I would also like to thank all the people of the “Autonomous Intelligent Systems” group in Freiburg, especially Patrick Pfaff and Rainer Kuemmerle for their help with the dataset, Felix Endres and Matthias Luber for the fruitful discussions and Kai Wurm for the technical support. I thank Dirk Hähnel from Stanford University for providing the laser range data. I especially thank my German and American family who always supported me during my studies and my girlfriend Marina Krapf, who supported me very much in various ways.

Contents

1 Introduction **1**

1.1 Related Work 3

1.2 Outline 5

2 From Pointclouds to Motion Features **9**

2.1 Data Preprocessing 9

2.1.1 Mapping the Environment 10

2.1.2 Map Differencing 13

2.2 Motion Features 16

2.2.1 Clustering the Pointcloud 16

2.2.2 Map Patches as Local Motion Features 18

2.2.3 Extracting Features with Gabor Filters 18

3 Motion Perception and Estimation **23**

3.1 Predicting Motion using Gaussian Processes 23

3.1.1 Motion Estimation as a Regression Problem 24

3.1.2 Selecting Samples for Learning 25

3.1.3 Gaussian Process Regression 27

3.1.4 Merging the Predictions 37

3.2 Alternative Approaches to Motion Estimation 37

3.2.1 Predicting Motion using the Iterative Closest Point Algorithm 38

3.2.2 Predicting Motion using the Constant Velocity Motion Model 39

4 Evaluation **41**

4.1 Experimental Setup 41

4.2 Evaluation Methods 42

4.3 Evaluation of the Features 43

4.3.1 Heading Prediction 44

4.3.2 Speed Prediction 46

4.4 Comparison on Real Data 46

4.5 Discussion 63

5 Conclusions **65**

6 Future Work **67**

Bibliography **69**

1 Introduction

Recognizing dynamic objects and predicting their motion are central tasks within robotics. In recent years, new sensors like radars and long range scanners have been developed that make outdoor navigation at high speeds possible. This led to a growing interest in application domains like driving assistance and autonomous driving. For these applications, a reliable detection and tracking of moving objects is indispensable. This is, however, also one of the most demanding problems. Consider the traffic situation depicted in Figure 1.1. There is a large number of moving objects of many different types, for instance cars, trucks, pedestrians, motorcycles, etc. To complicate matters even more, objects of the same type have different shapes and colors. In addition, dynamic objects might move at a different speed and into different directions. To operate an agent in this environment an accurate and reliable estimation of the motion of these moving objects is essential.

Motion is commonly described as a change in the position of an object over time. This intuitively gives rise to the idea that if the same object can be identified in two subsequent observations, we can estimate the objects movement from its displacement. Regardless of the sensor used for object detection (e.g. laser or camera) this approach generally leads to several problems:

1. **Modeling:** In order to identify an object, a model of the object or a class of objects is needed. Choosing the right model representation is a challenging problem because models need to be both discriminative and general. They have to be general enough, to robustly capture instances belonging to the same class, but also be discriminative enough so that objects are not assigned to a false class. For instance, consider constructing a model of a class of objects like cars. This is a hard task because of the great variety of the instances of objects within these classes. Just consider Figure 1.1 which shows a traffic scene is Paris and the different shapes of the trucks and the smaller cars. Additionally, it is not clear what kind of object classes should be modelled. In case of traffic situations, the most important objects to be modelled are probably cars, pedestrians and bicycles. But what happens when a motorcycle is observed?

2. **Occlusions:** Due to the nature of most sensors objects, can only be observed partially. If only a part of an object can be observed it might not fit any of the models previously learned. For instance, for any driver in Figure 1.1 some parts of the scene are just blocked by other cars. Hence, the driver might only observe some of the objects partially and others not at all.

3. **Data Association:** An object must not only be robustly detected in subsequent observations, it also must be assured that the object found is really the same object in order to infer

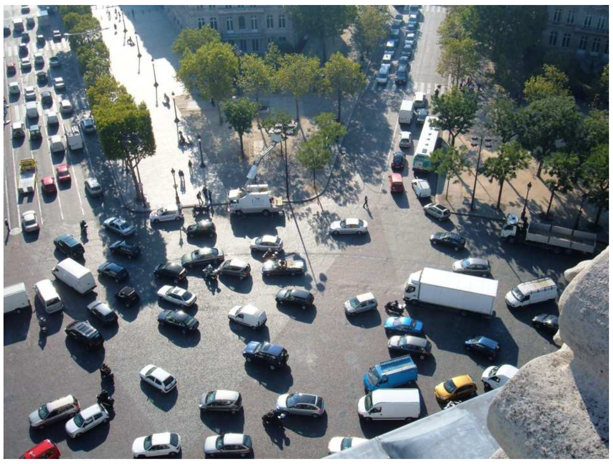


Figure 1.1: Snapshot of a traffic scene taken in Paris from the top of the Arc de Triomphe. (© BrokenSphere/Wikimedia Commons)

motion. Especially in the case of a moving observer the appearance of an object might change even if the object is static, just because the object is seen from another side.

These problems motivate the key idea of this thesis. Instead of detecting the objects first and then estimating their motion the idea is to detect motion first before recognizing the dynamic objects. For some relevant applications, quick and reliable movement detection is required. It is more important to know something is moving, into which direction and speed than to identify what kind of object it is.

Techniques for movement detection can generally be divided into three main categories: appearance-based, feature-based and motion-based. *Appearance-based* methods try to detect objects based on their appearance. The idea is that if an object and its position can be identified in two consecutive observations, the movement can simply be obtained by measurering the displacement of the object. This approach also matches the intuitive understanding of motion, the displacement of an object over time. These approaches are able to recognize objects even if they are not moving. *Feature-based* motion detection approaches realize that recognizing the same object in subsequent observations might not always be possible because the appearance might change, for instance if a part of an object is occluded in one of the observations. Instead of matching recognized object these methods match feature like lines or corners which are assumed to be more stable. *Motion-based* approaches detect moving objects by determining motion queues in the observations. A measurement is considered dynamic if its motion queue does not match the expected global motion, which is derived from the motion of the observer. Estimating motion is usually a hard problem but has the distinct advantage of not relying on prior knowledge about the form or the features of a dynamic object.

This thesis presents an approach to estimating motion that only relies on local motion features. The goal is to infer motion by detecting local changes in the environment and constructing features describing these local changes such that they are as independent of the underlying moving object as possible. This approach offers some distinct advantages for estimating motion:

- No model of the underlying dynamic object is required.
- The problem of associating objects from different measurements does not need to be explicitely solved. (The data association in our approach is solved by constructing a grid map of the environment.)

1.1 Related Work

Detection and tracking of moving objects has been studied for several years and there is a large amount of literature on this topic. Recently, the development of reliable motion detection techniques was encouraged by the Urban Challenge 2007, an autonomous driving competition where cars had to manage real driving situations in an urban environment with other dynamic objects. The state of the art methods in this field usually rely on visual and laser range information. In the following we give an overview of these methods.

Vision-based methods

Moving object detection and tracking was based on vision first because cameras are cheap and small sensors and have been available for many years. Vision-based approaches started by using monocular cameras and were appearance and feature-based. Dellaert and Thorpe [1997] apply the Hough transform to gradient images to extract line segments from which prior hypothesis of moving vehicles are generated. Also using a monocular camera, Ferryman *et al.* [1998] extract linear features for constructing a model of the observed object. Motion-based vision approaches rely on computing a motion flow field for a sequence of images which means that a point-to-point correspondence between pixels of different images is found. Depending on the criteria to be optimized there are several methods available for finding point-to-point correspondences. Talukder *et al.* [2003] use optical flow to determine the correspondence. Haag and Nagel [1999] combine optical flow and edge extraction to achieve a more stable result. Another approach to estimate the flow field of vehicles is suggested by Di Stefano and Viarani [1999]. They use the block-matching algorithm which separates an image frame into smaller blocks and finds the best matching block in the other image.

Vision-based methods in general have problems reliably operating under changing lighting conditions. In case of an autonomous agent operating outdoors the recognition of moving objects must be possible at all times, day and night. Additionally, cameras have the disadvantage that they do not provide the distance to objects explicitely. Although it might be possible to determine which part of an image is moving, the estimation of the speed and direction is much more complicated, compared to using a sensor that directly provides the distance information. Because of these shortcomings mainly laser range sensors are now used for detecting and tracking dynamic objects.

Range-sensing-based methods

Most methods for detecting motion in range scans are feature-based. Typically, these methods segment the laser scan first and take each segment as a prior hypothesis of a moving object. The association of objects from different scans is solved by matching features extracted from the object hypothesis in both scans. Having obtained an association for the objects, they are tracked using a Bayesian filtering scheme which also provides an estimate of the future motion of the object. Fuerstenberg *et al.* [2002], Castro *et al.* [2004] and Navarro-Serment *et al.* [2008] segment the range data and track the center of mass of the segments with a Kalman filter while solving the data association problem using the nearest-neighbor approach. These methods are mainly applied to indoor people tracking. For tracking, a fixed reference point on the object is needed to correctly estimate the motion of an object. For tracking people and small objects, the center of mass of a range scan segment seems to be a good reference point because the area occupied by a person is usually a small, circular region. For instance, Luber *et al.* [2008] use an extended Kalman filter to track moving objects like people, bicycles and skaters. Unfortunately, the center of mass of a larger object is not a good reference point. For instance, if only a small part of a car is seen and the larger part is occluded the appearance of the object and also the observed center of gravity might change, even if only the observer moves and not the object itself. In this case one would detect motion even if the observed object were static.

The problem of changing appearance is commonly tackled by fitting a parametric model to the observed data. Morris *et al.* [2008] extract linear features like corners and line segments from the segmented laser scan which are fit to the scan points using the RANSAC algorithm (Fischler and Bolles [1981]). From these features the object center is extracted and tracked using a Kalman filter. Similarly, Zhao and Thorpe [1998] and MacLachlan and Mertz [2006] approach the problem of changing appearance by extracting and matching linear features like corners of objects. To improve tracking, they then fit a rectangular box to the moving object assuming the objects to be tracked are cars. Usually, the data association is solved using the nearest-neighbor approach. MacLachlan and Mertz [2006] additionally observe that for tracking vehicles it is sufficient to assume that the parametric model of a vehicle (a rectangular box) overlaps in subsequent scans. Hence, they associate the objects by determining which models overlap. The methods parametrically modeling dynamic objects are primarily designed to track vehicles and rely on much a priori knowledge about the objects. Parametric models generally have problems adapting to unknown objects. Our approach does not require a model of a dynamic object and thus is not limited to known object types.

In addition to extracting features directly from range data, other methods introduce map representation of the environment for extracting hypothesis of dynamic objects. Petrovskaya and Thrun [2008] construct a radial difference map of the environment, separate the map into clusters and calculate the center of gravity of the clustered segments which serves as reference point for tracking. To keep the calculated reference point fixed they use a particle filter to estimate the geometric properties (width and length) of the dynamic objects. To the knowledge of the author, they are the only ones using a particle filter for tracking the dynamic object hypothesis. Just like the other methods fitting a parametric model, their approach lacks the ability of adapting to unknown objects. Additionally, the performance of this method largely depends on the quality of the model fitting.

Bobruk and Austin [2004] suggest an occupancy grid based approach. They construct an occupancy grid map for each range scan and construct a difference map from which moving object hypothesis are generated by clustering map regions where occupancy changes occurred. The object hypothesis are tracked using a Kalman filter. Prassler *et al.* [2000] suggest a closely related approach but argue that building an entire occupancy grid map is not efficient. They generate a grid map which stores for each cell a list of timestamps at which the cell was observed. Free space is not considered in this approach. The map is clustered into areas in which occupancy changes occurred. These are considered hypothesis of moving objects and are matched to the clusters derived from the previous occupancy map, also using the nearest-neighbor method. Since they also track humans in dense areas, they suggest to propagate the motion estimate directly without using a Bayesian filter because there is no real model for human locomotion in such crowded areas. These grid based approaches are closely related to this thesis because they also model motion as changing map occupancy. However, although using a grid map these approaches do not make use of the data association implicitely encoded in the difference map. The problem of matching features over time remains. Our approach instead makes use of the implicit data association of a difference map to estimate motion directly from this representation.

Motion-based approaches such as iterative closest point (Besl and McKay [1992]) and range flow (Spies *et al.* [2002]), an adaptation of optical flow for range data, have also been applied. They are most commonly used for correcting the pose estimation to improve localization and mapping. Wang *et al.* [2007] suggest a very interesting approach to moving object detection. They assume that dynamic objects are measurements that do not agree with the SLAM result. These measurements are then stored in a moving object map which is segmented for deriving initial hypothesis for moving objects. The data association of objects in subsequent scans is then determined by the best match calculated from ICP. The advantage of this approach is that no parametric model of the dynamic objects is required. However, ICP may converge to a local optimum solution only and a good prior estimate of the motion is needed.

This thesis uses features extracted from a grid based representation for solving the data association problem. Therefore neither prior information about the new location of the object is needed nor does the data association have to be explicitly solved. Additionally, instead of parametrically modeling a dynamic object, we extract local features from which motion can be estimated. Therefore the estimation of motion becomes independent of the underlying object.

1.2 Outline

The goal of this thesis is to characterize dynamic objects such that an estimation of their movement direction and speed is possible. There are several steps needed to estimate motion from range data which are visualized in Figure 1.2. The flowchart shows four different methods and the single steps performed in each method. Processes are visualized as gray nodes while white nodes denote states. The single steps will be explained throughout this thesis. The figure shall

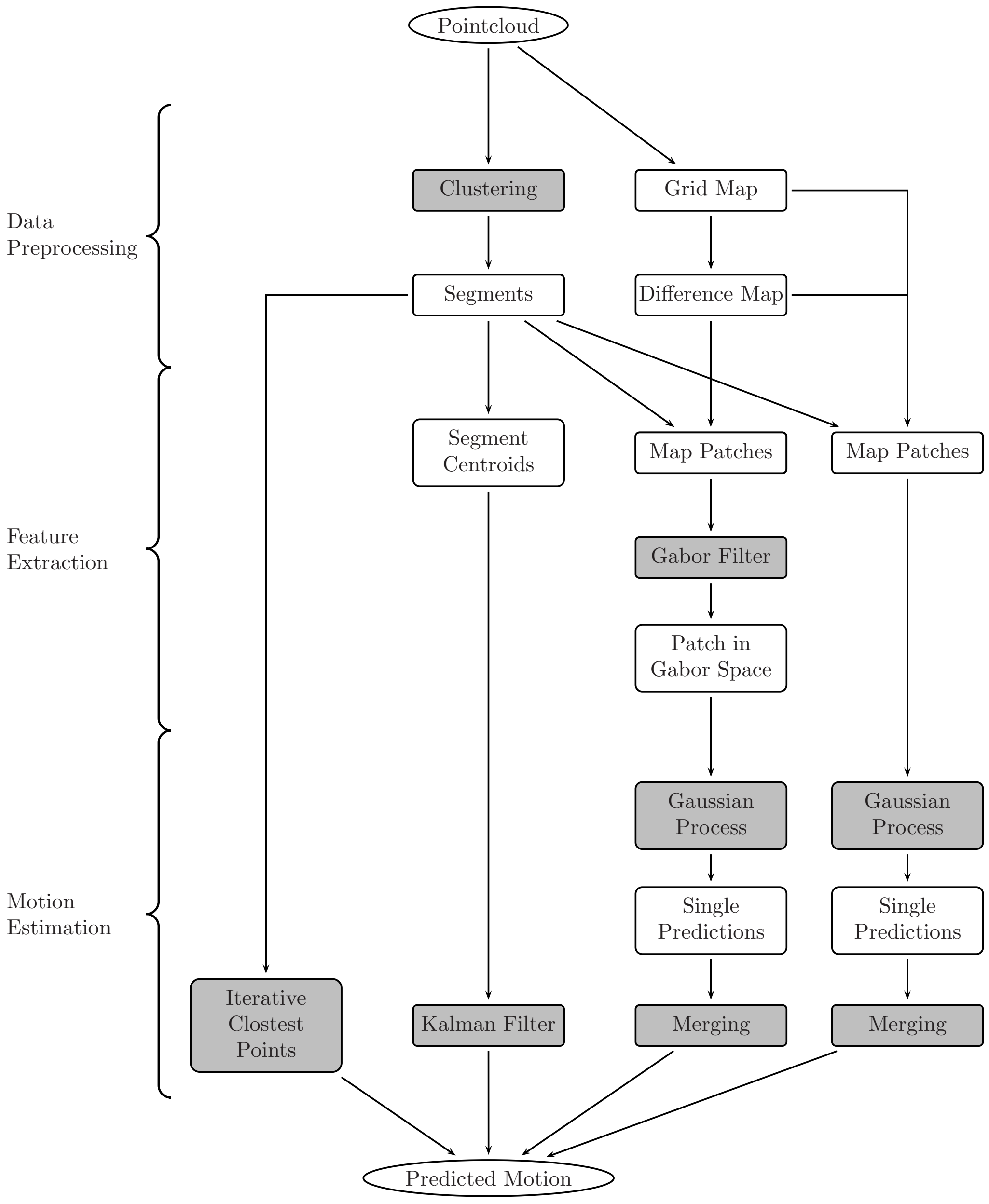


Figure 1.2: The figure depicts four possible motion estimation processes. Given the pointcloud data, the oval root of the chart, different steps are taken to predict motion. In the graph shaded nodes denote processes while white node denote states. The oval nodes mark the beginning and end states. In this thesis we propose two methods for motion estimation based on the Gaussian process regression, depicted as the first and second path from the right. The other two paths denote the methods we will compare our approach to.

serve as an overview of the methods applied and we will refer to it to clarify the stage of the estimation process. The main focus of this thesis lies on the methods for computing motion from local features by using Gaussian process regression. Additionally, motion is computed using two other state of the art methods, namely ICP and Kalman filter tracking, to be able to compare our derived method with these techniques.

This thesis is structured as follows. Chapter 2 describes the process of representing the given data such that motion features can be extracted and explains why these features are chosen. In Chapter 3 we describe Gaussian process regression in general, show why estimating motion using local features is a challenging problem and how it can be tackled using Gaussian process regression. In Chapter 4 we evaluate the quality of Gaussian process learning and in addition compare the result to ICP and Kalman filtering. The thesis closes with Chapter 5 and Chapter 6 which summarize the results and propose possible future additions or improvements.

2 From Pointclouds to Motion Features

This chapter describes the steps taken to compute features suitable for motion extraction and prediction. In the first part, we describe the preprocessing needed to obtain a representation from which motion features can be extracted. The second part of this chapter describes the extraction of the features itself from the data representation found and argues why these are suitable for motion features.

2.1 Data Preprocessing

In this section, we describe the used data and show how the data structures are constructed that can be used for feature extraction in Section 2.2. The way in which the data is processed and represented is an important issue because the choice of features directly depends on the data representation.

The data used in this thesis consists of 3D pointclouds. A pointcloud is a set of laser end points $\mathcal{Z}_t = \{z_t^i\}_1^n$ where $z_t^i = (x, y, z)$ is a single endpoint of a laser measurement taken at time t. The coordinates of the laser endpoints are given relative to the origin of the sensor. A sample pointcloud is depicted in Figure 2.1. The approach presented in this thesis does not rely on a specific sensor for generating the pointcloud but generalizes to any pointcloud. In addition to the pointcloud, the pose $\Theta = (x, y, z, \phi, \theta, \psi)$ of the robot is given where x, y, z denotes the location of the robot in its local coordinate system, ϕ denotes the pitch, θ the roll and ψ the yaw of the robot.

Generally, motion is detected by observing the environment at different times. To decide which measurements or objects are dynamic we need to solve the data association problem by identifying corresponding objects in both observations. As mentioned in Chapter 1, there are several possibilities for detecting motion using laser range data, some of which include motion detection based directly on the pointcloud data. These approaches need to solve the data association problem based on a point-to-point basis. For each point in the current pointcloud a corresponding point in the other pointcloud needs to be found. These approaches usually run into problems when objects disappear entirely or the number of points changes form one scan to the other. Hence, we would like to represent the environment in a more general way such that the representation includes the data association implicitely. One possibility for achieving this goal is by building a map of the environment. That means that we discretize the environment into rectangular cells and mark whether these cells are occupied or not based on the sensor data provided. The data association between different maps is then given implicitely by the position of the different map cells and is independent of the single data points. In this thesis, we use grid maps (see Section 2.1.1) for representing the environment at time t and for modeling changes in the environment (see Section 2.1.2).

