Thesis for the Degree of Licentiate of Engineering

Localization for Autonomous Vehicles

Erik Stenborg

CHALMERS

Department of Electrical Engineering
Chalmers University of Technology
Göteborg, Sweden 2017

Localization for Autonomous Vehicles
Erik Stenborg

Technical report number: R012/2017
ISSN 1403-266X

Department of Electrical Engineering
Signal Processing Group
Chalmers University of Technology
SE–412 96 Göteborg, Sweden

Typeset by the author using LaTeX.

Chalmers Reproservice
Göteborg, Sweden 2017

Abstract

There has been a huge interest in self-driving cars lately, which is understandable, given the improvements it is predicted to bring in terms of safety and comfort in transportation. One enabling technology for self-driving cars, is accurate and reliable localization. Without it, one would not be able to use map information for path planning, and instead be left to solely rely on sensor input, to figure out what the road ahead looks like. This thesis is focused on the problem of cost effective localization of self-driving cars, which fulfill accuracy and reliability requirements for safe operation.

In an initial study, a car equipped with the sensors of an advanced driver-assistance system is analyzed with respect to its localization performance. It is found that although performance is acceptable in good conditions, it needs improvements to reach the level required for autonomous vehicles. The global navigational satellite system (GNSS) receiver, and the automotive camera system are found to not provide as good information as expected. This presents the opportunity to improve the solution, with only marginally increased cost, by utilizing the existing sensors better.

A first improvement is regarding global navigational satellite systems (GNSS) receivers. A novel solution using time relative GNSS observations, is proposed. The proposed solution is tested on data from the DriveMe project in Göteborg, and found capable of providing highly accurate time-relative positioning without use of expensive dual frequency receivers, base stations, or complex solutions that require long convergence time. Error introduced over 30 seconds of driving is found to be less than 1 dm on average.

A second improvement is regarding how to use more information from the vehicle mounted cameras, without needing extremely large maps that would be required if using traditional image feature descriptors. This should be realized while maintaining localization performance over an extended period of time, despite the challenge of large visual changes over the year. A novel localization solution based on semantic descriptors is proposed, and is shown to be superior to a solution using traditional image features in terms of size of map, at a certain accuracy level.

List of appended papers

Paper I M. Lundgren, E. Stenborg, L. Svensson and L. Hammarstrand, **"Vehicle self-localization using off-the-shelf sensors and a detailed map"**, 2014 IEEE Intelligent Vehicles Symposium, Proceedings pages 522-528.

Paper II E. Stenborg and L. Hammarstrand, **"Using a single band GNSS receiver to improve relative positioning in autonomous cars"**, 2016 IEEE Intelligent Vehicles Symposium, Proceedings pages 921-926

Paper III E. Stenborg, C. Toft and L. Hammarstrand, **"Long-term Visual Localization Using Semantically Segmented Images"**, Submitted to 2018 IEEE International Conference on Robotics and Automation

Preface

More than four years ago, when I was about to start this project, I thought, "Localization is almost solved, and what little remains ought to be easy! Five years is way too much time allocated - what am I going to do after three years when the problem is solved, and the DriveMe cars are roaming the streets of Göteborg with my code in them?". Now, almost five years later, I'm very happy that I've reached this half-way milestone, that the Licentiate thesis marks, and I wonder if I'm ever going to make a dent in the mountain of work that remains until the problem really is solved.

I have learned a lot, though. I have learned that research takes time. I have learned that the path to success is full of failures that nobody sees, but the person who makes them. I have learned that all those failures is part of the process. And I have learned to appreciate the advice from those who have gone through the process, and emerged successfully on the other end, a little bit wiser than the rest of us who are in the middle of it.

I would like to show my gratitude towards my academic supervisors Lennart Svensson and Lars Hammarstrand for your advise, for your unwavering support, for our interesting discussions about both professional and other matters, and for all help I have received to reach this milestone. A very big thanks also to my industrial supervisor Joakim Sörstedt and my manager Jonas Ekmark for giving me this opportunity, for your enthusiasm, for your support of my work, and for believing in me. I would also like to thank Vinnova, Volvo Car Corporation and Zenuity for financing my work.

Thanks to all friends and colleagues (both in the past and the present) in the signal processing and the image analysis groups at Chalmers, at the active safety department at Volvo and at Zenuity. Lunches and fika wouldn't be nearly as entertaining without you. Special thanks to Malin, for helping me getting started when I first joined the group at Chalmers and for co-authoring the very first published paper with my name on it, to Carl for being such a rock and helping me with my present work, to Anders for being a good room mate and accepting my quirks (I apologize for shooting you the first day we met). Special thanks also to Karl, Abu, Maryam, Daniel, Samuel, Erik, Yair, and all the rest of you already mentioned, for proof reading or otherwise helping me with this thesis and the papers in it.

A warm thanks to my parents. Without you, I literally wouldn't be here today. Finally, I would like to extend a very warm thanks to my beloved wife, Yoko, and to my lovely daughters, Lina and Hanna, for being such a wonderful family. I love you.

Erik Stenborg
Göteborg, 2017

Contents

Part I

Introductory Chapters

Chapter 1

Introduction

Autonomous vehicles have several advantages to vehicles which must be driven by humans, one being increased safety. Despite an increase in number of cars and total distance driven, traffic injuries has declined in Sweden [1] and the USA [2], from being the leading cause of death in age groups below 45, to "merely" being in the top ten [3]. To continue this trend, we increase the scope of traffic safety from just protecting passengers in case of an accident, to mitigating or preventing accidents before they happen. Based on studies such as [4], where it is noted that a vast majority of accidents are caused by human error, we can see that the largest potential to further increased safety, lies in letting technology assist drivers by automating the driving task, and thus creating self-driving cars.

Other reasons for self-driving cars are economy, as people can be more productive while on the road; equality, as blind and otherwise impaired people then can ride by themselves; and also environment, since the need for parking in cities could be reduced if all cars could leave by themselves after the rider has reached the destination.

Now, let us look into the problems we need to solve to make autonomous vehicles available. These problems include human factors, economy, legal liability, equality, morality, etc. This thesis focuses on technical problems, specifically those regarding precise localization. Economy and safety restrict possible solutions, but apart from that, we can view the problem of localization independently. The technical problem of autonomous driving can be described as a function which maps sensor data to control signals for the car, primarily wheel torque for longitudinal acceleration and steering torque for turning the car. The rest which needs controlling in the car, e.g. controlling flow of fuel and air to the engine, is already solved.