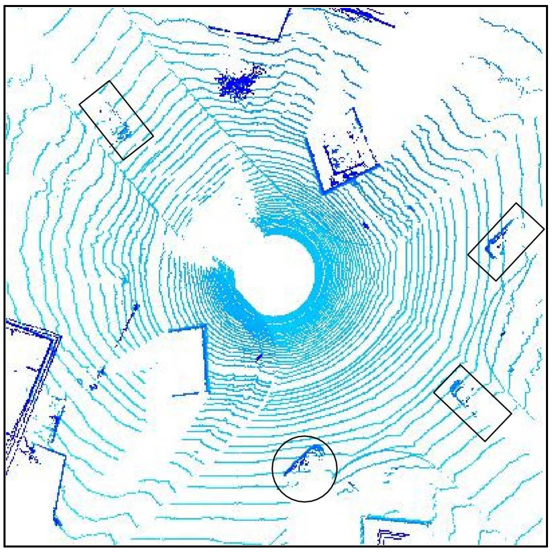


Figure 2.1: Bird-eye view of a sample pointcloud. The observer is located in the white circle, in the middle of the pointcloud.

2.1.1 Mapping the Environment

Grid maps are a well established technique to represent the environment in robotics. Mapping the environment is a hard problem for a number of reasons, for instance the noise in the perception of the sensor and the size of the environment to be mapped. Mapping 3D outdoor environments increases the size of the grids tremendously. Previous solutions to the 3D outdoor mapping problem include multi-level surface maps Triebel *et al.* [2006] which implement a efficient data structure for 3D outdoor mapping. However, in this work, the main goal is not to build a map of the environment but to construct a data structure that can be used to detect motion caused by dynamic objects. In this work a 2D projection of the 3D space is used for mapping. This means that a 2D map of the given 3D input space is used for representing the environment and detecting motion. A 2D map was chosen because it is computational efficient and also applicable to 2D pointclouds.

Figure 2.4a shows a bird-eye view of a pointcloud. In this figure, the sensor is located in the white circular area in the middle. Using a 3D laser range sensor some laser beams pass over small objects. This introduces the problem of assigning an occupancy value to small objects and leads to the question if map cells representing small objects should have an occupancy value equal to cells representing tall objects. We solve this problem by introducing a simplification and assuming that the objects of interest can be measured in a height interval between 0.5 meters and 1.0 meters. This assumption intuitively holds for many objects especially for those of interest like cars and people. It means that only scans that traverse or end in a certain height interval are considered. This is similar to generating a slice of the pointcloud at a certain height level. Figure 2.2 depicts the laser beams considered.

To construct a grid map there are several grid mapping algorithms mentioned in the litera-

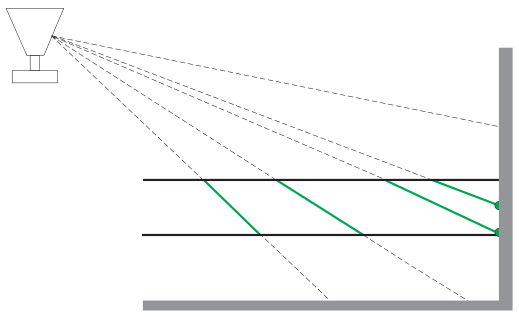


Figure 2.2: Laser measurements considered for mapping. The laser range sensor is shown at the upper left of the figure. The lines in the figure denote the laser beams. We only consider the beams and the endpoints in a certain height range. The parts of the beams considered are depicted by a solid line. The considered endpoints are denoted by circles.

ture. The two most common ones are the occupancy grid mapping algorithm, developed by Elfes [1987] in the mid-eighties and the reflection map algorithm which differ in the way the posterior occupancy probability of a map cell is estimated. The standard for occupancy grid mapping is calculating the posterior over all maps given the measurements $z_{1:t}$ and the pose $x_{1:t}$

$$p(m|z_{1:t}, x_{1:t}). \tag{2.1}$$

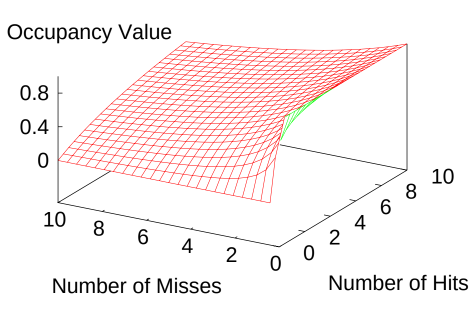
If we assume independence between the individual map cells the posterior can be written as the product of the marginals of all map cells $m^{[xy]}$ at position $< x, y >$

$$p(m|z_{1:t}, x_{1:t}) = \prod_{x,y} p(m^{[xy]}|z_{1:t}, x_{1:t}). \tag{2.2}$$

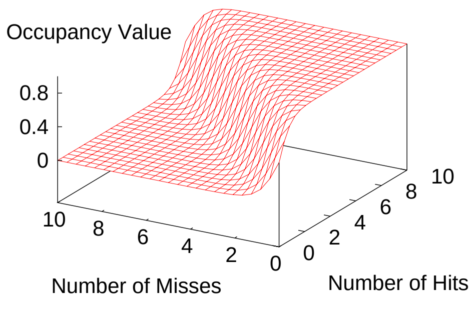
The independence assumption is of course a very strong assumption. If, for instance, a cell represents a part of a wall and therefore is occupied then the probability that neighboring cells are occupied is also very high. Because occupancy is binary, a cell is either occupied or not, the occupancy grid algorithm is based on a binary Bayes filter to find the most probable map (see Thrun [2001] for details).

The reflection map algorithm computes the probability that a map cell reflects a beam. Let $f(x_t, n, k)$ be a function that returns the index of map cell k covered by beam n at pose x_t with $k \leq z_t$. Given the sensor model

$$p(z_t|x_t, m) = \left\{ \begin{array}{rr} \prod_{k=0}^{z_t, k=n-1} (1 - m_{f(x_t,n,k)}), & \text{max range} \\ m_{f(x_t,z_{t,n})} \prod_{k=0}^{z_t, k=n-1} (1 - m_{f(x_t,n,k)}), & \text{otherwise} \end{array} \right\} \tag{2.3}$$



(a) Counting Model



(b) Occupancy Grid model

Figure 2.3: Occupancy function of a map cell calculated based on the counting model and the occupancy grid model.

we can compute the most likely map by maximizing

$$m^* = \text{arg max}\ \mathcal{P}(m|z_{1:t}, x_{1:t}). \tag{2.4}$$

Applying Bayes rule and assuming a uniform prior $p(m)$ this is equivalent to

$$\begin{aligned} m^* &= \text{arg max}\ P(z_{1:t}|m, x_{1:t}) && (2.5)\\ &= \text{arg max} \prod_{t=1}^{T} P(z_{1:t}|m, x_t) && (2.6)\\ &= \text{arg max} \sum_{t=1}^{T} \ln P(z_{1:t}|m, x_t). && (2.7) \end{aligned}$$

For the sake of brevity we omit the rest of the derivation and refer to the derivation by Hähnel *et al.* [2003] which concludes that the most likely map m^* is given by

$$p(m^{[xy]}) = \frac{\text{hits}(x,y)}{\text{hits}(x,y) + \text{misses}(x,y)}, \tag{2.8}$$

assuming the map cells to be independent. Surprisingly, the occupancy probability of a map cell can simply be calculated by counting the number of times a beam hit or missed a cell. The reflection map algorithm has the distinct advantage that the occupancy transition function is smooth.

Consider Figure 2.3a which depicts the occupancy probability of a map cell using the counting model and the occupancy grid model. While the occupancy grid method basically models a binary decision function as to whether the cell is occupied or not, the counting model results in a smoother transition between occupied and free. Motion is a continuous process where smooth changes in the environment occur. To best represent this continuous process we choose the reflection map algorithm over the occupancy grid algorithm. The hits needed for the reflection map are given by the pointcloud. The misses are computed by ray tracing the beam from the origin to the endpoint. The ray tracing is carried out after the projection of the pointcloud onto the 2D plane using the Bresenham line tracing algorithm (Bresenham [1965]). In order to account for noise in the measurement some mass of the hits and misses was spread to neighboring cells which resulted in a smoother occupancy estimate.

In the following, we use this map representation to detect motion and compute motion features. We do not map the environment over time but compute the map representation for each new pointcloud. It remains open how to set the prior for the map cells, the assumed occupancy value for cells that are not measured. Setting the prior is important because different priors result in different motion features. The next section specifies the difference map, the general data structure from which motion features will be extracted. As we will see, the properties of the difference map are important for setting the prior.

2.1.2 Map Differencing

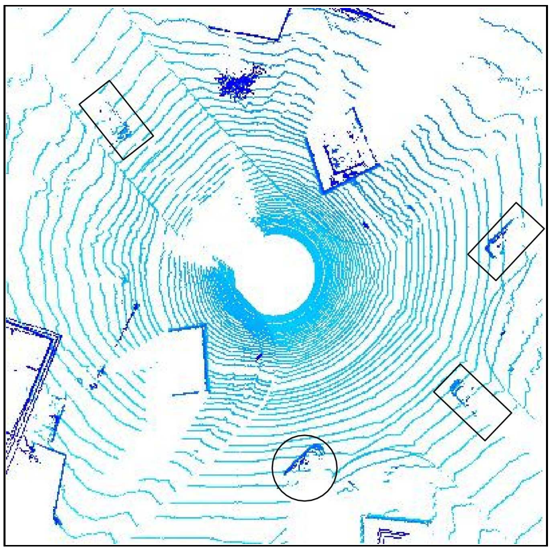
A common approach to motion detection is map differencing. Previously, this approach was applied by Petrovskaya and Thrun [2008] who use a radial map representation. Map differencing intuitively means that grid maps of the environment, representing snapshots of the environment at different points in time are compared. Given the maps m_1 and m_2 located in the same frame of reference the difference map cell $dm(m_1, m_2)^{[xy]}$ at position x, y is defined as

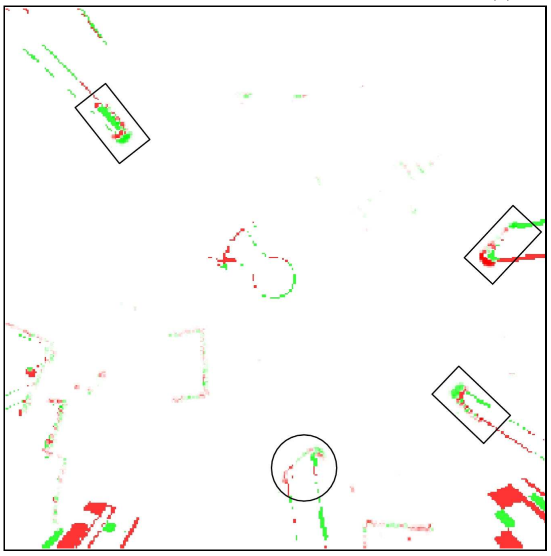
$$dm(m_1, m_2)^{[xy]} = m_2^{[xy]} - m_1^{[xy]}, \tag{2.9}$$

with $dm^{[xy]} \in \mathbb{R}[-1; 1]$. The resulting map models the change in occupancy between the two given maps. Throughout this work, this map is called difference map. In order to generate smooth changes in the occupancy values the maps used for this differencing are based on the reflection map algorithm (see Section 2.3). Figure 2.4b and Figure 2.4c depict examples of difference maps. Red (dark gray) map cells mark a loss of occupancy while green (light gray) cells mark a gain of occupancy. A dynamic object therefore moves from red (dark gray) to green (light gray). Comparing the difference map to the pointcloud depicted in Figure 2.4a gives an intuition about the appearance of dynamic objects in the difference map. Measurement noise and noise in the pose estimation can result in static objects causing occupancy changes, too. However, dynamic objects usually result in a much higher occupancy change.

Choosing the Map Prior

Choosing the prior of the map cells correctly is very important for the construction of the difference map. Depending on the prior the occupancy change of a map cell can be very different. A standard approach is choosing the prior to be 0.5 which expresses the assumption that nothing is known about a cell so the probability for being free is the same as for being occupied. A different approach to obtain the prior is to learn it form data. We estimated the prior by averaging the occupancy of a set of maps resulting in the value of 0.01. Choosing a very low prior like 0.01, on the one hand, introduces the problem that a map cell changing from the prior respectively from “unknown” to “free”, results in a very small occupancy change. On the other hand a map cell changing from “unknown” to “occupied” results a in large occupancy change. Figure 2.4 depicts


(a) Pointcloud


(b) Difference Map: Prior= 0.5

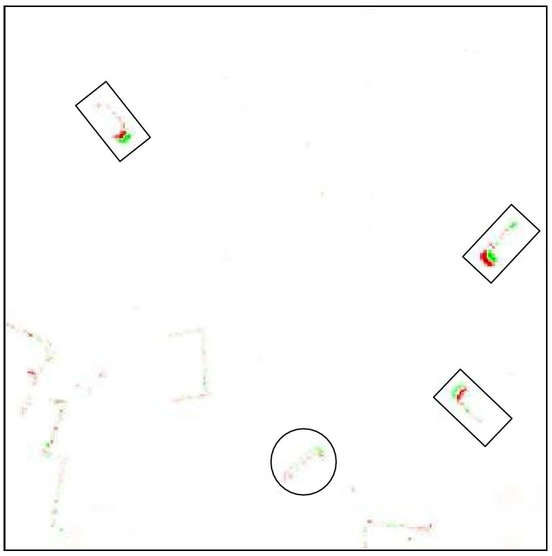

(c) Difference Map: Prior= 0.01

Figure 2.4: Example of a pointcloud and the corresponding difference maps. Note the pattern the dynamic objects, denoted by rectangles, generate on the difference map. The occupancy change is much higher for those objects then for the almost standing car denoted by the circle. A gain in occupancy is colored green (light gray) while a loss of occupancy is colored red (dark gray). The cars therefore move from red (dark gray) to green (light gray). The position of the sensor is generated is in the middle.

two difference maps computed from reflection maps using different priors. One can easily see the effect. Using a prior of 0.5 results in a higher occupancy change. It is still not obvious which prior should be preferred. In the experiments (see Chapter 4) we evaluate our approach to motion prediction using both priors, 0.5 or 0.01, to decide which one is the better choice for retrieving motion features.

We have now obtained a representation of the data that is able to capture the motion of objects. In the following section, we extract features from the difference map that are suitable for motion estimation.

2.2 Motion Features

Having obtained a data representation which is able capture motion information in form of occupancy changes in the environment, we can now describe how features for motion detection can be obtained. The goal is to first generate hypothesis of moving objects by clustering the range scan. Each of these hypothesis or clusters is then used directly for discovering movement or is extended with other information from which motion can be estimated (see Figure 1.2).

2.2.1 Clustering the Pointcloud

In order to identify object hypothesis we first cluster the given pointcloud. The general goal of clustering is to group objects in such that the objects belonging to one group are as similar to each other as possible and as dissimilar as possible to objects belonging to other groups. Here, each cluster is defined as a set of points so that each point within a cluster has a small distance to its neighbors but a larger distance to points in another cluster, according to some distance measure. Clustering methods are divided into partitioning and agglomerating methods. One of the most common partitioning method is K-Means which was introduced by MacQueen [1967]. The disadvantage of K-Means is that the distance matrix specifying the distance of all points to the cluster centroids must be calculated in each step. The runtime is bounded by $O(M * N^2)$ where M is the number of steps needed for convergence. More important for our application is that the number of clusters must be specified in advance which is not possible because the number of objects in the environment is not known. Therefore, we use the agglomerative hierarchical clustering (AHC) method which differs from the partitioning methods mainly in the principle that clusters are merged instead of partitioned in order to achieve the wanted cluster number or size. The AHC algorithm is defined as

1. Treat every object as one cluster.
2. Merge those two clusters with the smallest distance according to some distance measure.
3. Stop merging clusters when a convergence criteria is meet. Common criteria are the number of clusters or the minimal distance between the clusters.

Several distance measures have been proposed in the literature. The three most common ones used in hierarchical clustering are

- **Single-link:** The distance of two clusters equals the minimum distance of all pairs of points taken one from each cluster which is also understood as nearest-neighbor clustering. It was first introduced by Florek *et al.* [1951].

$$dist(C_I, C_J) = \min_{i \in C_I, j \in C_J} dist(i, j). \tag{2.10}$$

- **Complete-link:** The distance of two clusters equals the maximum distance of all pairs of points taken one from each cluster. This measure is also called furthest-neighbor method.

$$dist(C_I, C_J) = \max_{i \in C_I, j \in C_J} dist(i, j). \tag{2.11}$$

- **Average-link:** The distance equals the distance of the center of mass or centroids of each cluster which is defined as

$$dist(C_I, C_J) = \frac{1}{|C_I| * |C_J|} \sum_{i \in I, j \in J} dist(i,j). \tag{2.12}$$

The choice of distance measure as well as the distance metric mainly depends on the clustering application. In this thesis, the single-link distance measure using the Euclidean distance metric defined as

$$dist(i,j) = \sqrt{(i_x - j_x)^2 + (i_y - j_y)^2}, \tag{2.13}$$

is used. Note that although our original data consists of a 3D pointcloud, clustering is only performed on the x, y coordinates of the points. Clustering in 3D space would increase the problem size and in our application 2D clustering proved to be sufficient because there are no objects that overlap based on their z coordinate. The single-link distance measure has the property that long structures can evolve which is often a disadvantage. In our application, however, it is an advantage for clustering the pointcloud. The reason is that the clusters to be found in the pointcloud often are long structures like walls of houses. The single-link method makes it possible to separate the pointcloud containing a car driving by a wall into two different clusters. This must not be the case using the average-link method because it adds points to clusters depending on their distance to the center of mass of the current segments. It could happen that some points on the wall are actually closer to the center of mass of the car that is driving by then to the center of mass of the wall and thus be added to the wrong cluster. The AHC method offers some advantages over the K-Means method

- Different distance measures for measuring the cluster distance are available.
- The maximal distance at which clusters can be merged can be specified.
- The number of clusters does not need to be known in advance.
- The distance matrix only has to be calculated once. In later merging steps the distance matrix can be reused to calculate the distance between the remaining clusters with

$$dist(i+j,k) = \alpha_i dist(i,k) + \alpha_j dist(j,k) + \beta dist(i,j) + \gamma |dist(i,k) - dist(j,k)|, \tag{2.14}$$

introduced by Lance and Williams [1967]. Several implementations of the AHC method are available. Commonly single-link AHC is implemented using the SLINK algorithm which has a runtime and memory complexity of $O(N^2)$ (Sibson [1973]). Although K-Means and AHC have the same upper bound for the runtime complexity, the AHC method often converges at lower costs then K-Means because of the recursive estimation of the distance between clusters. The resulting clusters of points are treated as hypothesis of dynamic objects and will be used for motion estimation in various ways throughout this work.

2.2.2 Map Patches as Local Motion Features

In this section, the extraction of the local map patches that are used for learning and predicting the speed as well as the direction of the dynamic objects are described. A map patch is small area of a difference or reflection map and is centered at a given point.

As mentioned before, the goal is to estimate motion from local features. There are two types of features we will use. Firstly, we will use the map patches itself as a feature for motion prediction. Secondly, we will apply a Gabor filter bank to the patches to reduce their dimension. The result is again used as a basis for predicting motion.

The map patches are obtained from the pointcloud data together with the maps constructed in Section 2.1.1 and 2.1.2. For each point in a cluster a patch from the corresponding maps is obtained. The exact algorithm is as follows

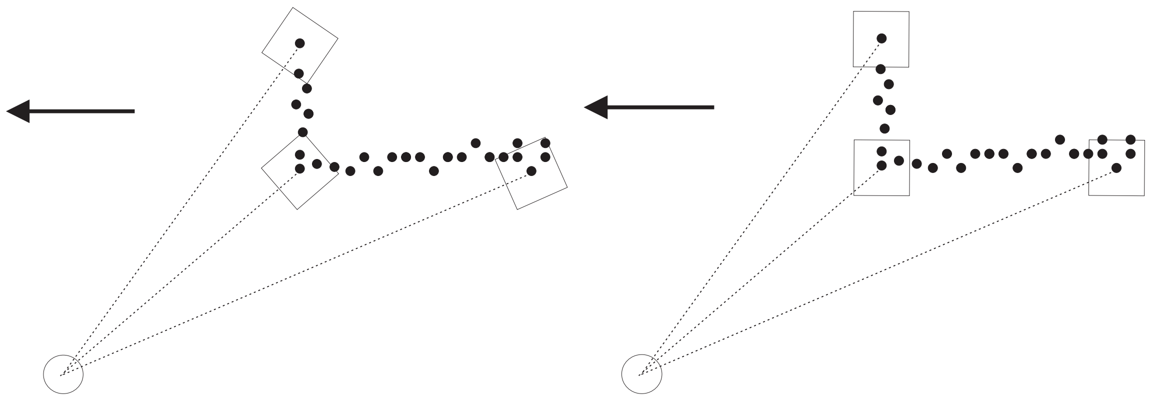
- Construct the difference map (see Section 2.1.1).
- Cluster the pointcloud using the AHC clustering technique described in Section 2.2.1.
- Extract a map patch for all points in a cluster out of the difference and the reflection map.

Figure 2.5 depicts the method of patch extraction. (For the sake of clearness only three patches are drawn. Actually, the map patches are constructed for each point in the cluster.) Different types of patches are extracted for predicting the speed and the heading. For heading estimation the patch is rotated into the direction from which the corresponding point is observed. Patches for speed estimation are rotated into the moving direction of the object. This already pose some requirements for the motion estimation algorithm. Before the estimation of the speed is possible the heading must be estimated.

It is not yet clear what size such a map patch shall have. Based on the idea that we are looking for local features from which motion can be inferred we pose two requirements on the patches. Firstly, the patch must be large enough to capture the motion cues. Secondly, the patch must be small enough to capture local motion features that are as independent of the structure of the underlying object as possible. If the patch is too large it might capture many properties of the actual underlying object and therefore does not generalize to other objects. Thirdly, we need to consider the rate at which the pointclouds are obtained and the possible speed of the objects we observe. For instance, for capturing the motion of a airplane, the frequency of the laser would be too low. Even if the laser would capture the plane the size of the patch would have to be huge. These parameters are examined closer and set in Chapter 3 where the learning problem and setup is described in more detail.

2.2.3 Extracting Features with Gabor Filters

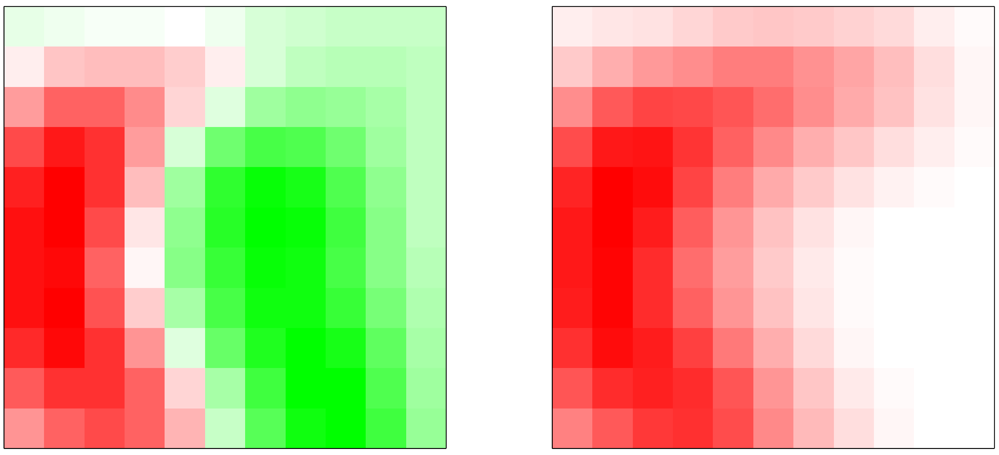
We consider two possibilities for estimating the direction of a moving object. One of these is the estimation using Gabor filters. This section describes how Gabor filters can be used to extract the direction given a difference map patch. We first describe Gabor filters in general and then explain how it is applied.



(a) Patches for angle estimation

(b) Patch for speed estimation

Figure 2.5: Generation of map patches that are used for angle and speed estimation. The arrow denotes the movement direction of the dynamic object, the circle denotes the observer and the dashed lines the observation direction.



(a) Difference map patch

(b) Reflection map patch

Figure 2.6: Example of a patch extracted from the difference map and the corresponding patch of the current reflection map.

Gabor Filter

Gabor filters are widely used in various research areas like face recognition (Cook *et al.* [2005]), fingerprint enhancement (Zhu *et al.* [2004]), texture classification (Sabri and Fieguth [2004]) and many more. The Gabor filter is a linear filter which is the result of the multiplication of a symmetric Gauss kernel and an oriented cosine function. It is defined as

$$g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x^{'2} + \gamma^2 y^{'2}}{2\sigma^2}\right) + \cos\left(2\pi\frac{x^{'}}{\lambda} + \psi\right), \tag{2.15}$$

where

$$x^{'} = x \cdot \cos(\theta) + y \cdot \sin(\theta), \tag{2.16}$$

and

$$y^{'} = x \cdot \sin(\theta) + y \cdot \cos(\theta). \tag{2.17}$$

In the formula above λ denotes the wavelength of the cosine factor, θ the orientation of the normal to the parallel stripes, ψ is the phase offset, σ is the sigma of the Gaussian envelope and γ the spatial aspect ratio, which basically specifies the ellipticity of the Gabor function. This high parameterization is one of the advantages of the Gabor filter and one of the reasons why Gabor filters are used to create filter banks and wavelets. Figure 2.7 depicts 16 Gabor filters with different rotation angles and a phase offset of $\pi/2$. The Gabor filter images are discretized on purpose to emphasize the visual similarity between the filters and the difference map patch depicted in Figure 2.6.

Application to Motion Prediction

Considering Figure 2.7, one can easily see that the Gabor filters look very similar to the sample patch shown in Figure 2.6. This visual similarity motivated the usage of the Gabor filter as a feature extractor. The almost obvious idea is that the Gabor filter best matching a difference map patch must result in a good estimate for its direction. We manually constructed a Gabor filter bank of 16 filters each of which have a rotation angle that is a multiple of $1/360$. The Gabor filter bank can be regarded basis vectors e_i of a vector space. We use the dot product $< p, e_i >$ of a difference map patch denoted by vector p with basis vector e_i of the the Gabor filter bank to project the patch into the Gabor space. The dot product is widely known as a measure for the similarity of two vectors. It implements an orthogonal projection of the patch onto the basis vectors and results in the best approximation of the patch in the Gabor space. The patch that is most similar to the specific Gabor filter results in the highest response. All parameters of the Gabor filter besides the rotation angle θ and the phase offset ψ were set by maximizing the response of the Gabor filter bank on a given test set. Figure 2.8 depicts an example of a patch and the result of the dot products. We can observe that some of the single filters yield a similar, although not the same result. That raises the question whether we can really estimate the angle just by using the maximum response filter. It is not obvious whether one of these responses is correct or whether a weighted average of them would results in a better estimation. Just taking the maximum dot product of the patch and the Gabor filters as a estimation of the heading does not seem feasible. Instead, we construct a feature vector of all dot products which is then used as

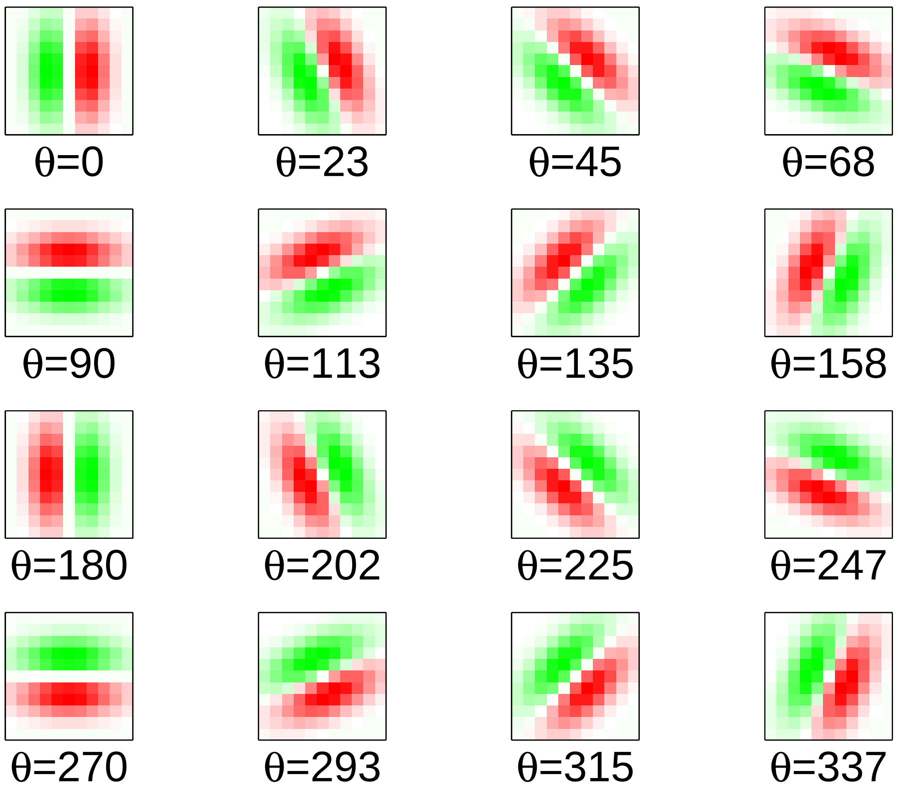
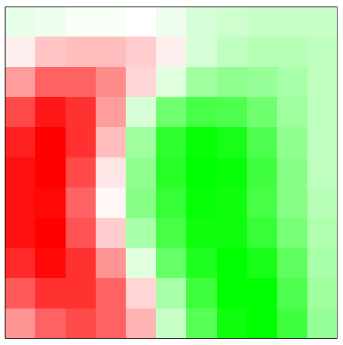
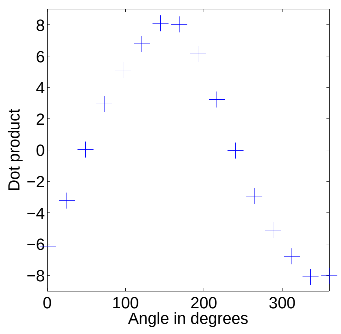


Figure 2.7: Gabor filter with different rotation angle θ and phase offset at $\pi/2$



(a) Patch from Difference Map



(b) Dot product of patch and filter bank.

Figure 2.8: Dot products of the patch from Figure 2.6a and the constructed Gabor filter bank.

input of a regression learner based on a Gaussian process. We expect that the learning algorithm achieves a better result by considering the entire feature vector. The regression problem and the learner are defined in the next chapter. In Chapter 4 we evaluate the estimation result using the Gabor filter results and the map patches as features.

3 Motion Perception and Estimation

The preceding chapter described the different data representations used and how motion features are extracted out of these respresentations. In the following, we focus on inferring motion from these extracted features (see Figure 1.2). In this chapter, we first propose a method for motion estimation that is based on Gaussian process regression. We argue that the estimation of motion for a set of dynamic objects can be formulated as a regression problem. We then describe how Gaussian process regression can be applied to our application domain and explain its advantages and disadvantages for motion estimation. Section 3.2 introduces two commonly used methods for motion estimation, namely the iterative closest point algorithm (ICP) and Kalman filtering.

3.1 Predicting Motion using Gaussian Processes

Our approach to motion estimation can detect and estimate the heading and the speed of a moving object out of range data. At each point in time t, the algorithm gets a pointcloud and the location of the sensor as input. Our proposed motion estimation algorithm, which is also depicted as a graph in Figure 1.2 is as follows:

1. Cluster the pointcloud at time t. Each cluster is taken as a dynamic object hypothesis C_i.
2. Compute the difference map at time t given the occupancy maps at time t and $t-1$ (see Section 2.1.2).
3. Predict the heading for each cluster C_i.
    a) For each point in the cluster C_i, extract a patch p_j for heading prediction using the occupancy map and the difference map (see Section 2.2.2).
    b) For each point in a cluster, predict the heading α_j based on the extracted map patches or their representation in the Gabor filter space.
    c) Merge the single heading predictions to one heading prediction α_i per cluster.
4. Predict the speed for each cluster C_i.
    a) For each point in a cluster, extract a patch p_j for the speed prediction using the occupancy map, the difference map and the predicted heading α_i. (see Section 2.2.2).
    b) For each point in a cluster, predict the speed v_j based on the extracted map patches.
    c) Merge the single predictions to one speed prediction per cluster.

This section concentrates on estimating motion given the features derived in Chapter 2. We proceed as follows. First, we formulate the estimation of motion as a regression problem. Second, we describe the process of selecting suitable samples for solving the regression problem. Third, Gaussian process regression is introduced in general and the extensions and steps needed to apply Gaussian processes to motion estimation are explained. Finally, we show how motion for a entire cluster can be obtained by merging the single predictions of each point.

3.1.1 Motion Estimation as a Regression Problem

As already mentioned in the beginning of the chapter we have to tackle two estimation tasks, estimating the heading α and the speed v of a dynamic object. In Chapter 2 we clustered the pointcloud and extracted the local environment for each measurement. Based on this local environment, we proposed two features, namely map patches and Gabor filter features. The goal is to use theses features for our estimation task.

Heading Estimation

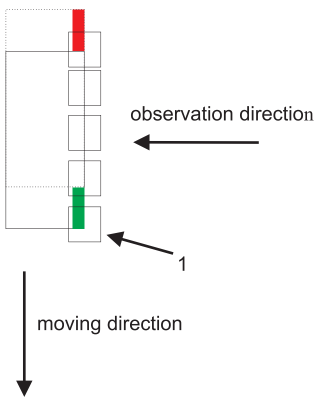
Estimating the heading of the moving objects is a challenging task. Predicting the heading directly would result in a discontinuity in the target function, at the function values $-\pi$ and π. That means that although the training samples are relatively close in input space the target value differs much on the output. Although the GP framework has not been introduced yet we should note that a non-continuous target function generally requires a non-stationary covariance function for achieving good prediction results. However, using non-stationary covariance functions results in problems like how to correctly estimate the noise at the discontinuity. We will also see in Section 3.1.2 that modeling the noise will be important for our application. It is advantageous to parameterize the target function such that the discontinuity vanishes. In our case, it is possible to split the estimation of the heading α with a non-continuous target function into the estimation of two continuous functions. If we use the sine and cosine representation of the heading, we avoid the discontinuity, because sine and cosine are continuous in their function values. Unfortunately, we require two estimation processes. Therefore we define the regression problem as estimating the functional dependencies

1. f_1 given $\mathcal{D} = \{x_i, \sin(\alpha_i)\}_1^n$,
2. f_2 given $\mathcal{D} = \{x_i, \cos(\alpha_i)\}_1^n$,

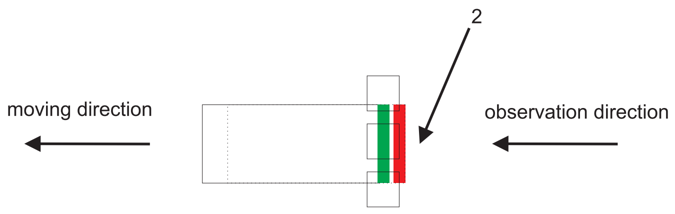
where $x \in R^D$ is a motion feature with the dimensionality D. This new parameterization has the advantage that it solves the problem of the discontinuity. The disadvantage is that the prediction now depends on two values and we are left with two regression tasks. If the prediction of either part is not correct then the entire angle prediction is not correct.

Speed Estimation

We define the speed estimation as a regression problem with $\mathcal{D} = \{x_i, v_i\}_1^n$ with $x \in \mathbb{R}^D$ is a map patch feature (see Section 2.2.2) and $v \in \mathbb{R}$ is the velocity of the object. The difference to heading estimation, is the rotation of the pattern used. This time the patch is not aligned to the observation direction but to direction of the estimated heading (see Section 2.2.2). Note that the speed prediction must be done after the heading estimation. The advantage of this approach



(a) Relative angle: 90 degrees



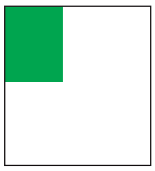
(b) Relative angle: 0 degrees

Figure 3.1: Car moving at different angles relative to the observation direction. The red (dark gray) and green (light gray) part denote the occupancy change on the difference map. The arrows denote the observation and moving direction of the dynamic object. The squares are sample patches. For an easier illustration, they are equally spaced.

is that learning the speed becomes independent of the moving direction. Estimating a speed function for every possible moving direction would increase the size of the needed training data tremendously. The disadvantage is that the prediction of the heading has to be fairly correct. If the patch is extracted given the wrong heading of the object it is likely that the speed estimation will also be wrong.

3.1.2 Selecting Samples for Learning

Having formulated the regression problem, we now describe how to select samples for learning. Consider Figure 3.1a which shows a car moving in a angle of 90 degrees relative to the observation direction. The car, represented by its bounding box, moves from the position depicted by the dashed line to the position depicted by the solid line. The red (dark gray) and green (light gray) areas mark the pattern that the car produces on the difference map. Compared to the full length of the observed car, only a very small part appears to be moving, namely the front and the back.


(a) Patch 1 from Figure 3.1a

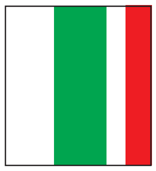

(b) Patch 2 from Figure 3.1b

Figure 3.2: Enlargement of two difference map patches of Figure 3.1.2

Only at these positions an occupancy change can be measured. Of course, the rest of the car is also moving but we are not able to detect it. The laser range measurements at these locations stay the same. Although Figure 3.1a represents the worst case scenario, we are facing the problem that there are only few patches in which motion information is encoded. The figure depicts some patches, denoted as equally spaced squares. In general the location of the patches is given by the measurements. Note that some patches do not capture motion information at all while others only contain information about a loss or a gain of occupancy. The patch marked with a one is enlarged in Figure 3.2a. One can easily see that this patch only contains information about a gain of occupancy which makes it harder to infer the direction. Figure 3.1b depicts a different situation. The observed car is moving into the same direction as the observer. Almost all observed parts of the car appear to be moving. In this situation, object motion can be captured easier by our motion features. Compare the two enlarged patches in Figure 3.1.2. It seems easier to infer the direction using the patch of Figure 3.2b, because it contains information about a gain and a loss of occupancy. However, situations where most map patches contain information about the motion of the object are rather uncommon. In our data only about 20 percent of the training samples contained motion information which also means that for about 80 percent of the training samples the motion prediction is not reliable.

This situation poses two problems. The first problem is that learning algorithms in general and Gaussian processes in particular are limited in the number of training samples they can use to infer the latent function. If a GP could learn from a large training set, with $\gg 1000$ samples it would not pose such a problem. Although most samples would not be suited for motion estimation, the number of suitable ones would be sufficient for estimating motion. Considering the dimensionality of data we are practically able to use a set of about 1000 samples for training a GP. Randomly selecting these training samples would lead to about 200 samples suitable for the estimation task which is not enough considering the high dimensionality of the data. Therefore the training samples can not be selected randomly and other methods for sample selection need to be tried. There are several methods for selecting samples valuable for training like selecting them according to their "predicted" information gain. Another common choice is cross validation which always leaves selects a subset of the training data for learning and uses the rest of the data for testing. This procedure is iteratively applied and the then the training set that resulted in the best prediction is used. However, cross-validation requires much time do to the iterative nature of the process.. We would rather rely on a method that can be applied once before learning. Therefore we chose to perform a weighted sampling. The training samples are assigned weights w in the interval $[0;1]$ defined as:

$$w = \frac{\sum_1^n occ(n)}{n} \tag{3.1}$$

with n being the sample corresponding to a map patch consisting of n cells. The samples are then drawn using importance sampling which simply selects the training samples with a probability according to their weight. This selection method has the distinct advantage that most training samples selected have a high weight but also samples with a low weight are drawn, although with a lower probability. Using some noisy samples for learning is also important in order to better estimate the noise function of the GP.

The second problem is that not all features of an observed object are equally good for estimating motion. We must decide which features are suitable for motion prediction and which are not. We approach this challenge using Gaussian processes (GP). One of the advantages of using GPs is that they naturally deal with noisy inputs and predict a full distribution, a mean of the function value and a variance for each input. As already mentioned in the beginning of the chapter we predict a motion for all features belonging to an observed object and combine them to an overall estimate. The combination of the single predictions is done weighting them using the predicted variance. The estimation of the variance is approached using heteroscadestic Gaussian process regression and is described in Section 3.1.3. The combination of the measurements given the predicted variance is described in Section 3.1.4.

3.1.3 Gaussian Process Regression

We have described the regression problem in general, as well as the process of obtaining well suited samples for learning. This section introduces the regression framework used as well as its advantages and disadvantages for our application domain. The goal is to robustly predict the motion of each observation. Thereby we are facing several problems. First, only few samples can be used for learning. Second, the features selected for motion estimation are high dimensional (e.g. the representation of a patch in the Gabor space is 16 dimensional,). Third, we must correctly estimate a variance for each prediction based on its input location. The idea is, that if several similar patches represent a contradictory motion, the noise at this input position is high. Accordingly, the predicted variance of an input at this location should be high. If the variance is predicted correctly we can merge the single predictions according to their predicted variance to obtain a combined prediction (see Section 3.1.4).

We start introducing the GP regression framework in general. We then describe two extensions of the framework that enable us to use a smaller sample set for training and to estimate a noise function of the input.

Given a data set $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$ with $x_i \in \mathbb{R}^D$ and noisy targets $y_i \in \mathbb{R}$ regression aims to infer the functional dependency

$$y_i = f(x_i) + \varepsilon_i, \tag{3.2}$$

where ε_i are samples drawn independently from a additive noise process with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. In the following, we denote the set of inputs with $X = \{x_i\}_1^n$ and the set of targets as $y = \{y_i\}_1^n$. Typically, the function space is not constrained enough by Equation 3.2 and additional constraints about the form of the function are needed for obtaining a unique solution of the regression problem that explains the data well. The set of possible functions can be restricted by requiring a certain function model, for instance the linear model

$$f(x) = w_0 + w_1 x + \ldots + w_n x^n \tag{3.3}$$

where $n \in \mathbb{N}$, and $w = w_1 \ldots w_n$ is a set of parameters or weights that have to be learned. But how do we determine the parameter values of the function f that yield the optimal solution?