There are several possible ways in which we can approach this problem of mapping sensor data to control outputs. One possibility is to decide on a sensor setup that seems reasonable, e.g., a set of cameras based on the fact

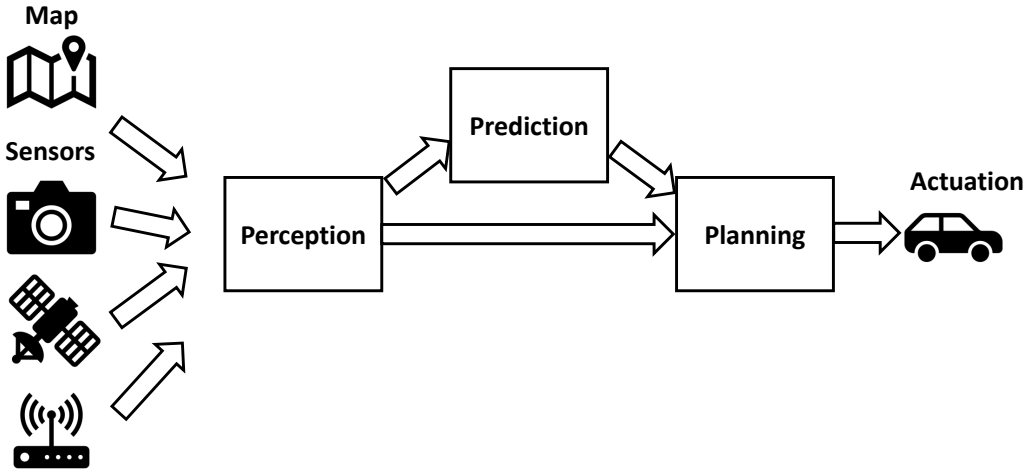

Figure 1.1: Flow of data and processing modules for autonomous driving. Localization is here considered part of the perception module.

that humans are able to drive a car mainly based on vision input, and then treat it as a machine learning problem, using either reinforcement learning or supervised learning with a human expert driver. This approach has so far rendered some success [5–7], but most larger scale projects are focusing on more modular approaches where the driving task is divided into smaller parts, which can be designed and verified more independently of each other.

In the modular approach, the problem is subdivided into a few major modules, where Figure 1.1 shows one example of such a subdivision. We have one module which is responsible for planning a trajectory and following it based on information from the other modules, another which interprets the sensor inputs into a simple structure that is meaningful for the driving task, and a third which predicts what other traffic participants will do. The output from the perception block will typically contain information about both the dynamic environment, such as position and velocity of other road users, and the static environment, such as which areas are drivable and which contain static obstacles.

When it comes to describing the static environment, there are two paradigms. One is to rely only on the input provided by the sensors on the own car, and the other is to rely also on a predefined map and relating the sensor inputs to that map. The map paradigm will provide more information about the static environment than what is visible using only the sensors, and is thus often preferred. However, to use a map, one must be able to localize the car in the map. Any uncertainty in position and orientation in relation to the map will translate into uncertainty about the drivable area and in consequence also the planned path. Thus, we can see that ac-

curate localization in relation to an accurate map of the close surroundings of the car, is an enabler for autonomous driving.

Then what is needed to solve effective localization for autonomous vehicles? Firstly, we need to understand what is needed in terms of accuracy, availability and price. These requirements are far from obvious, and could warrant a thesis in and by itself, but a short description of how one can reason about it is included in Chapter 2. Secondly, we need a map that connects the observable landmarks with the possibly unobservable, static features of the road that are needed for path planning, i.e. drivable surface, static obstacles, etc. A few thoughts on this map have also been included in Chapter 2. Thirdly, we need sensors which are capable of observing the landmarks, and sensor models which describe how the observations relate to the physical world. In Chapter 3, radars are briefly described, while cameras and global navigation satellite systems such as GPS, are described in further detail, with a focus on how they can be used for localization purposes. Lastly, we need algorithms that combine the given maps and the measurements into estimates of position and orientation. This is achieved using the Bayesian estimation framework described in Chapter 4, where we see how to combine the models of various sensors with a motion model for the vehicle to arrive at probabilistic models of position and orientation.

This thesis examines what level of localization accuracy that can be achieved using different types of automotive classified sensors. It further establishes a few missing pieces in reaching desired performance, and proposes solutions for some of them. This is done in three separate papers, briefly summarized in Chapter 5.

Chapter 2

Localization

As concluded in the introduction, accurate localization with respect to a map is a key enabler for using information from those maps in the motion planner. In this chapter we start with a brief overview of the research area, then have a look at what goes in the map, then look at how to use the map for localizing with respect to the drivable area, and finish by reasoning about how to derive requirements on accuracy and availability.

1 Overview

If we look at the problem of localization with a given map in a slightly larger context than for autonomous vehicles, we realize that it can be formulated in many different ways. We can roughly categorize problems in four categories, based on two criteria: metric vs. topological, and single shot vs. sequential. In Table 2.1, the references given below are categorized in either of the four categories.

The first division between metric or topological localization is regarding the result of the localization. Topological localization is when the result of localization is categorical and can be represented as nodes in a graph, e.g., a certain room in a house, or the location where an image from a training data set was taken. Examples here include place recognition by image retrieval methods, where an image database of geo-tagged images is used to represent possible locations, and then a query image is compared to the images in the database, and the most similar image along with its position is retrieved, see e.g., [8, 9]. Specialized loop closure detectors also belong in this category, see e.g., [10, 11]. These provide information to a larger system doing simultaneous localization and mapping (SLAM), when a robot has roughly returned to a previously visited place just by looking at image similarities. In this category, there are also some solutions which

	topological	metric
single shot / global	[8–11]	[15–17, 19, 20]
sequential / with prior	[12–14]	[21–23, 27, 28]

Table 2.1: Classification of localization problems with a few examples of proposed solutions.

focus on robust localization in conditions with large visual variations, see e.g., [12, 13]. They make use of sequences of images which as a group should match a sequence of training images in an image similarity sense. One could also argue that some hybrid methods, such as [14], belong in this category, even though they claim to be "topometric" in the sense that they provide interpolation between the nodes in the graph and thus give metric localization result along some dimensions where a whole array of training images has been collected.

Metric localization is more geometric in nature. The resulting location is in a continuous space, and most easily expressed in coordinates using real numbers. There are examples from the computer vision community that does direct metric localization using a single query image, see e.g., [15–18], but also examples where a form of topological localization is performed first as an initial step, see e.g., [19, 20]. Most localization in robotics, with the purpose of providing navigational information to a robot, requires metric localization, with some examples in [21–23]. Localization using global satellite navigational systems and inertial measurement units are also examples of metric localization.

The second division we have when categorizing localization, is between single shot localization and sequential localization. This is regarding what type of information is used for the localization. Single shot localization uses only one observation and no additional information from just before or after. This is relevant when using single images, see e.g., [8, 9, 11, 15–20], and sometimes also when using sensors that are specialized for localization, such as the global navigation satellite systems, see e.g., [24].