Classical approaches use different kinds of estimators to determine the parameters. Commonly the least-squares estimator

$$S = \sum_{1}^{n} \left(f(x_i; \hat{w}) - y_i \right)^2 , \tag{3.4}$$

is used which selects the parameter set $\hat{w}$ that minimizes the sum of squared errors S and results in a point estimate of the parameters..

Bayesian approaches generally model a posterior over the function f given the data with

$$p(f|X, y) = \frac{p(y|X, f)p(f)}{p(y|X)}, \tag{3.5}$$

where $p(y|X, f)$ is the likelihood of the targets given the data and $p(f)$ the prior over functions. The marginal likelihood $p(y|X)$ is independent of the parameters and given by

$$p(y|X) = \int p(y|f, X)p(f)\, df. \tag{3.6}$$

In case of a parametric model, the prior is given implicitly in the form of the function model and its parameters, for instance the linear model stated above. Having obtained the posterior (Equation 3.5), we are able to make a prediction f_* at input location x_* integrating over all possible functions

$$p(f_*|x_*, X, y) = \int p(f_*|x_*, f)p(f|X, y)\, df, \tag{3.7}$$

This contrasts the classical approach that results in a point estimate, i.e., a single function with a specific set of parameters estimated. Modeling of a posterior distribution over functions to be inferred instead of considering just a point estimate of the parameters as in the classical non-Bayesian case is the distinct advantage of the Bayesian approach to regression. However, for most applications the integrals used are intractable.

There are two common solutions to tackle this problem:

1. Sampling the posterior $p(f|X, y)$ directly using the Markov Chain Monte Carlo (MCMC, Geyer [1992]) method or another sampling technique.

2. Approximating the marginal likelihood

$$p(y|X) \cong p(y|f_{ML}, X)p(f_{ML}) \tag{3.8}$$

with the MaximumLikelihood function f_{ML} obtained by maximizing the objective function

$$log[p(y|X)]. \tag{3.9}$$

Maximizing the log of a function is equivalent to maximizing the function itself as the log is a monotonically increasing function of its input. In case of the linear model, the function that maximizes the likelihood is identical to the function found using the least squares estimator. There are several implementations possible for performing the maximization step. Often the objective function is maximized using gradient descent with respect to the parameters, if the gradient can be obtained.

The disadvantage of using a parametric model is that the assumed prior $p(f)$ must be consistent with the data generating process in order to achieve good prediction results. So we must perform a good guess about the underlying form of the function. For instance, using the linear model, the more parameters we choose the higher is the risk the maximum likelihood function is overfitting the data. The principle of Occam's Razor, i.e., simple models explaining the data should be preferred over complicated ones helps choosing the number of parameters. However, we must still have a good intuition about the form of the data generating process. In many cases the form of the function is not known and hard to understand intuitively, for instance when given high dimensional data that is not possible to visualize.

We address this problem by using Gaussian process (GP) regression, that was first studied in machine learning contexts by Williams and Rasmussen [1996]. GPs have recently become popular for modeling various machine learning problems because they have some distinct advantages such as

- naturally dealing with noisy measurements,
- representing distributions non-parametrically only in terms of the training data,
- naturally integrating into the Bayesian framework of model selection and density prediction.

The key idea is the assumption that all function values of f are jointly Gaussian distributed and therefore we can place a prior directly over the space of functions with

$$p(f|X, y, \theta) = \mathcal{N}(0, K), \tag{3.10}$$

without specifying a parametric model of the function f itself. Here, the mean function is set to 0 which is a common choice and not a limitation. $K \in \mathbb{R}^{n \times n}$ denotes the covariance matrix which must be squared and positive semi-definite. The entries $K(i, j)$ of the covariance matrix are given by the covariance or kernel function $k(x_i, x_j)$. A commonly used kernel function which is also used in this work is the squared exponential (SE) function

$$k(x_i, x_j) = \sigma_f^2 \cdot exp\left(-\frac{1}{2}\sum_{d=1}^{n}\frac{(x_{i,d} - x_{j,d})}{l_d}\right), \tag{3.11}$$

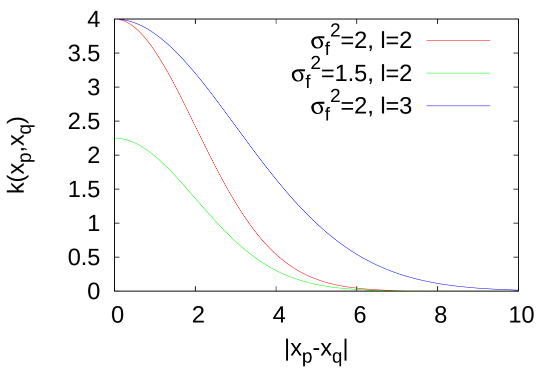


Figure 3.3: Stationary squared exponential covariance function: The function value $k(x_p, x_q)$ only depends on the distance between the input values x_p, x_q and the hyperparameters $\theta = (\sigma_f^2$, $l)$

.

where the parameters $\theta = \{\sigma_f^2, l_d\}$ are called hyperparameters. σ_f denotes the signal variance (amplitude) and l_d the lengthscale (or smoothness) of the function. The parameter d denotes the input dimension of the sample. Learning a Gaussian process means learning these hyperparameters from training data. Specifying and learning one lengthscale parameter per dimension is also known as automated relevance determination (MacKay [1998]). Figure 3.3 depicts the squared exponential function for one input dimension with different values for the lengthscale and the signal variance. The covariance of x_p and x_q depends only on the Euclidean distance between the two input locations and is therefore called stationary. The covariance function provides previously known knowledge about the function to be predicted, just as a parametric model specifies assumptions on the form of the function. Figure 3.4a depicts a set of five functions drawn from this prior and one can easily see the type of prior knowledge introduced by the covariance function. As all functions seem to be smooth and change only at a constant rate. Assuming independent, identically Gaussian-distributed noise terms, $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$ the likelihood is given by

$$p(y|f, X, \Theta) = \mathcal{N}(f, \sigma_n^2 I). \tag{3.12}$$

with $\Theta = \sigma_f, \sigma_n, l_d$. Because the likelihood and the prior are Gaussian we obtain the marginal likelihood $p(y|X, \Theta)$ in closed form with

$$p(y|X, \Theta) = \mathcal{N}(0, K + \sigma_n^2 I). \tag{3.13}$$

We note that $p(y|X, \Theta)$ is also Gaussian and as a result of the marginalization step the noise terms are simply added to the covariance. Using equation 3.5 we obtain the posterior distribution of $p(f|x, y, X, \Theta)$ that combines the prior with the data using the likelihood. For the posterior distribution

$$p(f|x, y, X, \Theta) = \mathcal{N}(\mu, \sigma^2), \tag{3.14}$$

that we condition on the training inputs D and the test inputs x (Eq. 3.7) we obtain the mean

$$\mu = K_*^T (K + \sigma_n^2 I)^{-1} y, \tag{3.15}$$

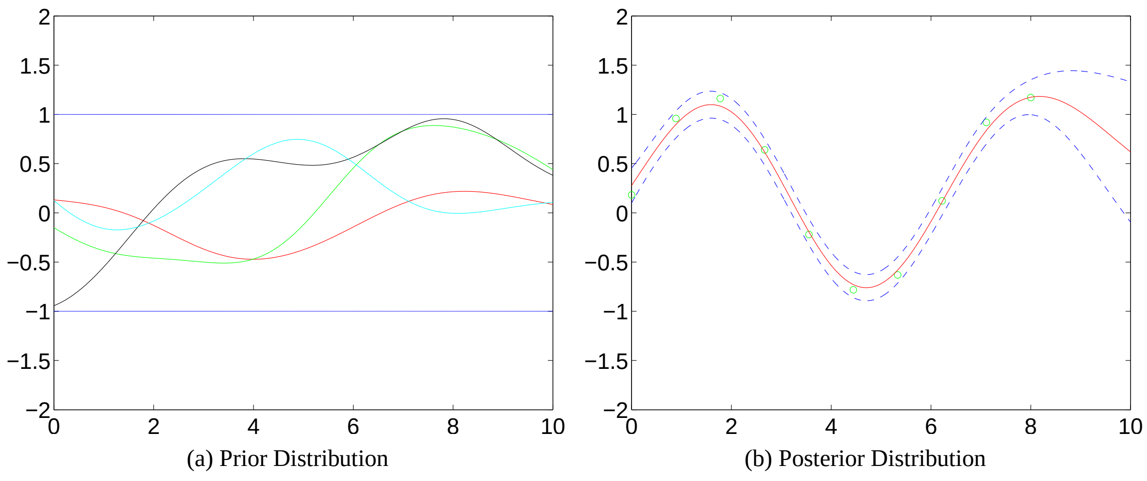


Figure 3.4: Random Functions drawn taken form the prior distribution 3.4a and the posterior distribution 3.4b which restricts the number of functions to those explaining the data well.

and the variance

$$\sigma^2 = K_{**} - K_*^T(K + \sigma_n^2 I)^{-1} K_*. \tag{3.16}$$

where K_{**} denotes the covariance between the vector of test inputs itself and K_* denotes the covariance between the vector of test inputs and the vector of training inputs. The mean of the posterior is a linear combination of the targets using the factor $K_*^T(K + \sigma_n^2 I)^{-1}$ as “weights”. Figure 3.4b depicts some functions drawn randomly from the posterior distribution. While for some applications it might be useful to obtain the full posterior distribution, we are usually only interested in making predictions for distinct inputs. In the above calculations we assumed the hyperparameters Θ to be given. To make predictions in the Bayesian setting we have to compute the marginal likelihood

$$p(y|X) = \int p(y|X,\Theta)p(\Theta)d\Theta. \tag{3.17}$$

This integral can typically not be solved analytically. This problem is usually solved by applying the MaximumLikelihood principle and approximating the integral by maximizing the objective function

$$log[p(y|X)] = \underbrace{-\frac{1}{2}y^T(K + \sigma_n^2 I)^{-1}}_{\text{data fit}} \underbrace{-\frac{1}{2}log|K + \sigma_n^2 I|}_{\text{complexity penalty}} \underbrace{-\frac{1}{2}log 2\pi}_{\text{constant}}. \tag{3.18}$$

Typically, the maximization step is implemented as minimizing the negative log-likelihood using gradient descent. The objective function balances the data fit and the complexity by using the second term which penalizes the complexity. The result of the maximization are the maximum likely hyperparameters Θ_{ML}. As stated above for linear Bayesian regression, the marginal likelihood $p(y|X)$ could also have been approximated by sampling it directly. The computational complexity of th hyperparameter estimation is dominated by the inversion of the factor

$(K + \sigma_n^2 I)$ and therefore in $O(N^3)$. Because the inversion has to be calculated for every hyperparameter set tried an efficient gradient descent method like the method of conjugate gradients (Fletcher and Reeves [1964]) is needed for the estimation of the parameters.

Given the maximum likely hyperparameters Θ_{ML}, we can then conclude for the predictive point estimate

$$p(y_*|x_*, D, \Theta_{ML}) = \mathcal{N}(\mu, \sigma^2), \tag{3.19}$$

that is conditioned on the training inputs D and the test input x_* we obtain the mean

$$\mu = k_*^T (K + \sigma_n^2 I)^{-1} y, \tag{3.20}$$

and the predictive variance

$$\sigma^2 = k(x_*, x_*) - k_*^T (K + \sigma_n^2 I)^{-1} k_*, \tag{3.21}$$

(see Rasmussen and Williams [2005]). Here, $k_* \in \mathbb{R}^n$ with $k_* = k(x_*, x_i)$. Notice that the runtime of the prediction is also governed by the factor $(K + \sigma_n^2 I)^{-1}$ and therefore in $O(N^3)$. Fortunately, this factor has to be calculated only once for all predictions to be made and if it is known the runtime decreases to $O(N)$ for the predicting the mean and $O(N^2)$ for predicting the variance.

Hence, GPs are a framework for regression that does not require a parametric model of the data generating process itself but models a posterior over functions only parameterized by training data and the hyperparameters of the covariance function. GPs have the disadvantage that training the GP requires time $O(N^3)$ where N is the number of training inputs. As mentioned in Chapter 2.1, one of the features used inferring motion are map patches. Although not specified yet, these patches are high-dimensional ($D > 50$). Learning a GP using such a dataset seems infeasible for the standard GP model introduced in this section. The reason is that using the squared-exponential covariance function we need to learn one parameter, the length-scale, for each dimension in addition to the noise and signal variance. Depending on the gradient descent algorithm used, the number of iterations needed to maximize the likelyhood of the objective function becomes very large. In practice, the inversion of the covariance matrix takes too long given a suitable sample size (> 1000). The next section introduces a sparse approximation based on pseudo-inputs that allows for using high-dimensional inputs.

Sparse GPR using Pseudo-Inputs

Gaussian processes are intractable for large data sets or data sets with a large number of dimensions because training a full Gaussian process model requires $O(N^3)$ time with N being the number of training samples. The higher the number of input dimensions the more parameters need to be learned. There are several methods addressing this problem with sparse approximations. A commonly applied method is selecting a set of inducing inputs. That means that we reduce the number of training samples by selecting a subset which is especially well suited for learning. Strategies for selecting inducing inputs include, e.g., the differential entropy score by Lawrence *et al.* [2002] that favors inputs largely reducing the predictive variance. A good evaluation of the different methods and further proposals for sparse approximation can be found in

Candela and Rasmussen [2005]. Snelson and Ghahramani [2006a] also introduced a sparse approximation of a full GP based on inducing inputs. They relax the condition that the inducing inputs must be a subset of the training data and consider the inducing inputs as additional parameters to be learned. These inputs are referred to as pseudo-inputs. In this thesis, we apply this approach to motion estimation. Being able to learn the pseudo-inputs, we expect that only a small number of them is needed. We introduce the sparse GP (SPGP) by first deriving its Gaussian process prior and comparing it to the standard Gaussian process prior. Second, we show how the pseudo-inputs can be learned from data and single point predictions can be obtained.

The central idea of this sparse GP is to approximate a full GP learned on the training set $\mathcal{D}$ of size N by a GP learned on a smaller set of pseudo data $\bar{\mathcal{D}}$ of size M with $M \ll N$. The GP learned on the pseudo dataset is taken as a special parameterization of a standard GP. The pseudo targets are not real targets and therefore modeled noise free. They are located in the same space as the real targets and are actually equivalent to the latent function values f. Given a set $\bar{\mathcal{D}}$ with pseudo targets $\bar{f} = \{\bar{f}_m\}_1^M$ and pseudo-inputs $\bar{X} = \{\bar{x}_m\}_1^M$ Snelson and Ghahramani [2006a] place a Gaussian prior

$$p(\bar{f}|\bar{X}) = \mathcal{N}(0, K_M), \tag{3.22}$$

on the pseudo-targets and assume them similarly distributed to the real data. K_M denotes the covariance matrix computed from the pseudo-inputs. The covariance function used in this work is the squared-exponential covariance function with automatic relevance determination, already introduced in Section 3.1.3. Having placed an Gaussian prior on the pseudo-inputs we condition the noise free latent function f on the pseudo-data and the real inputs. We approximate

$$p(f|\bar{f}) \approx \prod_{1}^{N} p(f_n|\bar{f}), \tag{3.23}$$

to obtain

$$p(f|\bar{f}) = \mathcal{N}(K_{NM}K_M^{-1}\bar{f}, K_N - K_{NM}K_M^{-1}K_{MN}). \tag{3.24}$$

Having obtained the conditional, we compute the prior $p(f)$ of latent function by marginalizing over the pseudo targets

$$\begin{aligned} p(f) &= \int p(f|\bar{f})p(\bar{f})d\bar{f} & (3.25)\\ &= \mathcal{N}(0, K_N^{SPGP}), & (3.26) \end{aligned}$$

with

$$\begin{aligned} K_N^{SPGP} &= K_{NM}K_M^{-1}K_{MN} + \Delta, & (3.27)\\ \Delta &= diag(K_N - K_{NM}K_M^{-1}K_{MN}). & (3.28) \end{aligned}$$

Intuitively, we can consider the SPGP being standard GP with a special covariance function, parameterized with the pseudo input locations $\bar{X}$:

$$\underbrace{\mathcal{N}(0, K_N)}_{\text{GP prior}} \approx p(f) = \underbrace{\mathcal{N}(0, K_N^{SPGP})}_{\text{SPGP prior}}. \tag{3.29}$$

Note, that although the underlying covariance function $k(x_n, x_{n'})$ is stationary, the covariance of the SPGP is not stationary. The reason is that the covariance of the SPGP does not only depend on the distance between x_n and $x_{n'}$ but also on the position of the pseudo-inputs.

Given the covariance function $K^{SPGP}(x_n, x_{n'})$ the marginal likelihood can be obtained similarly to the standard GP approach

$$p(y|X, \bar{X}, \Theta) = \mathcal{N}(0, K_N^{SPGP} + \sigma^2 I). \tag{3.30}$$

The second main idea of the SPGP is to view the pseudo-inputs $\bar{X}$ as additional parameters to be optimized. Thus we find the set of pseudo-inputs by jointly maximizing the marginal likelihood with respect to $\bar{X}$ and Θ. The exact gradients depend largely on the covariance function $k(x_n, x_{n'})$ used. Any covariance function can be used as long as it is differentiable with respect to the pseudo-inputs which otherwise cannot be learned.

Similarly to the standard GP, the predictive point estimate is obtained conditioning on the test input x_*, the data $\mathcal{D}$, the hyperparameters Θ and the pseudo-inputs $\bar{X}$

$$p(y_*|x_*, \mathcal{D}, \bar{X}_{ML}, \Theta_{ML}) = \mathcal{N}(\mu_*, \sigma_*), \tag{3.31}$$

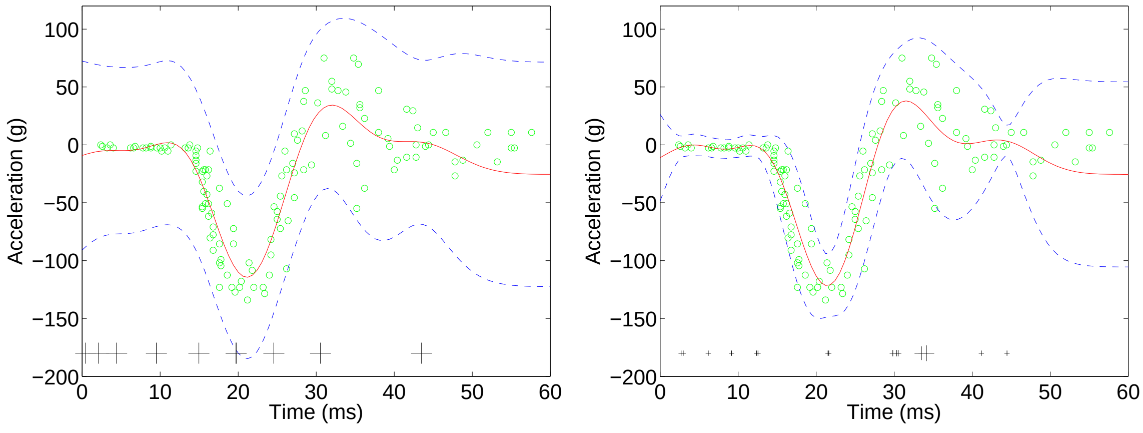
with

$$\mu_* = k_*^\top Q_M^{-1} K_{MN} (\Lambda + \sigma^2 I)^{-1} y, \tag{3.32}$$

$$\sigma_*^2 = K_{**} - k_*^\top (K_M^{-1} - Q_M^{-1}) k_* + \sigma^2, \tag{3.33}$$

with $Q_M = K_M + K_{MN}(\Delta + \sigma^2 I)^{-1} K_{NM}$. We annoted some of the conditioning variables with the subscript ML to emphasize the fact that the predictive point estimate is conditioned on the maximum likely parameters learned.

The computational cost of the SPGP are dominated by the calculation of the Q_M. Because Δ is diagonal the inversion of the factor $(\Delta + \sigma^2 I)^{-1}$ is in $O(N)$. Hence, the matrix multiplications are in Q_M are in $O(M^2 N)$. This is one of the advantages of this approach. The training costs are reduced to $O(M^2 N)$ and the cost for each test case are reduced to $O(M^2)$ for predicting the variance and $O(M)$ for predicting the mean. Additionally, the pseudo-inputs are not static and are shifted to input locations which improve the GP prediction as they are not limited to be a subset of the training set. An example of a SPGP regression is depicted in Figure 3.5a. Note that the crosses specify the location of the pseudo-inputs. Another advantage of the SPGP is estimating the pseudo-inputs and the gradient smoothly at the same time. Most other approaches using inducing inputs require an interweaving of the two which leads to a non-smooth gradient estimation as the gradient changes sharply with every new parameter set. The disadvantage of this method, however, is the fact that the number of hyperparameters to be learned increases to $O(MN + \Theta)$ which complicates the gradient calculation and introduces more local minima into the marginal likelihood. With the SPGP we have introduced a sparse approximation of the full GP that largely



(a) Sparse GP with pseudo-inputs (b) Sparse GP with a noise parameter for each pseudo-input

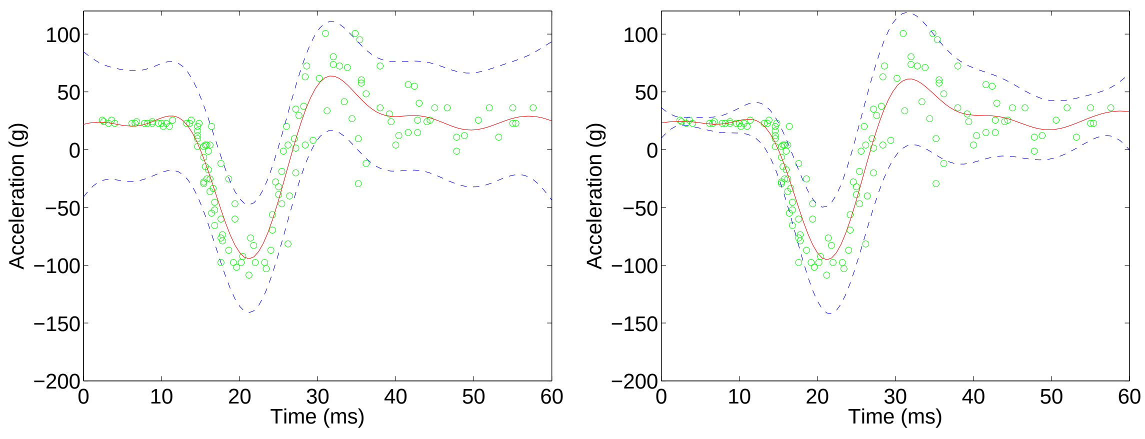
Figure 3.5: GP prediction using the Sparse GP with pseudo-inputs. The crosses denote the location of the pseudo-inputs. Figure 3.5a corresponds to the case where no input-dependent variance is assumed for the pseudo-inputs. Figure is shows the prediction of the Sparse GP modeling a noise variance for each pseudo-input independently. Large crosses denote a high noise for the corresponding pseudo-input while small crosses denote a low variance.

reduces the computational complexity. As already mentioned, our approach depends on the correctly estimating the noise function (see Section 3.1.2). The next section shows that the SPGP already models some kind of input-dependent noise and shows how the SPGP can be extended to improve the noise estimate.

Heteroscedastic Gaussian Process Regression

The standard GP only models a constant noise rate but in practice many problems require the exact modeling of local noise. For our application, the correct prediction of local noise is also important because of the challenging distribution of the training data, already mentioned in Section 3.1.2. The goal of this section is to show that the SPGP already models some kinds of local noise. In addition, we present an approach by Snelson and Ghahramani that extends the SPGP model to improve the noise estimate.

GPs regard all training inputs as noisy random variables drawn from a single Gaussian distribution. We assumed identically distributed Gaussian noise for all training inputs. If this assumption is violated for a set of random variables, this data set is called heteroscedastic. The effect of heteroscedastic GP modeling is depicted in Figure 3.1.3 which shows a GP based on one-dimensional inputs only. Approaches extending the standard GP model to heteroscedasity were introduced Goldberg *et al.* [1998] and Kersting *et al.* [2007]. These approaches estimate the noise as function of the input and place a separate prior on the noise function. Kersting *et al.* [2007] use a second GP to model the variance function. A third GP then estimates the mean given the learned noise function of the second GP. If the process has not converged, it is restarted with the combined used for the mean prediction. The advantage of this approach is that it can directly be



(a) Standard GP (b) Heteroscedastic GP

Figure 3.6: Result of using standard GP regression on a sample set 3.6a and heteroscadestic regression 3.6b.

applied to a large variety of GP models without changing the GP model or its covariance function. The sparse GP with pseudo-inputs by Snelson and Ghahramani already models some kinds of input-dependent noise. Considering Figure 3.5b where the location of the pseudo-inputs is denoted by a cross below the graph, we can observe that the sparse GP shifts the pseudo-inputs away from noise which results in an increased noise at locations the pseudo-inputs are not located. This approach is extended in Snelson and Ghahramani [2006b]. The main idea is to model and learn a noise parameter for each pseudo-input. Although this is not intuitively clear right away because the pseudo-inputs are not real inputs and therefore not drawn from a noisy random process, this method allows the pseudo-inputs to be spread over the entire input space. The result is a better model of the noise and the mean. The covariance matrix of the pseudo-inputs is adapted to

$$K_M = K_M + \sigma_h I \tag{3.34}$$

The noise parameters are now integrated as extra parameters into the maximization of the marginal likelihood of the SPGP. The marginal likelihood is then also maximized with respect to these parameters which of course has the disadvantage that more parameters have to be learned. Figure 3.5b depicts the result of modeling a noise parameter for each pseudo-input. The crosses in the figure denote the location of the pseudo-inputs while the size of the crosses denotes the learned noise. The larger the cross, the larger the noise is. One can easily see two properties of this Gaussian process. Firstly, the noise is estimated more correctly. Secondly, learning an individual noise parameter for each pseudo-input the GP seems prone to overfitting. The noise decreases very quickly in the vicinity of pseudo-inputs that are assigned a low noise and increases quickly in the absence of pseudo-inputs. Therefore, care must be taken when learning the GP to assure the noise function is learned correctly.

3.1.4 Merging the Predictions

As mentioned before, our approach first clusters the pointcloud and obtains a single prediction for each point in a segment (cluster). Each cluster corresponds to a possibly moving object. To compute a motion prediction for a segment we must merge the single point predictions obtained. We use Gaussian processes because they predict a full Gaussian distribution and not only a point estimate. In last section we showed how the noise function can be estimated using the SPGP and the prediction of the variance can be improved. Assuming the variance to be predicted correctly, the single GP predictions are merged to one prediction for the entire cluster. We use two strategies to merge the predictions.

First, in order to obtain a prediction for an entire set of estimations we multiply the single prediction with

$$\mathcal{N}(\mu,\sigma^2) \propto \mathcal{N}_1(\mu_1,\sigma_1^2)\cdot \ldots \cdot \mathcal{N}_i(\mu_i,\sigma_i^2)\cdot \ldots \cdot \mathcal{N}_n(\mu_n,\sigma_n^2), \tag{3.35}$$

with

$$\sigma^2 = \left(\sum_1^n \sigma_i^{-2}\right)^{-1}, \tag{3.36}$$

and

$$\mu = \sum_1^n \sigma^2 \sigma_i^{-2}\mu_i. \tag{3.37}$$

The product is not associative and all Gaussian must be multiplied in the same step. Intuitively, the result of this product is a voting of the predictions weighted by their variance. The variance of the resulting Gaussian distribution is smaller then the variance of the single predictions.

Second, we fit a Gaussian distribution to the predicted variances. Hence, we calculate the mean of the predicted variances and their variance. We then select only those measurements which have a small predicted variance compared to the entire set of predictions. A small variance means a variance which is further away from the mean then one standard deviation. Those predictions are then merged by voting using the strategy described before.

Merging the predictions is the last part of our motion estimation approach. In the following, we describe two alternative methods for motion estimation. We use these methods in Chapter 4 for comparing our approach with.

3.2 Alternative Approaches to Motion Estimation

In the last section we proposed a method for motion estimation based on Gaussian process regression. This section describes two alternative approaches used in the literature, namely the iterative closest point algorithm (ICP) and the Kalman filter using the constant velocity motion model. We briefly explain these approaches and explain their application to motion estimation. In Chapter 4 we compare them to our proposed method.

3.2.1 Predicting Motion using the Iterative Closest Point Algorithm

ICP is the dominant method for registrating 3D scans (Besl and McKay [1992]). In robotics, ICP is often used as scan matcher to improve the pose of the robot (Lu and Milios [1997]). The general idea of ICP is given two point sets we like to find the transformation (translation t and rotation R) such that they match best. In Chapter 2.1 we presented the available range data and the method for clustering the scan. Each segment obtained by clustering contains as set of 3D points belonging to this segment. In this thesis we use the ICP algorithm to find the best corresponding segment in the next scan by matching the points belonging to the segments. We use the obtained transformation to predict the movement of the segment. Thus the ICP method can be understood as another motion-based method (see Chapter 1) for estimating the movement of dynamic objects.

Iterative Closest Point

The problem of ICP is defined as follows. Given two point sets $X = \{x_i\}_1^N$ and $Y = \{y_i\}_1^N$ we want to find the rigid-body transform (translation t and rotation R) that minimizes the sum of squared errors

$$E(t, R) = \frac{1}{N_c} \sum_{i=1}^{N_c} ||x_i - Ry_i - t||^2. \tag{3.38}$$

In this equation N_c denotes the number of corresponding points. These are points for which a correspondence in both datasets was found. Thus if the correspondence of the points were known the calculation of the transformation is easy and can be done in closed form. However, in most application scenarios this correspondence is not known. Since the ICP algorithm has been introduced many improvements and additions have been made. A nice overview of the different methods can be found in Rusinkiewicz and Levoy [2001]. We will introduce a general state of the art method to solve ICP which is also used in this thesis.

The general ICP algorithm as introduced by Besl and McKay [1992] is specified as follows

1. Associate each point of X with a point of Y.
2. Compute the transform that minimizes the mean squared error between the associated points (see equation 3.38).
3. Apply the found transform to X and update the mean squared error.
4. Iterate the above steps until a convergence criteria is met, e.g., the mean squared error does not decrease anymore.

We implement the data association performed in step 1 using a nearest-neighbor search based on the Euclidean distance. Having obtained the data association, we calculate the transform in closed form using the SVD based method by Arun *et al.* [1987]. We construct matrix $A = \sum_1^N x_i y_i^T$ and perform the singular value decomposition

$$A = U \Sigma V^*. \tag{3.39}$$

Arun *et al.* [1987] show that the mean squared error (see equation 3.38) is minimized and the solution is unique if

$$R = UV, \tag{3.40}$$

$$t = \mu_X - R\mu_Y. \tag{3.41}$$

In this equation μ_X an μ_y denote the mean of the data sets X and Y. Figure 1.2 in the introduction depicts the application of ICP to the application of motion estimation. At first we cluster the pointcloud and use each cluster as a dynamic object hypothesis. In the algorithm described above we implicitely assumed the association of the different clusters to be given which is a strong assumption. Actually, solving this data association problem is one of the major problems in many application domains. There are several solutions to this problem possible. For instance, one could use nearest-neighbor data association, associating those cluster from subsequent scans whose center of mass is closest. Another type of data association already mentioned in Section 1.1 is to fit a bounding box to the segments and associate those segments whose bounding box overlaps. However, for evaluating this method (see Chapter 4), we provide the data association, disregarding this problem for now.

3.2.2 Predicting Motion using the Constant Velocity Motion Model

In addition to ICP, the second method we use for our approach with is a Kalman filter based on the constant velocity motion model. The key idea underlying this method is to cluster the scan points, calculate the center of mass of each cluster and use the center of mass as a reference point for tracking. This method is a feature-based approach for predicting motion, as already described in the introduction.

Kalman Filter

The Kalman filter is based on the Bayes filter equation which is given by

$$\underbrace{p(x_t|z_{1:t}, u_{1:t})}_{\text{current belief}} = \eta \underbrace{p(z_t|x_t)}_{\text{sensor model}} \int \underbrace{p(x_t|x_{t-1}, u_t)}_{\text{motion model}} \underbrace{p(x_{t-1}|z_{1:t-1}, u_{1:t-1})}_{\text{previous belief}} dx_{t-1}. \tag{3.42}$$

with x_t represents the estimated state of an object x, z_t the measurements and u_t the transitions (in our case motion) performed at time t. A derivation of the Bayes filter equation is given in Thrun *et al.* [2005]. Intuitively, the Bayes filter means that the current belief about the state of an object is given as the last known state of the object applied to the beliefed motion and corrected by new measurements. The Kalman filter implements the belief about the state of an object and the sensor model as Gaussian. If the motion model is implemented as a linear transition model we stay in the Gaussian world and therefore the motion model stays a Gaussian. In this case the integral can be solved in closed form which is one of the characteristic advantages of this filter.

State Representation We represent the state x at time t of an dynamic object as

$$x_t = (x, y, v_x, v_y), \tag{3.43}$$

with x, y being the position of the object and v_x, v_y the corresponding velocity of the object in x respectively y direction.

Motion Model When observing objects in the environment we have no prior information about there movement. In order to still be able to make predictions, we therefore have to make assumptions about there behavior. It seems reasonable to assume that the speed of observed dynamic objects stays the same in this interval. Applying this assumption the *constant velocity motion model*

$$x_t = x_{t-1} + \begin{pmatrix} v_{x-1} \\ v_{y-1} \\ 0 \\ 0 \end{pmatrix} \Delta t + \varepsilon_z \tag{3.44}$$

is used to model the state transitions with $\varepsilon_z \sim \mathcal{N}(0, \sigma^2)$. The Gaussian noise is added in order to account for small changes in the speed of the object. The advantage of using the constant velocity motion model is its smoothing effect. It makes it less prone to errors, if single wrong observations are obtained. The disadvantage, however, is that if the assumption that an dynamic object keeps its speed is violated, this motion model might result in an erroneous prediction.

Sensor Model The sensor model in general corrects the prediction by the measurements taken. The measurements in this case are the centers of mass computed from the clusters of the current scan which is described in more detail in Chapter 2.1. The sensor model is then given by

$$z_t = x_t + \varepsilon_z, \tag{3.45}$$

where ε_z describes the noise in the measurement which is also assumed Gaussian distributed with $\varepsilon_z \sim \mathcal{N}(0, \sigma_z^2)$. The exact value of σ_z^2 is chosen considering the maximum error of the estimated center of mass.

As for the ICP algorithm presented before a matching of corresponding segments is needed for the application of the Kalman filter. The implementations possible for realizing this data association are mainly the same as for the ICP approach, e.g., using the nearest-neighbor approach.

In this chapter, we proposed a novel method for estimating motion based on local features only. Using Gaussian process regression we compute a motion for each measurement. Having clustered the pointcloud we are able to compute a combined prediction for each cluster merging all single predictions belonging to this cluster. In the next chapter, we evaluate our approach for predicting motion and compare it to the ICP and Kalman filtering method.

4 Evaluation

In the previous chapter, we proposed a Gaussian process approach to motion estimation. We also presented two alternative, commonly used approaches, the ICP algorithm and the Kalman filter. In this chapter we evaluate our proposed method and compare it to those alternative approaches. The goal is to answer the following questions:

- How is the GP prediction influenced by the map prior?
- Which of the two proposed features is better for estimating the heading? The map patch features or the Gabor filter features?
- How good is the estimate of the noise function learned by the SPGP and the SPGP-HET?
- Can the GP based approach to motion estimation robustly predict the angle and speed of the dynamic objects?
- How accurate is the prediction compared to using the state of the art motion estimation methods ICP or the Kalman filter?

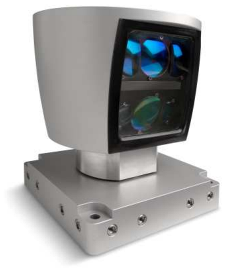
The chapter is structured as follows. First, we introduce the general setup that includes a description of the available range data and the specification of the method used to obtain the training and test sets. Second, we specify the methods and measures used for the evaluation. Third, we evaluate the GP learning based on the different features proposed in Chapter 2.2. The best of these features is selected and the GP prediction is tested on a validation set of five selected tracks. This GP prediction is also compared to the mentioned alternative approaches.

4.1 Experimental Setup

For our experiments, we used pointcloud data provided by the Stanford Racing Team. The data was collected during the Urban Challenge 2007 using a so called velodyne laser range scanner. The scanner as well as the robot “Junior” that competed for Stanford are depicted in Figure 4.1. The data corresponds to a city environment with other moving cars. Typical scenes are shown in Figure 4.2 and Figure 4.4. The velodyne laser provided pointclouds, each consisting of about 80.000 points, up to 10 times per second. The points represent the endpoints of the laser range measurements. The max range of the velodyne is about 120 meters. However, the usable range is only about 40 to 50 meters. Beyond this distance the measurement density decreases quickly.

The pose $\Theta = (x, y, z, \phi, \theta, \psi)$ of the robot was also provided. x, y, z denote the location of the robot in its local coordinate system, ϕ denotes the pitch, θ the roll and ψ the yaw of the robot.

(a) Junior, ©Stanford Racing Team



(b) Velodyne laser, ©Velodyne

Figure 4.1: The robot "Junior" and the laser that were used for obtaining the pointcloud data.

We assume the pose is correct which is a strong assumption but experiments showed that the pose was very accurate in the x and y position and the yaw. The z position as well as the pitch and roll were more prone to noise. This seemed mainly do to uneven terrain like bumps in the road.

Regression and Gaussian processes are a supervised learning scheme. That means that the training dataset D consists of pairs of inputs X and corresponding outputs Y. Hence, we need to obtain both, the inputs and corresponding outputs for training the GPs. Furthermore, to compare the estimate of our motion estimation method with the other approaches, the real motion had to be given. We obtained this ground truth by manually labeling dynamic objects in the scans, fitting a rectangular box to the range scan data of the dynamic object. This box is fit once using different scans and different observation directions such that the width and the length of the dynamic object are measured correctly. The center of the box and its rotation are then stored for each scan labelled. Using this information, we are able to specify its speed and heading. The heading could be labelled with a precision of about two degrees while the speed of the vehicle was labelled with a precision of about one meter per second. The reason for the high speed labeling error is the changing appearance of the objects in different scans. The speed labels were corrected by averaging over the last eight scans. We selected and labelled nine different tracks to obtain the training data. Each track consisted of about 100 to 200 scans. Altogether about 130.000 map patches were generated as training data consisting of many different point of views and moving directions.

4.2 Evaluation Methods

In this section, we explain the methods used for evaluating the learning results and argue why these method are suitable for evaluating our method.

Cross-Validation

K-fold cross-validation is a commonly used tool to evaluate a learner. The goal is to ensure that the prediction result achieved by a learner does not depend on a special set of "great" samples that are, e.g., manually selected. K-fold cross-validation divides the sample set into K parts. One part is retained for testing the learned model. The other $K-1$ parts are used for training. This process is repeated K times such that each of the K parts is used for testing the model once. Depending on the application there are different measures for evaluating the learning result. We use the root-mean-square error (RMSE) for evaluation which is given by

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2}, \tag{4.1}$$

where N denotes the number of samples tested and $e_i = Y_i - \bar{Y}_i$ the residual of the observed and the predicted value. The RMSE penalizes the outliers much, due to the squaring. In addition, we also state the mean error which does not penalize outliers as much. This results in a good intuitive estimate of the error in practice. In addition, to evaluating the error for each fold, the error is commonly averaged over the folds to give an average estimate for the learning result.

We perform a standard 10-fold cross validation to evaluate the learning. Thus we select a sample set of 10.000 samples using importance sampling without replacement on the entire sample set. (In Section 3.1.2 we described the problem of generating training samples in detail.) We then divide the selected samples into ten parts by sampling each part randomly without replacement and perform cross-validation.

Using a Validation Set

In addition to cross-validation, we also use an independent validation set. A validation set is independent of the training and test set used used for cross-validation before. In this thesis, we use a validation set to evaluate our approach on unseen data and for comparison to the alternative methods. The goal is to evaluate whether the learner is prone to overfitting or generalizes to unseen data. We used a validation set of five tracks representing common traffic situations for evaluation.

4.3 Evaluation of the Features

In this section, we evaluate our proposed method for motion prediction using cross-validation. First, we determine the effect of the map prior on the quality of the heading prediction. Second, we evaluate which of proposed features, namely the map patches or the Gabor filter features are better suited for predicting the heading. Third, we evaluate the speed prediction using map patches as features. Finally, we determine the effect of the noise function on the heading and the speed prediction. Selecting only those samples that result in a low predictive variance we expect the error to decrease.

In this thesis, we introduced two different sparse GPs. As we observed in Section 3.1.3 the sparse GP with pseudo-inputs (SPGP) already models some kind of input-dependent noise by shifting the pseudo-inputs away from the noisy input locations. The second GP learns the noise function of heteroscadestic datasets by learning an individual noise term for each pseudo-input (see Section 3.1.3). In the following, we refer to this GP as SPGP-HET. The SPGP-HET is expected to better estimate the underlying noise function than the SPGP. As both GPs learn a different noise function, we give the result for both of them.