Sequential localization means using a sequence of observations and some motion model to connect them, or alternatively, to have useful prior knowledge on the location from some other source. This is the type of observations available for on-line robot navigation, see e.g., [12, 13, 21–23, 25–28]. We note that the type of localization needed for autonomous vehicles falls in the intersection of sequential localization and metric localization. Thus, the rest of the thesis will focus on metric, sequential localization.

Localization for autonomous vehicles has been done for quite some time using expensive sensor setups, including survey grade GNSS receivers, fiber

optical inertial measurement units and multi beam rotating laser scanners. Some of the most well known examples are the contributions to the DARPA Grand Challenge in 2005, see e.g., [25], and the DARPA Urban Challenge in 2007, see e.g., [26]. Some spin-offs from these DARPA challenges, such as Waymo, use similar technology, arguing that the cost of sensors will fall enough to make it viable for consumers in a near future. Others argue that more simple sensors should be enough for accurate and robust localization, but this is yet to be shown in practice. This cost constraint is the reason that this thesis is focused on solutions using sensors that are expected to be readily available in future cars.

Localization with a given map, and mapping given known positions and orientations, are considered subproblems to the more general problem of simultaneously estimating both location and map (SLAM). See e.g. [29, 30] for a survey of the SLAM problem. Although localizing with a given map would be sufficient for an autonomous vehicle, there is still a need to build the map and keep it updated. To keep costs down, this should should be performed without huge fleets of special mapping vehicles using very expensive positioning systems. Thus, viewing both the localization and mapping problems as SLAM problems is attractive from economical reasons, in spite of it being slightly more complicated.

In the 1990's, SLAM was usually solved using extended Kalman filters, see e.g., [31, 32], where the state vector described the location of landmarks of the map and the most recent robot location. There were both convergence and scalability issues with this approach, due to a fixed linearization point, and an ever growing landmark covariance matrix that needed inverting in every update. In 2002, solutions using particle filters [33] were presented. They provided better performance for larger problems, due to not having to encode the correlation between landmarks, since they are conditionally independent given the robot trajectory. A few years later the trend shifted towards viewing the SLAM problem as a continuously growing smoothing problem, as in e.g., GraphSLAM [34], and iSAM [35, 36]. This is still how it is typically solved today, see e.g. [37, 38] for some more recent work. For people with a background in computer vision rather than robotics, it could be worth noting that, when using point features from camera images in GraphSLAM or iSAM, the solution is virtually identical to continuously solving a growing bundle adjustment problem with optimization methods that consider the sparse and incremental nature of the problem [39, 40].

2 Map representation

As previously noted, the problem in this thesis is self localization for autonomous vehicles using automotive grade sensors, and a map. For this we need a map, and although we have hinted that this can be solved using SLAM, it could be useful to discuss what to put in the map.

We know that the path planner needs some navigation information, e.g., drivable area, lane connectivity, traffic rules, suitable speed profiles, etc., to do its job. This navigational information is connected to positions in the world through markings on the ground, and traffic signs, which can be observed by a forward looking camera. These cues are what we would like to localize with respect to. However, some sensors, e.g., cameras looking backwards, radars, or GNSS receivers, can not directly observe them. Instead we may choose to use some other observable landmarks, and encode their relation to the navigational information in the map. With a map that holds both the navigational information needed by the path planner, and observable landmarks, the car can use on-board sensors to detect angle and/or range to the landmarks, and triangulate a position relative to the navigational information in the map.

Now, these observable landmarks that we choose to add to the map can be quite sensor specific. Although geometric features, such as points or curves, are a popular choice, there are plenty of other options. One alternative is a grid map, where the world is discretized in a 2-D or 3-D grid, where each pixel (or voxel in the 3-D case) describes some property about that small part of the world. An often used property is occupancy, i.e., if the grid cell is transparent or opaque to the relevant sensor, often indicating that the cell is empty or occupied, hence its name [41, 42]. Another property that has been used successfully with grid maps is reflectivity, as in e.g., [43]. A version of grid mapping that has been popularized with the introduction of the Kinect RGB-D camera, is the truncated signed distance function. Here, each voxel stores the signed distance to the nearest opaque surface, if it is below some threshold distance. In indoor environments, where the size of the map is quite limited, 3-D grid maps have been demonstrated with impressive results. However, a grid representation of the world scales badly with the number of dimensions to be described, and although work has been done to compress the 3-D grids maps, see e.g., [44], most often this approach is used for 2-D maps.

For large 3-D maps, a more common approach is to store sparse features that are possible to describe with a few parameters. Points, lines, and B-splines are popular, but not the only possible choices. When using these types of features, we should take care to construct feature detectors that

are able to reliably detect these features in the sensor data.

Reference frame

For the motion planner, the only relevant information needed is the relative position from ourselves to other traffic participants, and to relevant objects in the immediate surroundings. Where the origin of the metric map of the world lies is irrelevant, as long as all relative positions and angles are correct. For most sensors, especially the cameras and radars used in this thesis, this is also true. However, GNSS provides global localization with respect to a certain origin of the Earth, and thus, if we want to use GNSS to localize in the map, the map must be aligned to the world origin as defined by the coordinate system used in GNSS.

Historically, maps have been thought of as 2-D projections of the surface of the Earth to a flat surface, such as a piece of paper or a screen. It is mathematically impossible to map the surface of a sphere to a plane without distortions and discontinuities, but there has been many proposals for suitable projections that does this without too much distortion, or with some special property preserved at the expense of some other error being introduced.

For our needs, however, we can drop the requirement that the map must project well to a 2-D plane. We are merely interested in using the map for localization and path planning, and for that we need to keep track of 3-D position of landmarks, and the road. The map must at least be able to store a 3-D representation of landmarks and the road, in any part of the world without discontinuities.

The most straight forward reference frame for a global 3-D map is an Earth centered and Earth fixed (ECEF) Cartesian coordinate frame that follows the rotation of the Earth. In this coordinate system there are no problems to map even Scott-Amundsen station at the south pole without discontinuities, and the transformation to a local east-north-up (ENU) frame is a simple rotation and translation. In the experiments in this thesis, however, the area where the car is driving is small enough, that a map in the local ENU frame, with a flat Earth assumption, is sufficient.

3 Accuracy requirements

Now, let us look into what performance we require of our localization. What accuracy of localization is needed for autonomous driving? How often and long can the localization be allowed to have an error over a certain threshold, and what is that threshold? These are difficult questions to answer, but in

order to evaluate performance, it is important to at least have an idea of the magnitudes, and a method to argue about them.

One possible starting point is a safety analysis. There are tools and frameworks to aid in such analysis, and they help us through hazard analysis and risk assessment to identify critical scenarios and the safety goals that arise from them. The hazards are classified according to their risk, which leads to a classification also of the safety goals. The ISO standard 26262 defines five Automotive Safety Integrity Levels (ASIL): QM, A, B, C, and D, with D being the highest level. The classification is based on how frequent the situation is (exposure), how severe the damages would be, and how controllable the situation is by the driver.

Let us take an example with the hazard of drifting out of the lane in a curve and colliding with oncoming traffic at 20 m/s or higher when in autonomous mode. This hazard is very severe in terms of expected damages to the people involved in a head on collision, the situation is quite frequent (high exposure), and it has low controllability since the driver is not expected to be alert when the car is in autonomous mode. Thus, it gets the highest classification, ASIL D, which roughly translates to a desired frequency of error of less than 1 in 1000 years. Next, we can split the hazard into a fault tree, which should contain all possible faults that could cause it. There can be several possible causes, e.g., skidding if there is ice, not getting enough steering torque from the power steering, or lane information reported to the path planner being wrong. If the lane information is based only on using a map and localization, with ASIL D, both the map and the localization are required to give a lateral error that is small enough such that the vehicle does not cross over to the other lane ($\sim$1 m).

This margin applies to the combined error of all components in the autonomous car, from sensing to actuation. To get the requirements for a specific module in the chain of modules presented in Figure 1.1, one could subtract the error introduced by the last module of the chain, and then propagate the remainder backwards through the various transformations that each block causes. Localization resides in the perception module, early in the chain, and this type of backwards propagation of permissible error could be quite cumbersome. A slightly less difficult way to get a rough understanding of the error margins, could be to simulate errors in the localization and propagate them forward to see what the final control error will be.

Chapter 3

Sensors and observations

When self-localization of cars is the topic, people often come to think of a Global Navigation Satellite System (GNSS), often the Global Positioning System (GPS), which has as its main purpose to measure position of the receiver. However, as we have seen, there are also other sensors, primarily used for other purposes, which can be quite useful also for localization. Cameras and radars, as examples of this type of sensors, are primarily used to detect obstacles and other road users in the local environment around the car, and often more than one of each is required to get a good enough understanding of the surroundings. Together with GNSS, and motion sensors such as accelerometers, gyroscopes and wheel speed sensors, they provide the measurements that we use for localization.

So, how do these sensors work, and how do we use them for localization? One common factor is that they capture electromagnetic waves (light or radio) that was either transmitted or reflected by some object in the environment. They typically record the angle of arrival, or measure distance to the object by recording the time delay of the captured signal. Sometimes the intensity, and possibly some other features of the signal are also recorded. With sensors measuring angle to landmarks, the known positions of the landmarks can be used to triangulate the ego-position, see Figure 3.1. With sensors measuring distance, trilateration is used, see Figure 3.2. In both cases more than one landmark has to be detected to calculate an unambiguous position. In this chapter we look into more details on how the sensors used for localization in this thesis work, and how the measurements are modeled.

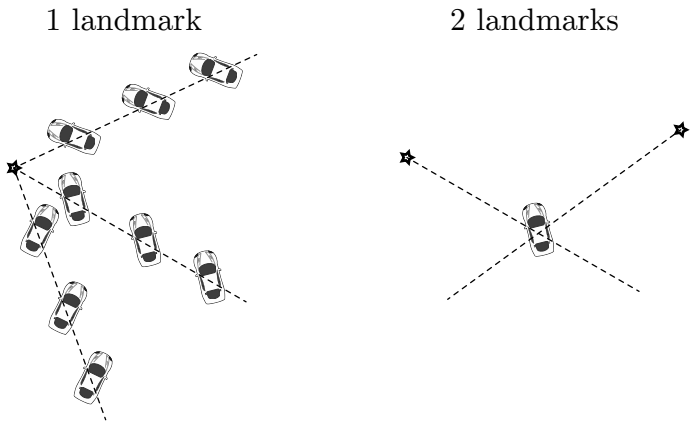

Figure 3.1: Triangulation from landmarks with known positions. With only one landmark, there is no unique solution, but with two landmarks we get a unique solution in 2-D.

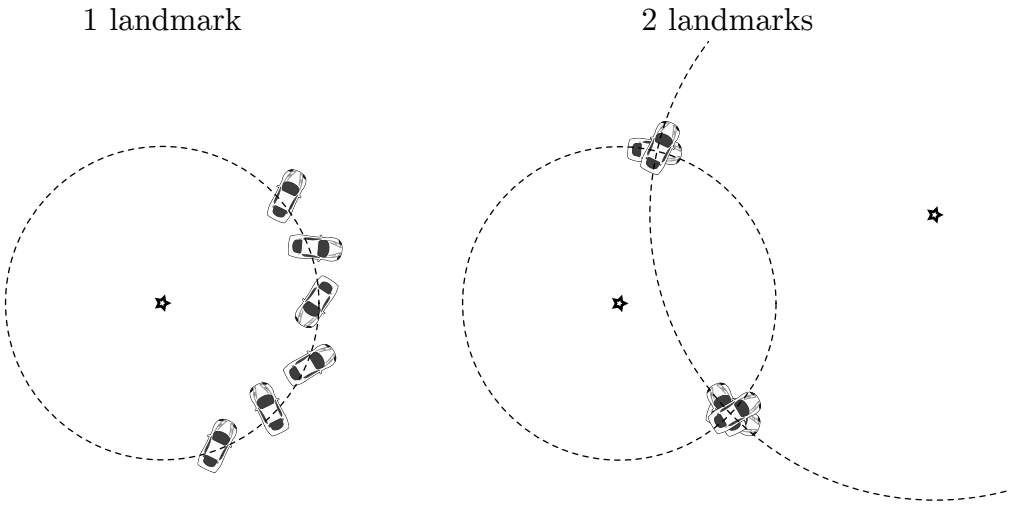

Figure 3.2: Trilateration from landmarks with known positions. With only one landmark, again there is no unique solution, and with two landmarks we get two possible positions in 2-D, but no information about orientation.

1 Global Navigation Satellite System

GNSS is, in a localization context, a rather unique sensor compared to other sensors. Its sole purpose is localization, while cameras and radars are primarily used in the self-driving car for their ability to detect obstacles and other road users. GNSS is also the only sensor which has a global frame of reference.

All GNSS systems, including GPS, which is the most well known, use the same principle of positioning. Kaplan and Hegarty describe this in detail in their book [45]. Essentially, a number of satellites with accurately known Earth orbits, transmit signals at very well defined times, and the receivers measure the delay from the transmission to the reception. From this delay the receiver can calculate the range to the satellite. By using multiple simultaneous satellite observations, the receiver is able to trilaterate its position.

Coarse/acquisition (C/A) code ranging

The signal that is transmitted from the satellite consists of a carrier signal, on top of which there are up to two other signals, modulated using Binary Phase Shift Keying. One of the signals is a repeating sequence of pseudo random numbers, which functions as a unique identifier for the satellite, and is also what is used for range calculations. The other signal, which is transmitted at a slower bit rate, is the navigational message. It provides information about the satellite such as its status, orbital parameters, corrections to its clock, etc.