Before evaluating the prediction we specify the exact size of the patches, the number of training samples and the number of pseudo-inputs used.

Patch Size

For choosing the patch size, several constraints have to be met. The patches had to be small enough to capture only local features that are as independent of the underlying object as possible. In addition, the frequency of the laser range scanner and the speed of the objects to be tracked had to be considered. The higher the laser frequency and the slower the moving object, the smaller the patches can be. Finally, we chose the patches to be square, each side of the square being 1.10 m long. We used a 11×11 grid to represent the patches. Considering the frequency of the laser of 10 Hertz and the size of a grid cell, we are able to detect motion within the range of 2 -10 m/s.

Number of Training Samples and Pseudo-Inputs

The number of training samples and the number of pseudo-inputs is mainly limited by practical considerations. We found that using 100 pseudo-inputs and 1600 training samples still resulted in a reasonable learning time while providing good results. Generally, we expect the prediction result to improve using more samples.

4.3.1 Heading Prediction

The goal of this section is to evaluate the prediction of the heading of moving objects. We randomly selected a set of 16.000 samples from the entire training set for cross-validation. In the following we evaluate the two features proposed in Section 2.2:

1. The representation of a difference map patch in the Gabor filter space. Upon extraction we compute the dot product of the map patches and the Gabor filter bank that was constructed in Section 2.2.3. The result is a 16 dimensional representation of the patch.

2. Using the original map patches directly as an input to the GP. We extract a map patch out the difference and the reflection map. Considering the size of the patches this constitutes 242 dimensional feature vector.

The map patches are aligned to the direction in which they are observed. Additionally, to these two different features, we also evaluate the effect the map prior has on the training data. We found that the map priors 0.01 and 0.5 are possible (see Section 2.1.2).

As mentioned in Section 3.1.1 we split the heading prediction into estimating the sine and the cosine of the heading and learn one GP for each part. Upon the prediction we restore the angle.

The average RMSE of the heading predictions, averaged over all folds, is given in Table 4.1. The main result of this experiment is that using the Gabor features for prediction only results in

	Gabor Features		Map Patch	
	Map Prior: 0.01	Map Prior: 0.5	Map Prior: 0.01	Map Prior: 0.5
SPGP	59.32	61.46	44.67	41.29
SPGP-HET	59.79	62.21	44.06	42.51

Table 4.1: Average RMSE in degrees for the heading prediction.

a rough estimation of the heading. The prediction has an error of about 60 degrees which corresponds to an opening angle of 120 degrees. The RMSE using the map patches is about 15 degrees lower and therefore shows a much better result. In the following, we will therefore only use map patches for predicting the heading.

The results differ as to which prior to prefer comparing the Gabor features and the map patch features. On the one hand, using a map prior of 0.01 slightly improves the RMSE using the Gabor filter features. On the other hand, the prior of 0.5 is the better choice when using map patches. However, because we selected the map patches as features we choose 0.5 as the map prior.

The prediction quality of the SPGP and the SPGP-HET seem to be similar. Using either one only changes the RMSE slightly. At this point, the result is not satisfactory because the GP only gives a rough estimation of the angle. However, one of the main advantages of using GP regression is that the GP is capable of predicting a full distribution. We expect the result to improve using only those samples with a low variance. A sample with a low variance means a sample with a predicted variance that is further away from the mean of the variances then one standard deviation. Because we use two GPs to predict the heading (one for predicting the sine and one for predicting the cosine part of the heading) we only selected those samples having a low variance on each part. Table 4.2 depicts the RMSE using this approach and map patches as features. As we can observe the average RMSE decreases much. This result shows that the noise

	Map Prior: 0.01	Map Prior: 0.5
SPGP	22.45	19.76
SPGP-HET	21.93	18.48

Table 4.2: Average RMSE in degrees using map patches as features. The RMSE was only calculated for inputs with a predicted low variance.

function can be used as a good estimator of the quality of the input sample. Using the SPGP-HET improves the result which is evidence that the SPGP-HET better estimates of the noise then the SPGP.

The result presented here is computed based on a training set of single patches only and not on entire segments. In Section 4.4, we evaluate our approach on a validation set of five different tracks. Merging the predictions as described in Section 3.1.4 we expect to further improve the result such that a robust prediction of the heading becomes possible.

4.3.2 Speed Prediction

In this section, we evaluate our approach for predicting the speed. We only use the map patches as features as the Gabor features resulted in a much higher error predicting the heading. As the prediction of the speed is based on the result of the heading prediction, a correct prediction of this angle is essential. For this reason, we rely on using a map prior of 0.5. This prior resulted in the smallest error using the map patches as feature. We performed cross-validation on the same training set used for evaluating the heading prediction but with adapted map patches. In difference to the heading prediction the map patches are now aligned to the heading of the object (see Section 2.2.2). The average RMSE of the folds is given in Table 4.3.2. As for the angle

SPGP	3.62
SPGP-HET	3.21

Table 4.3: Average RMSE in $\frac{m}{s}$ obtained by using cross-validation and map patches as features.

prediction the SPGP-HET shows the best result. We also calculate the RMSE only using the samples for which a low variance is predicted. A low noise variance means a variance further away from the mean of the variances than one standard deviation. The RMSE using this measure is depicted in Table 4.3.2. Again, the RMSE drops tremendously which is good evidence that the

SPGP	1.74
SPGP-HET	1.53

Table 4.4: Average RMSE in m/s obtained by using cross-validation and map patches as features. The RMSE is only computed using samples with a low predicted variance.

noise function is learned well. The SPGP-HET achieves a better result than the SPGP.

Having set the map prior and chosen the features suitable for estimating motion the next section evaluates our approach on a validation set of five tracks. Merging the prediction for an entire segment, we expect the error for the speed and the heading estimation to decrease. We also compare our approach to the alternative methods described in Section 3.2.

4.4 Comparison on Real Data

We showed that the map patches, computed using a map prior of 0.5, are the best of our proposed features for estimating motion. In this section, we use these features and a validation set of five tracks to evaluate our approach and compare it to ICP and Kalman filtering. Instead of evaluating single predictions, we compare these methods predicting motion for an entire object. For our approach, that means that we cluster the pointcloud, predict a motion for each point and compute a motion estimation for an entire segment by merging the single predictions.

The goal is to determine which merging strategy, merging all predictions or only those with a low variance, is to be preferred. In addition, we evaluate which GP, the SPGP or the heteroscadestic SPGP (SPGP-HET) performs better using the merging strategies above. As already mentioned, it is important that the noise function is learned correctly, because in many situations large parts of the observed object are not reliable for motion prediction. Hence, we expect the SPGP-HET to result in a smaller error then the SPGP. Additionally, we compare the GP prediction to the alternative approaches. This allows us to finally state advantages and disadvantages of the methods and determine whether one approach is be preferred over the others in all situations or not.

We created a validation set consisting of five common traffic situations. Each of these situations consist of 30 to 100 subsequent scans and about 200 predictions per scan. The situations were chosen such that as many different driving situations and especially observation directions could be covered.

This section is structured as follows. We evaluate one situation after another, first describing the situation and then presenting the result for the heading and speed prediction separately. For clearness, we abbreviate the name of the GP methods in the following. Table 4.5 states the abbreviations and their meaning. In this table, heteroscadestic GP refers to the sparse GP learning a noise variance for each pseudo-input (see Section 3.1.3). At first the different situations are

	Merging all predictions	Merging best predictions	Heteroscadestic GP
SPGP-ALL	-	-	-
SPGP-BEST	-	X	-
SPGP-HET-ALL	X	-	X
SPGP-HET-BEST	-	X	X

Table 4.5: Abbreviations of the methods applied and their meaning

described and the four different prediction methods (SPGP, SPGP-HET combined with the two merging methods) as well as the ICP and Kalman filter method are tested and evaluated for each track. Secondly, we give an overall conclusion at the end of this section.

Situation 1

Situation Description

A car approaches and passes the robot on the left side (see Figure 4.2). This situation is very common in urban traffic scenarios. Also, it is very valuable to us because many different observation directions are covered. At first, the object is seen only from the front and moves with an angle of approximately 180 degrees relative to the observation direction. Thus it approaches the observer almost directly. While it passes, the object is only seen from the side and moves at an angle of 90 degrees relative to the observation direction. Afterwards, the back of the object is seen moving at 0 degrees relative to the observation direction. For the GP prediction we expect to perform well when the other car is approaching and after it the passed.

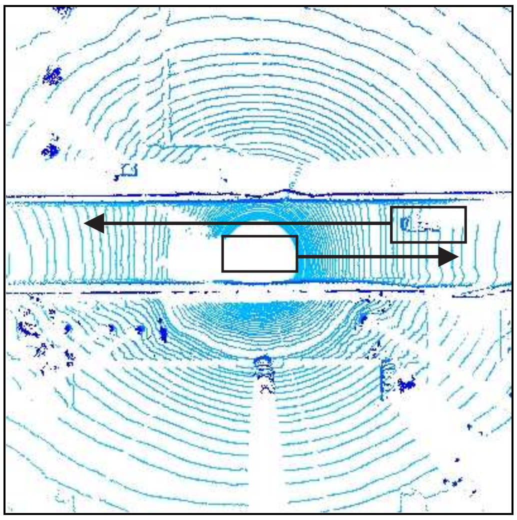


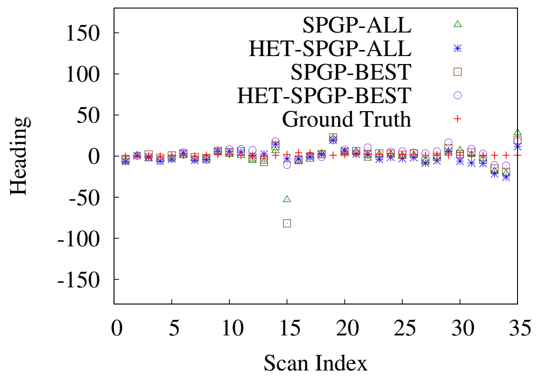
Figure 4.2: Situation1: Another car approaches driving into the opposite direction and passes the robot on the left side.

	RMSE	ME
SPGP-ALL	12.65	6.96
SPGP-BEST	16.32	7.61
SPGP-HET-ALL	8.7	6.5
SPGP-HET-BEST	7.97	6.26
KF	7.21	5.2
ICP	33.76	12.93

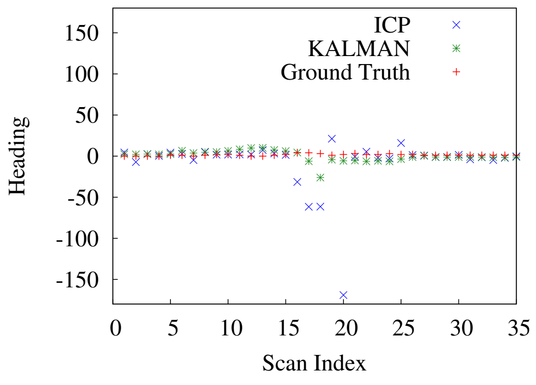
Table 4.6: Situation 1: RMSE and ME of the heading prediction in degrees.

Heading Prediction

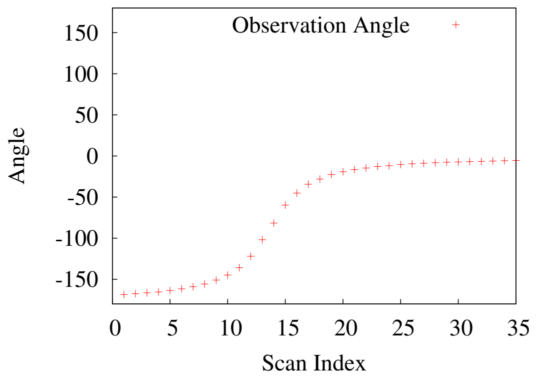
In this situation the Kalman filter results in the best angle prediction. The changing center of mass of the observed object measurements does not have a huge effect on the prediction. The ICP method results in a much higher error then the other methods. Although ICP often predicts the angle with a small error (see Figure 4.3b) at some points it also makes larger errors. The reason is that at the scan with the largest error the other car is located right next to the robot and drives through an area which cannot entirely be observed (scans 15 to 17) due to the mounting of the laser. The result is a large change of the pointcloud data belonging to the moving object. Thus, the ICP algorithm can only find the best match considering the points available. The Kalman filter handles this situation better due to its smoothing motion model. The changing appearance of the car in this situation also results in motion patterns that do not capture the real motion of the object. Nevertheless, the GP methods result in a good prediction. The SPGP-HET-ALL shows the best result of all the GP predictions. Merging only the best predictions in this situation decreases the



(a) Heading Prediction GP



(b) Heading Prediction: best GP, ICP, KF



(c) Observation Direction

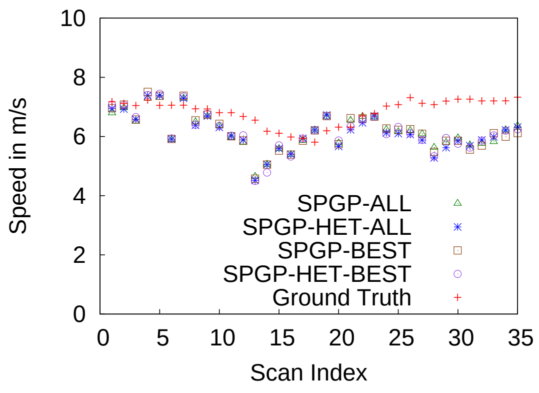
Figure 4.3: Situation 1: Evaluation of the heading prediction

error compared with the SPGP-HET. This is evidence that the noise function is learned correctly.

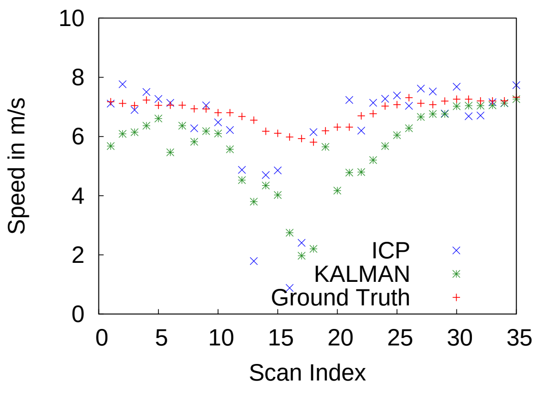
Speed Prediction

The best speed prediction in this situation is obtained using the SPGP-ALL. Although the ICP algorithm predicts the speed closer in most parts of the track it also makes some huge mistakes between the scans 12 to 15. While the GP prediction does not perform as well in this part and at the end of the track it makes less huge mistakes compared to ICP. The reason for the performance of ICP is mainly the car driving close to the robot. While driving by, a large part of the car cannot be observed by the laser and therefore the number of obtained laser measurements changes quickly. This causes the appearance of the observed object and the motion patterns to change largely. Hence, the Kalman filter shows a result similar to ICP. The changing center of mass of the observed object which changes because the object is observed from different points of view.

Hence, the center of mass of the observed mass seems to have a larger influence on the speed prediction than on the angle prediction, However, the GP methods are not as much influenced by this changing pattern (see Figure 4.3) as the others.



(a) Speed prediction: GP methods



(b) Speed prediction: ICP, KF

Figure 4.4: Situation 1: Evaluation of the speed prediction

	RMSE	ME
SPGP-ALL	0.89	0.8
SPGP-BEST	0.92	0.83
SPGP-HET-ALL	0.94	0.88
SPGP-HET-BEST	0.93	0.85
KF	1.59	1.25
ICP	3.28	1.53

Table 4.7: Situation 1: RMSE and ME of the speed prediction in $\frac{m}{s}$.

Situation 2

Situation Description

In this situation, a car is following the robot while both are turning left (see Figure 4.4). This situation poses the challenge that both, the observer and the moving object are rotating. In Section 3.1.1 we specified the estimation of motion as the process of estimating the moving direction and the speed of an object. This includes the implicit assumption that the translational velocity is much higher than the rotational velocity. In situations in which the rotational velocity cannot be neglected this might lead to wrong results.

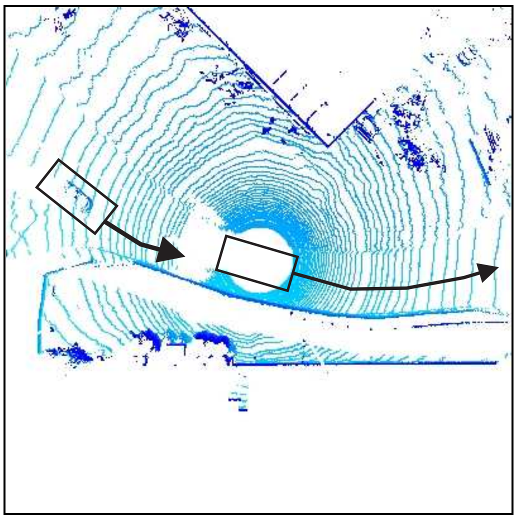


Figure 4.5: Situation2: The other car follows the observer and both make a long left turn.

Heading Prediction

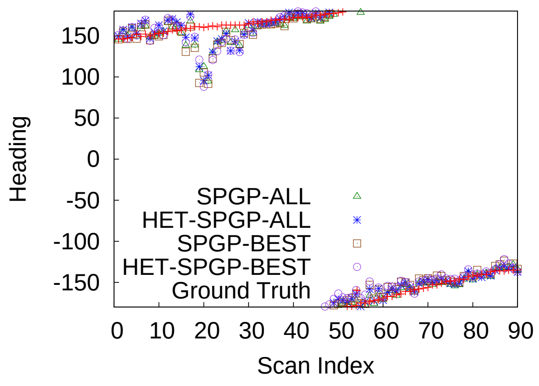
In this situation, the Kalman filter shows the best prediction results followed by the ICP algorithm. However, many GP predictions are very close to the true angle. Only in one part of the track the GP predictions result in a high error. In this part, the rotational velocity of the car following is very high compared to the rest of the track. Thus our assumption about a low rotational velocity relative to the translational velocity, is violated. This fact causes patterns that are similar to ones known from the training data but point into another direction. Additionally, these patches are assigned a low noise. This causes the error terms to increase merging only those predictions with a low variance. Hence, the SPGP-BEST and SPGP-HET-BEST result in a larger errors.

	RMSE	ME
SPGP-ALL	13.15	7.62
SPGP-BEST	15.94	9.67
SPGP-HET-ALL	14.34	8.9
SPGP-HET-BEST	15.93	14.34
KF	1.86	1.54
ICP	10.6	6.09

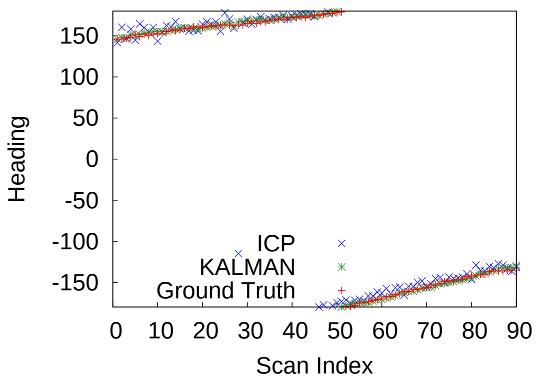
Table 4.8: Situation 2: RMSE and ME of the heading prediction in degrees.

Speed Prediction

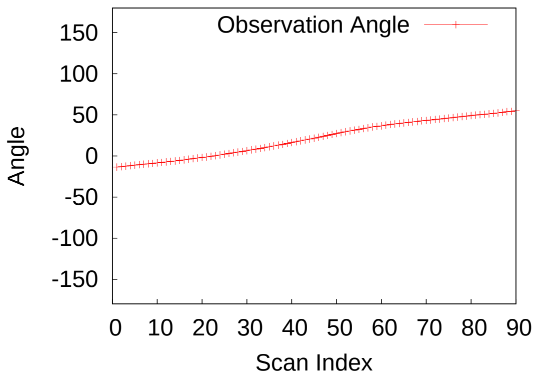
The observed car in this situation almost drives at constant speed. The best speed prediction in this situation considering is achieved by the Kalman filter (see Table 4.4). The speed prediction of



(a) Heading Prediction GP



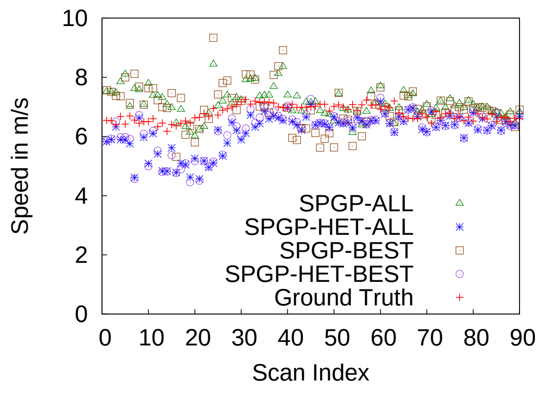
(b) Heading Prediction: best GP, ICP, KF



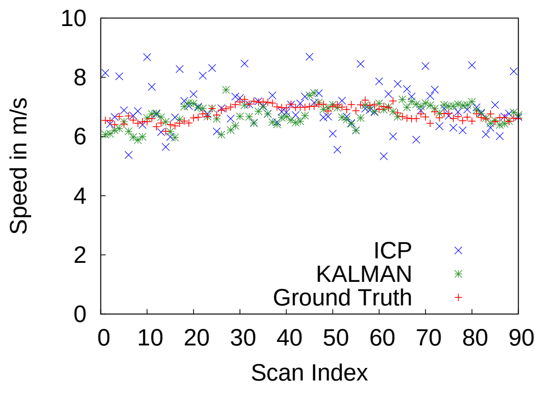
(c) Observation Direction

Figure 4.6: Situation 2: Evaluation of the heading prediction

the SPGP-ALL is also very close to the true speed in the second part of the track. In the beginning and especially from scan 15 to 35 the speed prediction is very much influenced by the wrongly estimated angle (see Figure 4.6). This fact supports our assumption that a good estimation of the angle is needed for a correct prediction of the speed. Considering the error terms in Table 4.4 we note that using only those single predictions with a low variance improves the RMSE in this situation. The ICP algorithm often results in a good speed estimate but also makes larger mistakes in all parts of the track. One reason for this is the scan density. Because the observed car is further away than in other tracks the number of laser measurements obtained from the car is smaller. Additionally, the observed shape of the car changes because of the rotation.