The normal method for determining position using GPS uses the sequence of pseudo random numbers to calculate a pseudo range from the receiver to the satellite. Since the sequence of pseudo random numbers (C/A code) is known in advance by the receiver, this can be done with a tracking loop that calculates how much a local copy of the C/A code must be time shifted to match the received signal, see Figure 3.3. If the receiver had a perfect clock, and if smaller error sources are ignored, one could simply multiply the time shift with the speed of light to get the distance to the satellite. Given three such measurements to different satellites, the 3-D position of the receiver could be trilaterated. However, the clock in a typical receiver is of relatively low quality, which is why the receiver time must also be treated as an unknown variable. That increases the requirement to four satellites, in order to calculate a solution for position and time. This is the basic idea of how simple positioning using the C/A code in GPS works. If a receiver does not make use of the carrier phase to smooth the pseudo range measurements from the C/A code, there is a noise floor of around 1% of

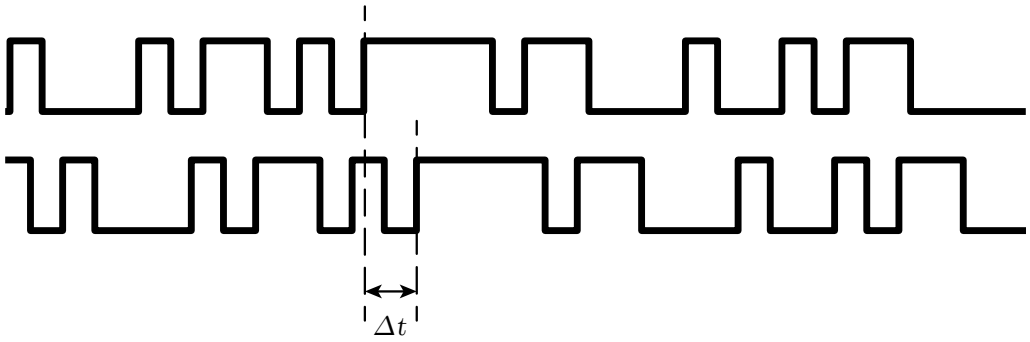

Figure 3.3: The time offset, Δt, when matching pseudo random numbers in the received C/A code signal to the internally generated sequence, is unique.

the bit length in the code, which is difficult to get below. The bit length is calculated as the bit length in seconds times the speed of light, and is around 300m. This means that the receiver tracking loop introduces noise on the pseudo range measurements that has a standard deviation of about 3m. However, since modern receivers use some tricks, such as carrier phase smoothing [46], the receiver noise is usually quoted as much lower today, see Table 3.1. Apart from the receiver tracking noise, there are other errors affecting the pseudo range measurements, leading to an average error that is typically quoted as around 7 m, for a standard single frequency receiver.

Carrier phase ranging

Now, consider the carrier signal, which has a wavelength of about 0.2 m; much shorter than the bit length of the C/A code. If the tracking loops in the receiver are able to determine the phase of the carrier signal within 1%, and if there is a way to remove additional errors, then potentially a very accurate range measurement to the satellite can be obtained.

The alignment to the carrier phase through phase lock loops in the receiver, is indeed accurate to about 1 mm. However, there is now another problem. Every cycle of the carrier signal looks the same. This means that if we examine the cross correlation between the received signal and the internal oscillator, we would find an infinite number of peaks with a spacing that corresponds to the wavelength of the carrier signal. Which peak the phase lock loop locks on to, is random. This causes the range measurement using the carrier phase, to be offset by an unknown integer N times the wavelength. This integer, N, must be determined if we want to use the carrier phase range measurement as a normal range measurement, see Figure 3.4.

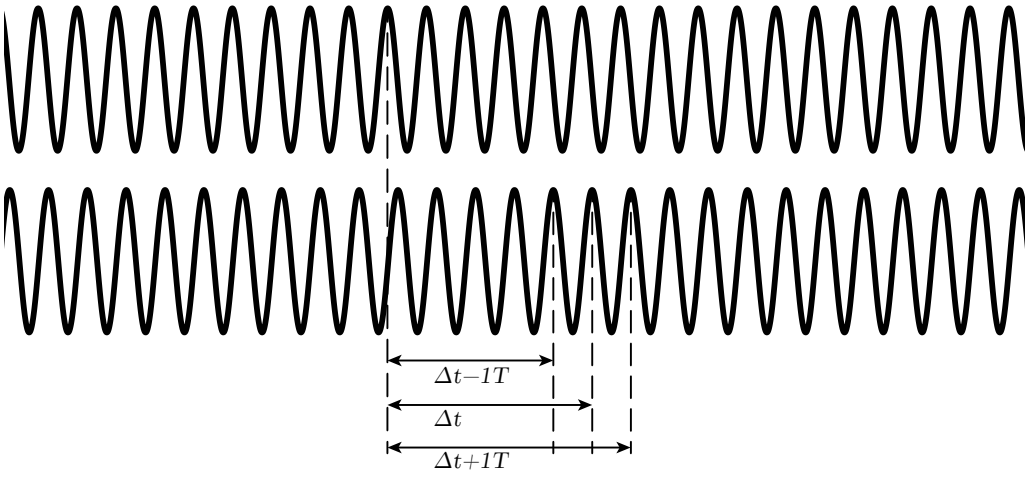

Figure 3.4: The true time offset, *Deltat*, when matching the received carrier signal to the internal oscillator, is not uniquely observable.

If we look a little bit deeper inside the receiver, we find that the phase tracking is not performed on the raw carrier signal, but instead on an intermediate frequency signal created by down mixing the carrier signal. When this intermediate frequency is 0 Hz, i.e., when the mixing signal has a frequency equal to the nominal frequency of the carrier, it is easiest to analyze what is going on. If there was no relative range rate between receiver and satellite, the down mixed signal would be constant. When there is a relative range rate, the frequency of the mixed signal equals the Doppler shift of the carrier signal. The change in phase of this signal, multiplied by the wave length of the nominal carrier signal, equals a change in range to the satellite. Thus, we get a measurement of how much closer, or further away, the satellite is now, as compared to when the signal was first acquired. Instead of wrapping around every whole cycle, the phase should continue counting also the whole number of cycles since it locked on to the signal. This "unwrapped" phase signal, counting number of whole cycles and the fractional part of the cycle, is what is considered the carrier phase measurement from the receiver.

In order to use the carrier phase measurements for localization, we need to solve the problem with the unknown integer of wavelengths between the receiver and satellite at the time phase lock was acquired. There are various methods for resolving this unknown variable when 5 or more satellites are visible and the conditions are otherwise good. The Least-squares Ambiguity Decorrelation Adjustment (LAMBDA) [47], is one popular method to resolve the integer ambiguity.

Error source	Uncorrected	Consumer receiver	High end receiver
Satellite clock	1.1	1.1	0.03
Satellite orbit	0.8	0.8	0.03
Ionosphere	15	7.0	0.1
Troposphere	0.2	0.2	0.2
Receiver noise	N/A	0.1	0.01
Multipath	N/A	0.2	0.01
Total	N/A	7.1	0.2

Table 3.1: Standard deviation of user equivalent range errors (UERE) in meters for a typical consumer receiver using one frequency band [45], and a high end, dual frequency receiver using corrections for satellite errors.

Error sources

In addition to the limited accuracy of the phase lock loops, error in the estimated range to a satellite also comes from other sources. There is one family of errors that relate to the accuracy of the navigational message. Both the atomic clock of the satellite, and the position as given by the orbital parameters encoded in the navigational message, may be wrong by up to a few meters. The clock error, which is measured in seconds, is multiplied by the speed of light, in order to be comparable to other error sources.

On its way from the satellite to the receiver, the signal passes through the atmosphere, and this affects the time of arrival. In the ionosphere, the Sun radiation creates electrically charged particles that delay the signal. This delay varies with the time of day and some other factors. When the signal enters the more dense troposphere, there is yet another delay, which varies less than the ionosphere delay. Thus, it is more predictable when the altitude and weather conditions of the location are known.

Finally, there is multi path error, which is a local error caused by the signal reflecting from surfaces near the receiver. While all the errors above are strongly correlated in space, such that two receivers with a base line of up to a kilometer, experience almost the same satellite errors and atmospheric errors, the multi path error caused by reflections is not correlated when the receivers are more than a few meters apart.

Corrections of errors

All the errors above, with exception for the multi path, are possible to largely compensate for, because they change relatively slowly over time, and are spatially highly correlated. Corrections of the errors come in two

different forms. Either one tries to model all parts and estimate them accurately from a few well known reference stations around the world, or one can bundle up all errors and correct the measurement directly with the help of a nearby reference station. The assumption in the latter case is that, if the base line between the receivers is short enough, then the total error at the two receivers will be almost identical. Thus, the errors will effectively cancel out, when taking the so called double difference [48].

The largest error source, the ionospheric delay, is inversely proportional to the carrier frequency, and thus, receivers that use two or more frequency bands can almost entirely eliminate this error. Single frequency receivers are restricted to rely either on a model of the ionosphere, or on the canceling effect of a nearby base station. Rudimentary state space corrections are part of the base GPS system, in the form of coefficients to Klobuchar's ionospheric model [49], and rough models of satellite clock and orbital parameters. The WAAS/EGNOS/MBAS corrections provide more detailed ionospheric corrections, satellite clock corrections and satellite orbit parameters. There are even more accurate corrections available with a delay of a couple of days from the International GNSS Service (IGS). Because of the delay, these corrections are more accurate than the on-line corrections, and thus, tend to be used in post-processing.

These corrections are used to a varying degree in receivers, depending on different design choices. Precise point positioning (PPP) is a technique that aims at resolving integer ambiguity of the carrier phase measurements by use of these corrections, but without the use of a nearby base state. In contrast, differential GNSS (D-GNSS) works with observation space corrections for the C/A code based pseudo range, and "real time kinematics" (RTK) is when observation space corrections are used to resolve carrier phase measurement ambiguities. The D-GNSS solution usually results in position estimates with error below 0.5 meters, while RTK with properly resolved integer ambiguity results in an error of a few centimeters. Observation space corrections are easy to apply, and converge quickly to an integer solution of the phase ambiguity when compared to state space corrections. However, they require a short base line ($\sim$10km) to the base station to work, and are thus less suitable on the sea, or in other areas where there are no base stations around.

Configuration aspects for GNSS receivers

There are many aspects to consider in the design of a GNSS receiver which all affect the end performance and cost. Number of frequency bands will affect ionosphere error and speed of acquiring integer ambiguity resolution in RTK systems. Use of carrier phase enables high accuracy techniques such as

Frequency bands	Carrier phase	Correction space	Filter design
L1*	no	observation*	loosely coupled
L1+L2	yes*	state	tightly coupled*
L1+L2+L5			ultra-tight / vector

Table 3.2: Configuration aspects for GNSS receivers. The asterisks mark the configuration used in Paper II.

RTK and PPP. Corrections can come either as state space corrections where each error source is estimated, or as observation space corrections where the observations of a nearby base station are subtracted from the mobile receiver observations to form single or double differences. Finally, the design of the localization filter where the GNSS measurements are combined with other position measurements from e.g., an Inertial Measurement Unit (IMU), can be coupled to various degrees. In a loosely coupled filter, the GPS receiver calculates a position/velocity/time (PVT) solution with no feedback from the position filter. The PVT solution provided by the receiver is then at some frequency, sent to the filter, where it is regarded as a measurement. A tightly coupled filter uses the pseudo ranges and phase information to each individual satellite, directly as measurements in the position filter. There they are combined together with measurements from IMU and other sources, as any other landmark measurement. One consequence of this is that, unlike in the loosely coupled filter, pseudo range measurements contribute to the position estimate even when there are only 2 or 3 visible satellites. In even more tightly coupled filters, the solution from the position filter is fed back into the tracking loops of the receiver. Thus, an IMU can help to reacquire the signal after short reception outages, or harden the receiver to spoofing. In Paper II we look at a specific configuration that has not been used much before, but offers a combination of low cost and high relative accuracy; see the combination marked with asterisks in Table 3.2.

2 Camera

Cameras are probably the most popular sensors to use with autonomous vehicles. They combine a low price with information rich measurements. One can also argue for the use of cameras by noting that humans can drive using only the same visual information that cameras capture, and that the road environment with traffic signs, lane markers, etc., is constructed with this in mind.