(a) Speed prediction: GP methods



(b) Speed prediction: ICP, KF

Figure 4.7: Situation 2: Evaluation of the speed prediction

	RMSE	ME
SPGP-ALL	0.61	0.37
SPGP-BEST	0.78	0.62
SPGP-HET-ALL	0.82	0.67
SPGP-HET-BEST	0.81	0.66
KF	0.4	0.34
ICP	1.04	0.68

Table 4.9: Situation 2: RMSE and ME of the speed prediction in $\frac{m}{s}$.

Situation 3

Situation Description

The observer approaches an intersection. The moving object is turning right (see Figure 4.4). As in the situation before, the rotational velocity of the observed car is high.

Heading Prediction

The Kalman filter approach results in the best angle estimation in this case and is very close to the true angle almost over the entire track. The ICP algorithm also result in a low error. In the beginning of the track, ICP result in the largest error. Again, this is caused by the high rotational velocity of the observed car as well as the movement of the robot itself which causes a large change in the appearance of the moving object. For the same reason, the GP methods result in a wrong angle prediction in this part. While the observed car passes (scan 20 to 30) the GP methods also result in an slightly larger error then for the rest of the track. The changing appearance of the object is caused by the other car driving closely to the robot. It results in patterns on the

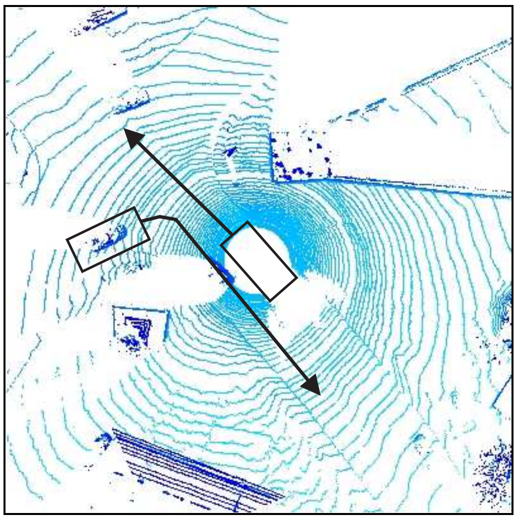


Figure 4.8: Situation 3: The observed car makes a sharp right turn and drives by the other car. The robot approaches and crosses the intersection.

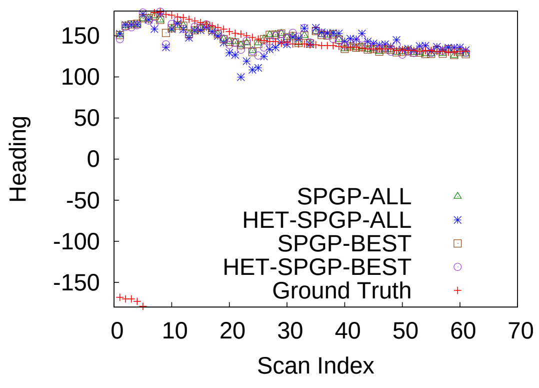
difference map which are different from the ones commonly describing this movement direction. Additionally, the observation direction in this stage is approximately 90 degrees (see Figure 4.9c) which constitutes the worst case for the GP prediction. The best performing GP method in this situation is the SPGP-ALL. Using only the best predictions in this case has a negative effect. The error of the SPGP-BEST increases. In this situation, the noise function modeled by SPGP-HET seems to have been estimated better than in the situations before. Using only those predictions with the lowest variance, the error of the error is greatly reduced (see Table 4.4).

	RMSE	ME
SPGP-ALL	11.18	8.06
SPGP-BEST	11.47	8.33
SPGP-HET-ALL	17.53	13.25
SPGP-HET-BEST	13.24	9.53
KF	1.70	1.44
ICP	6.36	4.44

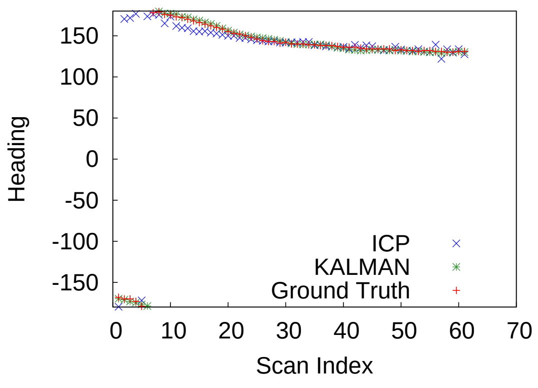
Table 4.10: Situation 3: RMSE and ME of the heading prediction in degrees.

Speed Prediction

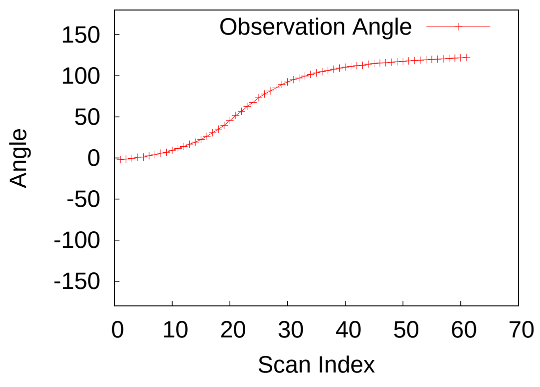
The best speed prediction in this case result from ICP algorithm which predicts the speed with a small error over the entire track. The GP methods also show a small overall error. All GP methods result in a very similar prediction result (see Figure 4.10a). The GP methods do not result in a much larger error in the beginning of the track compared to the entire track, although the angle



(a) Heading Prediction: GP



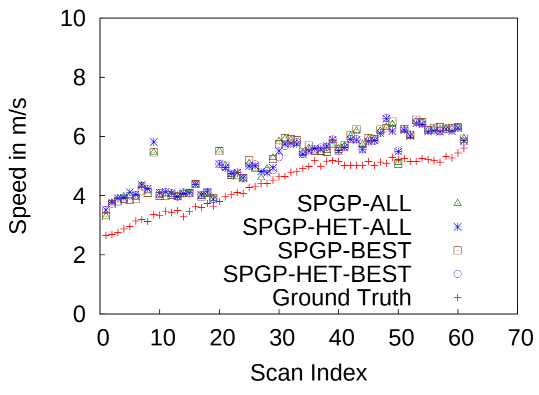
(b) Heading Prediction: best GP, ICP, KF



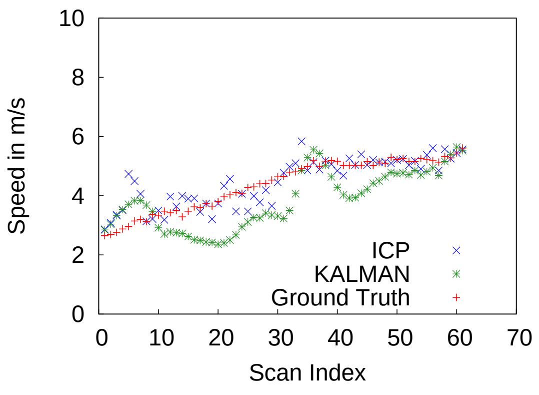
(c) Observation Direction

Figure 4.9: Situation 3: Evaluation of the heading prediction

estimation (see Figure 4.9a) was not as good in this part. Differing from situation two, the error in the angle estimation does not have a great influence on the speed estimation. The Kalman filter also results in a good speed estimate but performs slightly worse than the best GP (see Table 4.4). Due to the changing center of mass that is observed it underestimates the speed over large parts of the track.



(a) Speed prediction: GP methods



(b) Speed prediction: ICP, KF

Figure 4.10: Situation 3: Evaluation of the speed prediction

	RMSE	ME
SPGP-ALL	0.9	0.81
SPGP-BEST	0.91	0.84
SPGP-HET-ALL	0.88	0.78
SPGP-HET-BEST	0.83	0.7
KF	0.84	0.73
ICP	0.46	0.32

Table 4.11: Situation 3: RMSE and ME of the speed prediction in $\frac{m}{s}$.

Situation 4

Situation Description

The observer approaches an intersection while another car crosses it with an angle of approximate 90 degrees (see Figure 4.12c) relative to the direction the dynamic object is observed (see Figure 4.4). This is the worst case situation for the approach described in this thesis because only few parts of the observed vehicle, the front and the back, actually appear to be moving. For the other parts the distance measured stays the same. (Also see Figure 3.1a which depicts this situation.) Therefore we expect the GP prediction to not perform as well when the other car is located directly in front of the robot.

Heading Prediction

In this situation, both, the ICP algorithm and the Kalman filter perform better than the GP based approach and only result in an error of about two degrees. The best GP results in an RMSE of about 13 degrees (see Table 4.4). Considering the hardness of this situation, the maximal error of

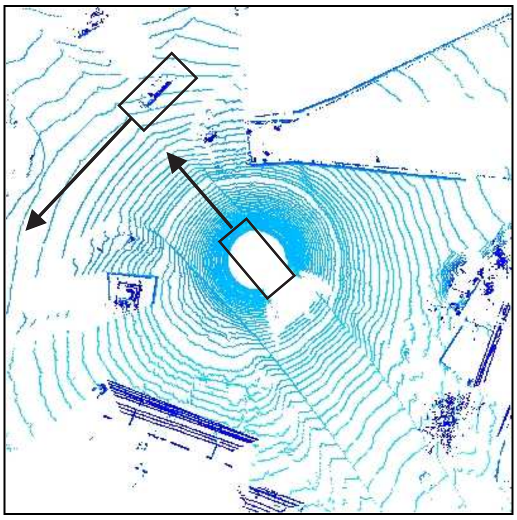
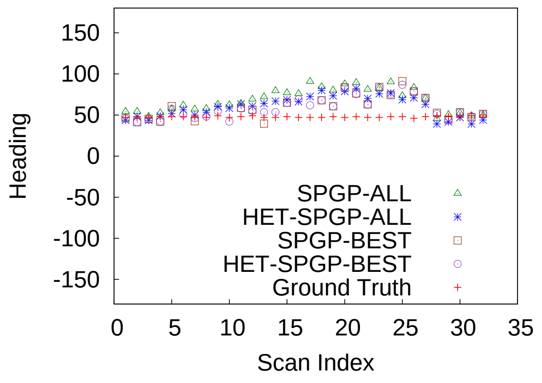


Figure 4.11: Situation 4: The robot approaches an intersection while the tracked car crosses the intersection.

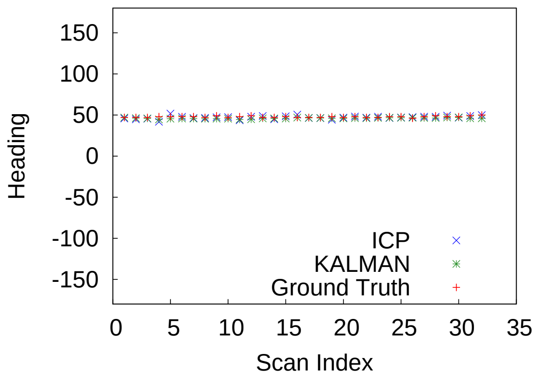
about 20 degrees is a good achievement. Consider Figure 4.4 which depicts the true heading and the single predictions made by the SPGP-HET. The predictions are sorted by their observation angle. We can easily note that many predictions are very close to the true angle. Especially those predictions made at the beginning and at the end of the object are close to the true angle (The beginning of the object is located towards the origin of the figure.). However, there are also many predictions pointing into the wrong direction. The error bars in the figure denote the standard deviation of the predicted sine part. Unfortunately, the errors bars do not differ much between those predictions pointing into the correct direction and those that are not. The large number of wrong predictions influences the result too much. Therefore, we can conclude that although the error is reduced using the SPGP-HET-BEST the noise function was not learned correctly.

	RMSE	ME
SPGP-ALL	24.67	20.17
SPGP-BEST	19.32	14.68
SPGP-HET-ALL	18.11	14.93
SPGP-HET-BEST	17.27	12.75
KF	2.07	1.76
ICP	2.05	1.49

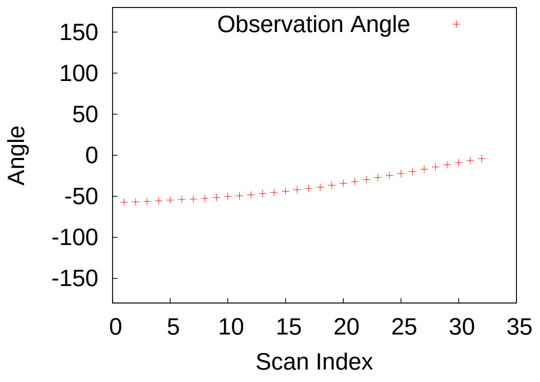
Table 4.12: Situation 4: RMSE and ME of the heading prediction in degrees.



(a) Heading Prediction GP



(b) Heading Prediction: best GP, ICP, KF



(c) Observation Direction

Figure 4.12: Situation 4: Evaluation of the heading prediction

Speed Prediction

The Kalman filter results in the best speed prediction in this situation. The ICP algorithm also shows a good speed estimation but results in larger errors in the beginning of the track. Despite a estimation error of the heading (see Figure 4.12a), the SPGP-HET-BEST results in a very similar result as the ICP. The slightly higher RMSE indicates that the GP has more outliers. The heading estimation does not effect the speed estimation as much as suspected.

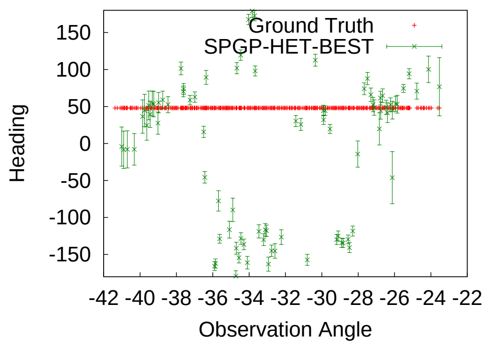
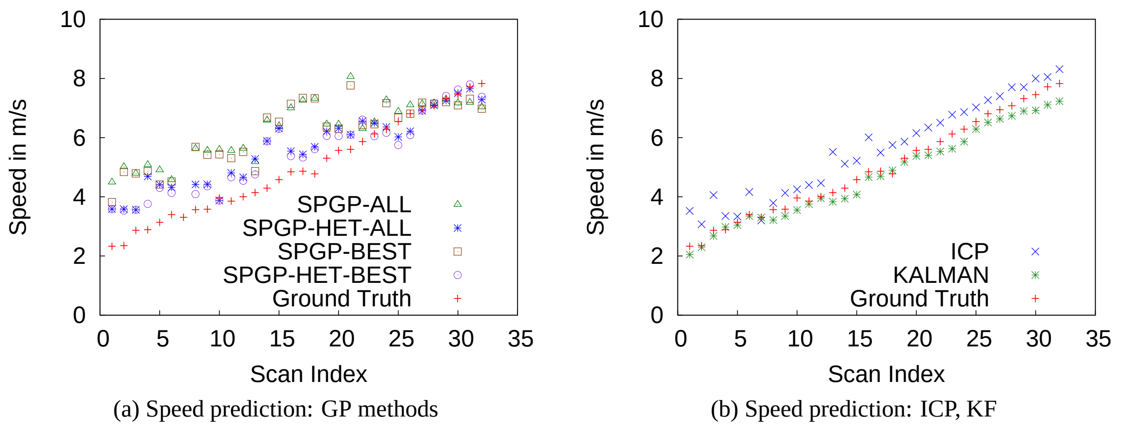


Figure 4.13: Situation 4: Detailed view of the single heading predictions at scan 20. The error bars denote the standard deviation of the sine part of the heading prediction.



(a) Speed prediction: GP methods (b) Speed prediction: ICP, KF

Figure 4.14: Situation 4: Evaluation of the speed prediction

	RMSE	ME
SPGP-ALL	1.59	2.54
SPGP-BEST	1.5	2.26
SPGP-HET-ALL	0.87	0.77
SPGP-HET-BEST	0.76	0.59
KF	0.32	0.27
ICP	0.69	0.62

Table 4.13: Situation 4: RMSE and ME of the speed prediction in $\frac{m}{s}$.

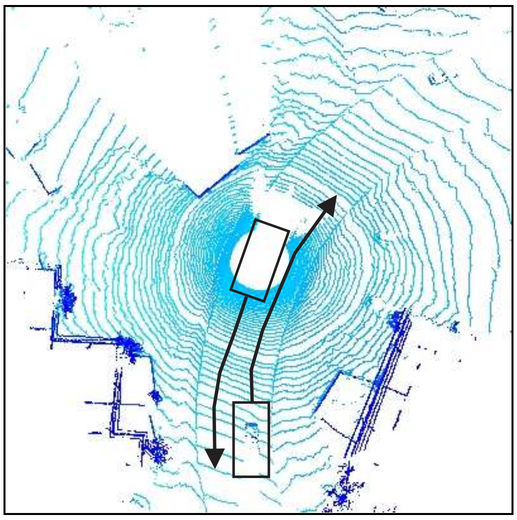


Figure 4.15: Situation5: The other car approaches the robot almost directly and passes it on the left side.

Situation 5

Situation Description

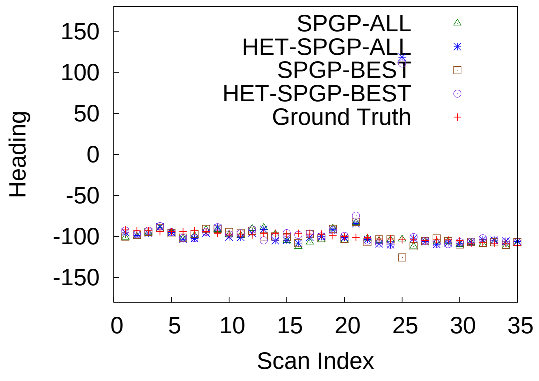
The observer is turning left while the observed car is making a long right turn (see Figure 4.15). The changing appearance of the observed object change quickly due to the translational and rotational movement of both cars.

Heading Prediction

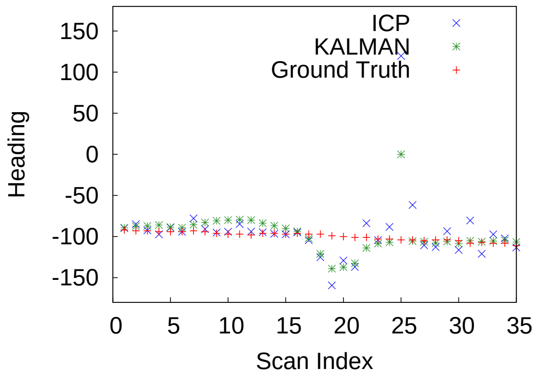
The SPGP-ALL prediction results in the lowest error in this situation. From Figure 4.16c we can observe that this result is obtained for a large variety of observation angles. Thus the prediction quality in this case does not depend on the observation direction. The SPGP-HET results in a much higher RMSE which is mainly due to one extreme outlier at the end of the track (see Figure 4.16a). Thus the mean error of its prediction is much lower. Using only the best single predictions as an estimator for the direction did not result in a better estimate in this situation. The prediction of the Kalman filter is very close to the true angle, almost over the entire track. The ICP method results in a comparable prediction quality in the beginning of the track. The error of the ICP prediction is mainly caused by the low scan density at the end of the track. The overall RMSE of the Kalman filter and the ICP method are mainly influenced by one outlier at the end of the track (see Figure 4.16b, scan 25). The GP methods result in much smaller error at this position.

	RMSE	ME
SPGP-ALL	6.39	4.87
SPGP-BEST	6.58	4.63
SPGP-HET-ALL	24.01	8.66
SPGP-HET-BEST	25.33	8.27
KF	18.91	8.46
ICP	29.03	15.14

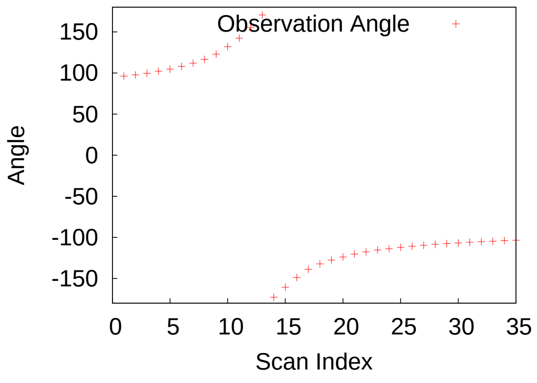
Table 4.14: Situation 5: RMSE and ME of the heading prediction in degrees.



(a) Heading Prediction GP



(b) Heading Prediction: best GP, ICP, KF

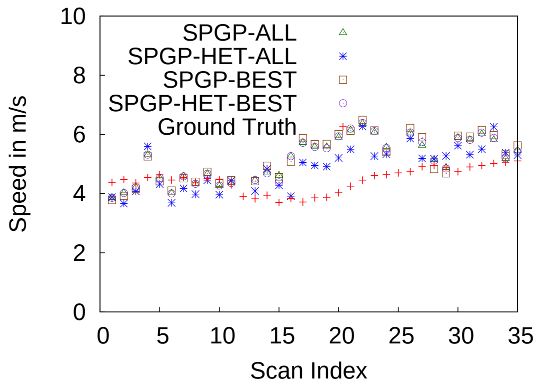


(c) Observation Direction

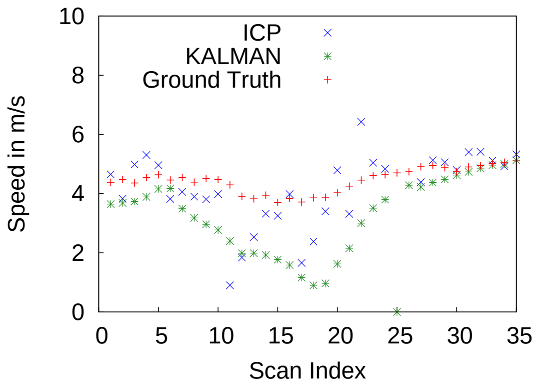
Figure 4.16: Situation 5: Evaluation of the heading prediction

Speed Prediction

The GP methods in this situation outperform the ICP algorithm and the Kalman filter (see Table 4.4). The Kalman filter results in a high error due to its motion model and the changing object appearance. ICP predicts the speed well in the beginning and at the end of the track. Most errors occur between scans 10 and 25 (see Figure 4.17b). The errors of ICP caused by a movement of the object similar to situation one. Driving by the robot the car is only partially observed. However, the ICP method results in larger errors and more extreme outliers in this situation then the GP methods.



(a) Speed prediction: GP methods



(b) Speed prediction: ICP, KF

Figure 4.17: Situation 5: Evaluation of the speed prediction

	RMSE	ME
SPGP-ALL	1.03	1.07
SPGP-BEST	1.08	1.2
SPGP-HET-ALL	0.78	0.62
SPGP-HET-BEST	1.04	1.08
KF	1.65	1.27
ICP	102.52	25.19

Table 4.15: Situation 5: RMSE and ME of the speed prediction in $\frac{m}{s}$.

4.5 Discussion

In this section, we summarize the result of the experiments and answer the questions raised in the beginning of the chapter.

How is the GP prediction influenced by the map prior? The cross-validation experiments in Section 4.3.1 showed that on the one hand, when using the Gabor filter features a map prior of 0.01 slightly improved the result. On the other hand, a map prior of 0.5 improved the heading estimate using map patches as features.

Which of the proposed features is better for estimating the heading? The map patch features or Gabor filter features? The experiments in Section 4.3.1 resulted in a much smaller RMSE when using the map patches as features compared to using the Gabor filter results. While the GP prediction based on the Gabor filter features resulted in an RMSE of 60 degrees, the RMSE could be reduced to about 40 to 45 degrees using the map patches as features.

Can the GP based approach to motion estimation robustly predict the angle and speed of the dynamic objects? The GP based approach is able to predict the angle robustly in many situations. In four out five situations the RMSE of the heading is less then twelve degrees. In one situation the GP based approach resulted in the best heading prediction when compared to the other approaches. Most errors of the GP prediction occurred in situations in which the rotational velocity of the tracked object could not be disregarded or the observed object moved in an almost right angle relative to the observation direction. However, in these hard situations a reasonable estimation of the moving direction was still possible. The SPGP-ALL showed the best overall performance of the GP methods. In cases where the SPGP-HET models the noise function correctly, its result is better then the SPGP-ALL.

The speed was also often estimated with a high accuracy resulting in an RMSE between 0.6 m/s and 1.6 m/s. In two out of five situations our approach showed the best result. The experiments showed that a wrong angle prediction influences the speed prediction. However, a wrong prediction of the heading, e.g., in situation three, must not necessarily result in a wrong prediction of the speed but is robust to small errors in the heading.

Is the noise function learned correctly by the SPGP and the SPGP-HET? The first experiments using cross-validation on a training set showed that modeling the noise function does have an effect. The RMSE was greatly reduced considering only those predictions with a low predicted variance. However, the experiments using a validation set of five tracks showed that in two situations the error in the heading. In three situations the error in the speed estimation could be reduced. That leads to the conclusion that for some parts of the input space the noise function was already learned correctly. There are two types of situations in which the error increased. First, situations in which the observed car has a high rotational velocity. Second, situations where the observed car moves in a 90 degree angle relative to the observation direction, e.g., situation four.

How accurate is the prediction compared to using the state of the art motion estimation

methods ICP or the Kalman filter? The overall performance of our approach for motion estimation is often comparable to the result of the alternative approaches. The Kalman filter often results in the best prediction of the heading. Its motion model smoothes the prediction such that larger errors can be avoided. The changing appearance of a dynamic object does not effect the estimation of the heading much. However, the speed prediction of the Kalman filter is effected much more. Our approach resulted in a better speed prediction in such situations. In most situations the Kalman filter showed a better performance predicting the heading of the moving object. In one situation the GP approaches perform better. The ICP algorithm showed good results predicting the angle as well as the speed in most situations. However, it also showed to be more prone to noise in some situations than the GP predictions. In situation one for example, the GP methods achieved a much better result when the moving object could only partially be observed. Additionally, in situations in which only few measurements of the dynamic object are obtained, the ICP algorithm resulted in large errors. Summarizing, our method for predicting motion does not always result in the smallest error. However, considering the different situations it did show a competitive overall performance.

In addition, we must note that we carried out the experiments having solved the data association needed for applying the ICP and Kalman filter method. The ICP algorithm computes the speed and direction from the displacement of two given pointclouds. The Kalman filter uses the center of mass of each pointcloud as measurement. For our experiments, the data association was done manually. In practice, this data association needs to be solved autonomously. If mismatches occur, the Kalman filter and the ICP method result in larger errors. This association of objects in subsequent measurements is not needed for the application of our method. Our approach solves the data association using the difference map and therefore does not depend on correctly matching objects.

5 Conclusions

In this thesis, we introduced a novel approach to motion estimation that can detect and estimate the heading and the speed of a moving object out of range data. At each point in time, the algorithm gets a pointcloud and the location of the sensor as input. Prior hypothesis of moving objects are then obtained by clustering the pointcloud. We use a difference map for capturing motion information. For each point in a cluster we predict a heading and a speed using Gaussian process regression and features extracted out of the difference map. A motion prediction for the entire object is achieved by merging the single predictions.

Chapter 2 showed how suitable motion features can be constructed from the laser range measurements. We constructed a difference map of two subsequent grid maps. The difference map captures the motion flow of the environment and solves the problem of associating the measurements of different scans. Each scan was also clustered to obtain prior hypothesis of dynamic objects consisting of all scan points belonging to this cluster. Given the difference map and pointcloud clusters we proposed two different features for learning. Firstly, for each laser measurement we extracted its local environment from the current occupancy map and the difference map. This so called map patch constituted the first feature. Secondly, we applied a set of Gabor filters to the difference map patches and used its resulting 16 dimensional representation, as a second feature.

In Chapter 3 we showed that predicting the speed and heading of a moving object can be regarded as a regression problem inferring the two latent functions from data. We also showed that it is advantageous to choose a different parameterization of the heading such that the function to be inferred is continuous. Therefore we parameterized the heading using its sine and cosine representation. Although the latent function is continuous on its output, we faced the challenge of learning two separate functions.

As another key insight to the problem of motion prediction, we observed that only few parts of a dynamic object may appear to be moving. This fact lead to two problems. Firstly, only few of the collected samples contained information about the movement of the object. Secondly, we had to find a measure for deciding which samples are suitable for motion prediction and which are not. To tackle the first problem we used importance sampling. A sample set was selected based on importance weights computed from the occupancy change modeled by a difference map patch. The second problem was tackled using Gaussian process regression which results in a mean prediction and an estimation of the variance.

Because of the high dimensional data and the number of training samples used, we had to rely on an approximation of a Gaussian process, the sparse GP with pseudo-inputs. To better model the noise function a heteroscadestic extension of the sparse GP was applied also. A prediction of the motion of a dynamic object is obtained merging all the single predictions for the points of a pointcloud cluster based on their predictive variance.

In Chapter 4 we demonstrated that using map patches as local features, our approach results in a competitive overall performance (see 4.5), compared to the state of the art methods ICP and Kalman filtering (based on the constant velocity motion model). The angle prediction using the Gabor features did not show such a good result. However, considering the low number of samples that could be used as well as the high-dimensionality and challenging distribution of the data, the accuracy obtained is remarkably high.

Our approach offers several advantages, although it does not perform better then state of the art methods in every situation:

- Only local motion features are used. A modeling of possibly occurring object classes is not needed.
- A data association based on matching objects or finding finding point-to-point correspondences between laser measurements is not needed. Instead we use a grid map to solve the data association problem. The experiments showed that especially in situations in which the number of measurements ending on the tracked object changes significantly or only few measurements are available in general, the ICP algorithm ran into problems. Our approach instead does not depend on the number of measurements as much.
- Our approach can efficiently be implemented in parallel which, for example, is not possible for the ICP algorithm. Each of the single predictions can be computed independent of each other. This is an advantage when implementing such a method in hardware, for example.

In addition to estimating motion directly, the GP approach could also be used to improve the result of the existing methods. One possible application could be improving the motion model of the Kalman filter using the speed prediction of the GP. Another application is a better initialization of the ICP transformation. A better initialization would reduce the probability that the ICP algorithm converges to a local maximum solution and lead to a better performance of ICP.

In general, predicting motion based on local features is indispensable for a operating autonomous agents robustly in a dynamic environment. While our approach already shows good results, improvements are possible. Some ideas for future research are given in the next chapter.

6 Future Work

Despite the promising results, there are possible improvements in several areas like feature extraction and learning. The better generation of features could aid an improved mapping. Reliably extracting the ground plane from the scans could result in better and more stable features. In principle, a laser with a higher frequency should also result in better features. The higher the turning rate the smaller the map patches can be. Smaller map patches means a smaller dimensionality and problem size. In addition, the patches become more independent of the structure of the underlying object.

Additionally to using map patches as a feature for regression, dimensionality reduction techniques could be applied. In this thesis, Gabor filters were used to reduce the dimensionality but their usage did not show a satisfactory result. Other, more general dimensionality reduction techniques that could be applied include the Principal Component Analysis (PCA) which reduces the dimension such that the information that explains the variance of the data is kept. The Canonical Correlation Analysis (Hotelling [1936]) seems to be a promising approach, too. It finds a projection of both, the multi-dimensional input and output variables such that their correlation is maximized.

As we have seen in Chapter 3 the data set used for learning is heteroscedastic but the noise function does not yet seem to be learned correctly. The SPGP-HET seems to be very prone to overfitting the noise function. Thus, the noise function could be learned using the approach by Kersting *et al.* [2007] which was also introduced in Section 3.1.3. This approach uses a second GP to learn the noise function and might not be as susceptible to overfitting. While the individual noise parameters of the SPGP-HET can be seen as modeling a non-stationary noise function, the approach by Kersting *et al.* has the advantage that the covariance function governing the noise function can be chosen arbitrarily. Choosing a smoothly changing covariance function, as the squared-exponential function, might result in a estimated noise function that does not to overfit as much. Another reason for errors in the estimation of the noise function, might be the sample selection method applied. By selecting training samples based on importance weights the number of samples with a high variance decreases. One solution could be using a higher number of randomly selected training samples. However, considering the high dimensionality of the data this solution is not feasible.

Although the SPGP reduced the computational complexity, the number of parameters to be learned increased largely because of the pseudo-inputs. Therefore, although learning the pseudo-inputs seems a good choice if the number of dimensions is low, learning them in a high-dimensional space is difficult. Other GPs that are based on using inducing inputs, e.g., using a subset of the training data (see Section 3.1.3), should be considered instead.

Bibliography

[Arun *et al.*, 1987] K.S. Arun, T.S. Huang, and S.D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698–700, 1987.

[Besl and McKay, 1992] P.J. Besl and N.D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[Bobruk and Austin, 2004] J. Bobruk and D. Austin. Laser motion detection and hypothesis tracking from a mobile platform. In *Proc. of the Australian Conference on Robotics & Automation (ACRA)*, 2004.

[Bresenham, 1965] J. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[Candela and Rasmussen, 2005] J.Q. Candela and C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

[Castro *et al.*, 2004] D. Castro, U. Nunes, and A. Ruano. Feature extraction for moving objects tracking system in indoor environments. In *Proc. of the Symposium on Intelligent Autonomous Vehicles (IAV)*, 2004.

[Cook *et al.*, 2005] J. Cook, V. Chandran, S. Sridharan, and C. Fookes. Gabor filter bank representation for 3d face recognition. In *Proc. of the International Conference Digital Image Computing (DICTA)*, page 4, 2005.

[Dellaert and Thorpe, 1997] F. Dellaert and C.E. Thorpe. Robust Car Tracking using Kalman Filtering and Bayesian Templates. *Proc. of the International Conference on Intelligent Transportation Systems (ITSC)*, 3207:72, 1997.

[Di Stefano and Viarani, 1999] L. Di Stefano and E. Viarani. Vehicle Detection and Tracking Using the Block Matching Algorithm. *Recent Advances in Signal Processing and Communications*, 1999.

[Elfes, 1987] A. Elfes. Occupancy grids: A probabilistic framework for robot perception and navigation. *Journal of Robotics and Automation*, 3(3):249–265, 1987.

[Ferryman *et al.*, 1998] J.M. Ferryman, A.D. Worrall, and S.J. Maybank. Learning enhanced 3d models for vehicle tracking. In *Proc. of the British Machine Vision Conference (BMVC)*, 1998.

[Fischler and Bolles, 1981] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[Fletcher and Reeves, 1964] R. Fletcher and C.M. Reeves. Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154, 1964.

[Florek *et al.*, 1951] K. Florek, J. Lukaszewicz, H. Perkal, H. Steinhaus, and S. Zubrzycki. Sur la liaison et la division des points d'un ensemble fini. *Colloquium Mathematicum*, 2:282–285, 1951.

[Fuerstenberg *et al.*, 2002] K.C. Fuerstenberg, K.C.J. Dietmayer, and V. Willhoeft. Pedestrian recognition in urban traffic using a vehicle based multilayer laserscanner. In *Proc. of the IEEE Intelligent Vehicles Symposium (IV)*, volume 1, pages 31–35, 2002.

[Geyer, 1992] C.J. Geyer. Markov chain monte carlo maximum likelihood, 1992.

[Goldberg *et al.*, 1998] P.W. Goldberg, C.K.I. Williams, and C.M. Bishop. Regression with Input-dependent Noise: A Gaussian Process Treatment. *Advances in Neural Information Processing Systems*, pages 493–499, 1998.

[Haag and Nagel, 1999] M. Haag and H.H. Nagel. Combination of edge element and optical flow estimates for 3d-model-based vehicle tracking in traffic image sequences. *International Journal of Computer Vision*, 35(3):295–319, 1999.

[Hähnel *et al.*, 2003] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1557–1563, 2003.

[Hotelling, 1936] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28:321–377, 1936.

[Kersting *et al.*, 2007] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. Most likely heteroscedastic gaussian process regression. In *Proc. of the International Conference on Machine Learning (ICML)*, Corvallis, Oregon, USA, March 2007.

[Lance and Williams, 1967] G.N. Lance and W.T. Williams. A general theory of classificatory sorting strategies: 1. hierarchical systems. *The Computer Journal*, 9(4):373, 1967.

[Lawrence *et al.*, 2002] N.D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse gaussian process methods: The informative vector machine. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 609–616, 2002.

[Lu and Milios, 1997] F. Lu and E. Milios. Robot Pose Estimation in Unknown Environments by Matching 2D Range Scans. *Journal of Intelligent and Robotic Systems*, 18(3):249–275, 1997.

[Luber *et al.*, 2008] M. Luber, K. Arras, C. Plagemann, and W. Burgard. Tracking and classification of dynamic objects: An unsupervised learning approach. In *Robotics: Science and Systems (RSS)*, 2008.

[MacKay, 1998] D.J.C. MacKay. Introduction to gaussian processes. In *Neural Networks and Machine Learning*, NATO ASI Series, pages 133–166. Kluwer Academic Press, 1998.

[MacLachlan and Mertz, 2006] R. MacLachlan and C. Mertz. Tracking of Moving Objects from a Moving Vehicle Using a Scanning Laser Rangefinder. In *Proc. of the International Conference on Intelligent Transportation Systems (ITSC)*, pages 301–306, 2006.

[MacQueen, 1967] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, 1967. University of California Press.

[Morris *et al.*, 2008] D.D. Morris, P. Haley, W. Zachar, and S. McLean. Ladar-based vehicle tracking and trajectory estimation for urban driving. In *Proc. of the Association for Unmanned Vehicle Systems International (AUVSI)*, San Diego, June 2008.

[Navarro-Serment *et al.*, 2008] L.E. Navarro-Serment, C. Mertz, N. Vandapel, and M. Hebert. Ladar-based pedestrian detection and tracking. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, May 2008.

[Petrovskaya and Thrun, 2008] A. Petrovskaya and S. Thrun. Model based vehicle tracking for autonomous driving in urban environments. In *Robotics: Science and Systems (RSS)*, 2008.

[Prassler *et al.*, 2000] E. Prassler, J. Scholz, and A. Elfes. Tracking multiple moving objects for real-time robot navigation. *Journal of Autonomous Robots, Special Issue on Perception for Mobile Agents*, 8, 2000.

[Rasmussen and Williams, 2005] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2005.

[Rusinkiewicz and Levoy, 2001] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proc. of the International Conference 3D Digital Image Modeling (3DIM)*, pages 145–152, June 2001.

[Sabri and Fieguth, 2004] M. Sabri and P.W. Fieguth. A new gabor filter based kernel for texture classification with svm. In *Proc. of the International Conference on Image Analysis and Recognition (ICIAR)*, pages 314–322, 2004.

[Sibson, 1973] R. Sibson. Slink: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

[Snelson and Ghahramani, 2006a] E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, volume 18, pages 1257–1264. MIT Press, 2006.

[Snelson and Ghahramani, 2006b] E. Snelson and Z. Ghahramani. Variable noise and dimensionality reduction for sparse gaussian processes. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2006.

[Spies *et al.*, 2002] H. Spies, B. Jahne, and J.L. Barron. Range flow estimation. *Computer Vision and Image Understanding*, 85(3):209–231, 2002.

[Talukder *et al.*, 2003] A. Talukder, S. Goldberg, L. Matthies, and A. Ansar. Real-time detection of moving objects in a dynamic scene from moving robotic vehicles. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 2, 2003.

[Thrun *et al.*, 2005] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. MIT Press, 2005.

[Thrun, 2001] S. Thrun. Learning occupancy grids with forward models. *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 3, 2001.

[Triebel *et al.*, 2006] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[Wang *et al.*, 2007] C.C. Wang, C. Thorpe, M. Hebert, S. Thrun, and H. Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research*, 26(6), June 2007.

[Williams and Rasmussen, 1996] C.K.I. Williams and C.E. Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems 8*. MIT Press, 1996.

[Zhao and Thorpe, 1998] L. Zhao and C. Thorpe. Qualitative and Quantitative Car Tracking from a Range Image Sequence. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society Press, 1998.

[Zhu *et al.*, 2004] E. Zhu, J. Yin, and G. Zhang. Fingerprint enhancement using circular gabor filter. In *Proc. of the International Conference on Image Analysis and Recognition (ICIAR)*, pages 750–758, 2004.