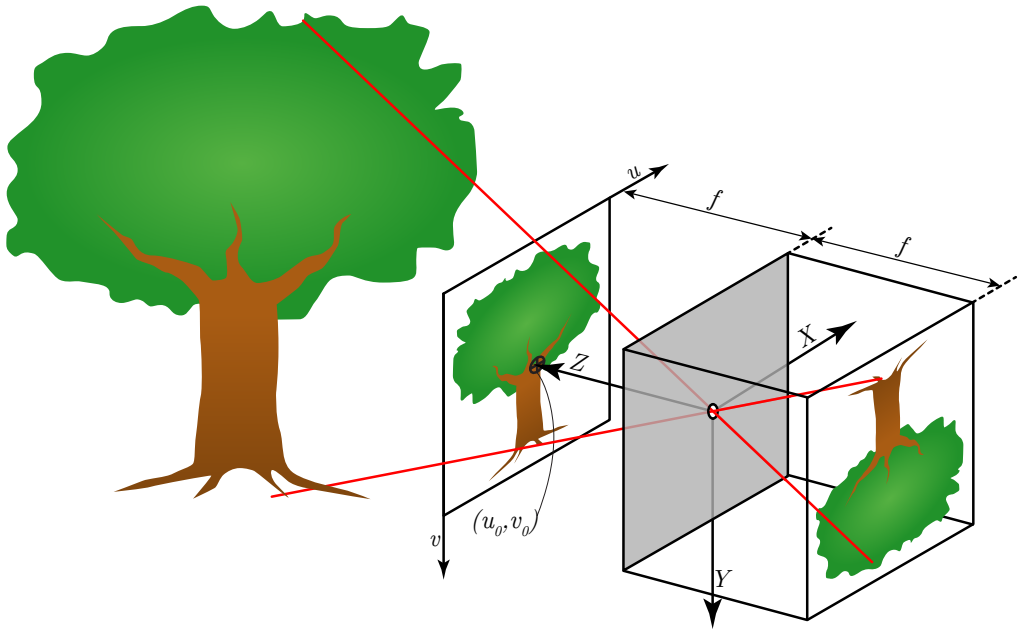

Figure 3.5: Pinhole camera projection of two points from an object, with image plane coordinate system (X, Y, Z), camera coordinate system (u, v), focal length (f), and principal point (u_0, v_0) marked.

Cameras typically do not emit any light, and relies on objects scattering light from various sources in the environment. They measure intensity of light in different angles from the focal point, but usually do not measure the range directly. As such, when using the camera for localization, we make use of several landmarks in the environment with known positions, measure the angle to them, and can then triangulate the ego position.

Pinhole camera model

To be able to use a camera in our localization process, we need a model which describes how the camera measurement, comprising the image pixels, relates to angles to objects. The camera sensor is essentially a grid of photo detectors where each one measures light intensity at a certain angle of incidence to the camera. The geometric relation between a light source in 3-D and the pixel on the flat image sensor which captures the light is modeled by the pinhole camera model in conjunction with a simple non-linear model for distortion [50].

Assuming the camera with focal length f is placed with the focal point (pinhole) in the origin, pointing along the Z-axis, and with the X-axis pointing to the right, a 3-D point, $[X, Y, Z]^\top$, will project to the image plane at

$[-fX/Z, -fY/Z]^\top$ according to the pinhole camera model. This is under the assumption that the image plane lies behind the focal point as in the rightmost projection in 3.5. For convenience, we most often pretend that the image plane lies in front of the focal point, the leftmost projection in 3.5, to get an image that is not upside down, and thus get rid of the negation, $[fX/Z, fY/Z]^\top$. In the case of digital cameras, the image coordinate frame normally has its origin in one of the corners such that the point in the middle of the image is at position $[u_0, v_0]^\top$, leading to an offset in image coordinates as $[fX/Z + u_0, fY/Z + v_0]^\top$.

This equation can be expressed conveniently when using homogeneous coordinates for both 3-D point, $\mathbf{U} = W[X, Y, Z, 1]^\top$, and the 2-D point in the image, $\mathbf{u} = w[u, v, 1]^\top$. We also need to loosen the assumption that the camera is located at the world origin, by introducing a rotation matrix from world coordinates to camera coordinates, $\mathbf{R}$, and a translation vector, $\tilde{\mathbf{C}}$, which gives the camera center in world coordinates. We then get the homogeneous camera coordinates as

$$\mathbf{u} = \mathbf{PU} \tag{3.1}$$

$$\mathbf{P} = \mathbf{KR}[\mathbf{I} \quad | -\tilde{\mathbf{C}}] \tag{3.2}$$

$$\mathbf{K} = \begin{bmatrix} fm_x & 0 & u_0 \\ 0 & fm_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.3}$$

where $\mathbf{P}$ is the complete camera calibration matrix, $\mathbf{K}$ is the intrinsic parameter matrix, m_u and m_v are conversion factors from metric units to pixel units for the focal length, and u_0 and v_0 define the principal point of the camera given in pixel units. Having separate m_u and m_v allows for different size of pixels in u (right) and v (down) directions.

Non linear distortion

For real cameras using optical lenses, the pinhole model is not particularly exact, but the errors can usually be described well with a relatively simple model. By combining the pinhole camera model with a non-linear distortion model, one can achieve sub-pixel accuracy. The distortion model we use is based on [50], and comprises two components: a radial distortion (3.6), and

a decentering distortion (3.7),

$$\hat{\mathbf{u}} = \mathbf{K} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \tag{3.4}$$

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \begin{bmatrix} x_u \\ y_u \end{bmatrix} + \mathcal{R} + \mathcal{D} \tag{3.5}$$

$$\mathcal{R} = \begin{bmatrix} x_u \\ y_u \end{bmatrix} (\kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6 + \dots) \tag{3.6}$$

$$\mathcal{D} = \begin{bmatrix} (2\lambda_1 x_u y_u + \lambda_2 (r^2 + 2x_u^2)) \\ (2\lambda_2 x_u y_u + \lambda_1 (r^2 + 2y_u^2)) \end{bmatrix} (1 + \lambda_3 r^2 + \lambda_4 r^4 + \dots) \tag{3.7}$$

$$r = \sqrt{x_u^2 + y_u^2} \tag{3.8}$$

$$w \begin{bmatrix} x_u \\ y_u \\ 1 \end{bmatrix} = \mathbf{R}[\mathbf{I} \quad | -\tilde{\mathbf{C}}]\mathbf{U}, \tag{3.9}$$

where $\hat{\mathbf{u}}$ is the distorted point in the raw image, x_d and y_d are the normalized distorted image coordinates, x_u and y_u are the normalized undistorted image coordinates, $\mathcal{R}$ is the radial distortion, $\mathcal{D}$ is the decentering distortion, κ_i are the parameters for radial distortion, λ_i are the parameters for decentering distortion, and w is the homogeneous scaling factor. Usually the distortion model is good enough using only κ_1 and κ_2, but often λ_1, λ_2, and κ_3 are also considered. These non-linear distortion parameters, together with the parameters in $\mathbf{K}$ (3.3), are the intrinsic parameters of the camera. They are usually determined in a calibration procedure by maximum likelihood estimation [51, 52]. With calibrated cameras, and assuming that the calibration stays constant over time, one only needs to determine the 6 free parameters of the camera pose for each image frame used in the localization. This is the procedure used in Paper III.

Image features

Now that we have a good model for how points in the world are projected to the camera image, we look into what we use the model for. The landmarks that we save in the map, must be possible to detect and localize in the images, and we do this using various feature detectors.

Typical examples of general image features are corner points, local minima or maxima of intensity, lines, or smooth curves. These features are often detected by looking at the gradient and finding where it is large (or 0 in the case of local extremal points), see [53] for an overview of general image feature detectors. These points or lines have no semantic meaning beyond

being points that are easy to detect and identify, and are quite general in the sense that they tend to work decently on most images, as long as the images are not extremely blurry or "smooth".

Often feature descriptors are used together with the detected features, in order to aid the data association (correspondence) between features at different time instances, or to landmarks in the map. There are many general purpose descriptors, e.g., SIFT [54], SURF [55], and BRIEF [56], which condense the local neighbourhood of a point in an image into a short vector which should be distinctive for this particular point. Again, these descriptors are general in the sense that they tend to work decently on most images.

Another approach to image features, is to detect features which have a special meaning to humans. In a road environment that could mean e.g., traffic signs, pedestrians, or lane markers. These detectors are harder to construct, but it is more obvious how the detections are useful in an autonomous car context, when compared to general feature points. Recent object detection algorithms are often based on supervised learning algorithms, specifically deep neural networks. These algorithms rely on having many manually labeled training examples to learn from. When that is available they have shown great performance, compared to hand engineered algorithms. These deep neural networks were first used to classify whole images, but also to detect interesting objects with bounding boxes within images. Recently there has been a surge in pixel wise classification, also called semantic segmentation. The task for a semantic classifier is to assign a semantically meaningful class, such as "human", "car", "building", etc., to each pixel in the image. With the introduction of fully convolutional networks [57], it is now possible to get quite good performance in this type of tasks, and one can see how it takes the bounding box approach one step further.

Automotive cameras detect and classify objects that are meaningful for the driving task, such as other vehicles, pedestrians, traffic signs, lane markers, etc. Usually these features are detected and located in the image, but sometimes the detected positions are transformed into a vehicle coordinate frame. This transformation into 3-D can be achieved after capturing two or more frames and using visual odometry to triangulate the position of the object, or by using a flat world assumption, and knowledge about the mounting position of the camera on the vehicle. In Paper I we use only the lane marker descriptions from a camera of this type.

3 Radar

Radars work by illuminating objects with their own "light" source, and measure the time it takes for the signal to return, and can hence deduce the distance to the object. They emit radio waves in a beam which is swept over the scene of interest. Some older models used a physically moving antenna, but modern radars use electronically steered beams. The antenna elements are arranged such that by shifting the phase of the transmitted signal slightly, a beam can be directed in a selected angle. By steering this beam, the angle to objects can be measured. Besides angle, range and reflectivity of the objects are measured, and most often also the Doppler shift in the returned signal is measured, which means that the relative speed between the radar and the object can be determined. As with automotive cameras, automotive radars do further processing to detect peaks in the raw data which should correspond to the objects of interest. The radar usually also tracks and filters these detections over multiple frames before publishing the information on the data bus. Since the speed of the vehicle relative to the ground is estimated relatively well by the wheel speed sensors, and the relative speed to a target is measured by the radar, it is possible to extract the stationary objects for use in the localization process, while disregarding the moving objects, which are not of interest for localization purposes. Traffic signs and other metallic poles, such as the ones holding up side barriers, are typical examples of objects that are both stationary and good radar reflectors, and thus, of interest as landmarks in a radar map.

In contrast to camera images, the detection from a radar contain relatively little information. There is, to our best knowledge, no way to produce an effective descriptor for the radar detections. Thus, it will be much harder to match radar detections over time, or to a radar map. One way of handling the measurement to map association is presented in Paper I, but there are many other possible solutions to the data association problem, see e.g., [58–61].

Chapter 4

Filtering and smoothing

Bayesian filtering and smoothing are tools which are very useful for the task of localization. They can be used to answer questions such as, "What is the most likely location of the car, when taking into account all measurements that we have done up until now?" or "How certain is this estimated location?".

The word "Bayesian" in "Bayesian filtering" means that we make use of Bayes' rule to update our belief of some state as we collect more measurements. This belief is expressed as a probability density (or probability mass function in the case of discrete variables), and is exactly what we use to answer questions about most likely value or uncertainty about some variable.

"Filtering", in this context, means that we have a sequence of noisy measurements from our sensors, and are interested in recursively estimating some state (e.g. the current location and orientation of a car), making use of all past measurements to infer this state. This is in contrast to "smoothing" where we are looking at the problem in an off-line setting, and thus also future measurements are available for all but the very last time instance.

An example of the on-line filtering setting, is the localization necessary for autonomous driving. The current location of the vehicle is most relevant for planning and control but the location one hour ago is not that useful. On the other hand, creating a map of the static environment around the road, is an example where smoothing makes more sense. Then, we are interested in creating the best possible estimate of the positions of landmarks, and we would like to use all available data. However, it is not critical to make the estimation on-line while recording the measurements. We may just as well save all the data and process it afterwards, when we return to the office.

1 Problem formulation and general solution

Let us define the state that we are interested in estimating as a vector of real numbers, $\mathbf{x}_k \in \mathbb{R}^n$, and the measurement we receive at time t_k as another vector, $\mathbf{y}_k \in \mathbb{R}^m$. The subscript k denotes the k:th measurement which is taken at time t_k. One measurement sequence consists of T discrete measurements, which means that $k \in [1, \ldots, T]$ and that $j < k \implies t_j \leq t_k$. We can now define filtering as finding the posterior density, i.e., the density over possible states after all available measurements have been accounted for, $p(\mathbf{x}_k|\mathbf{y}_{1:k})$, and smoothing as $p(\mathbf{x}_{1:T}|\mathbf{y}_{1:T})$. Here, $\mathbf{y}_{1:k}$ is short notation for $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k$, and T is the last time instance in the sequence.

To find the posterior density, we make a few simplifying assumptions. One assumption is that the current measurement $\mathbf{y}_k$ when the corresponding state $\mathbf{x}_k$ is given, is conditionally independent of all other states and measurements. Another assumption is that the state sequence is Markovian, which means that state $\mathbf{x}_k$ when given state $\mathbf{x}_{k-1}$ is conditionally independent of all previous states,

$$p(\mathbf{x}_k|\mathbf{x}_{1:k-1}, \mathbf{y}_{1:k-1}) = p(\mathbf{x}_k|\mathbf{x}_{k-1}) \tag{4.1}$$

$$p(\mathbf{y}_k|\mathbf{x}_{1:k}, \mathbf{y}_{1:k-1}) = p(\mathbf{y}_k|\mathbf{x}_k). \tag{4.2}$$

Density (4.1) captures the uncertainties in the state transition, and (4.2) captures the relation between the measurement and the state. The same relations can also be expressed as

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}) \tag{4.3}$$

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{r}_k), \tag{4.4}$$

where $\mathbf{q}_k$ and $\mathbf{r}_k$ are random processes, describing the error or uncertainty in the models. As (4.3) models the procession of states over time, it is called a process model, or motion model when the state involves position. Similarly, (4.4) models the measurements, and is called a measurement model.

When we have restricted the class of problems to Markovian processes with conditionally independent measurements, the filtering and smoothing problems can be solved with recursive algorithms. In the filtering case, there is a forward recursion, whereas in the smoothing case, there is also a backward recursion. The base case in the filtering solution is for $k = 0$, when we have no measurement, and we base our solution solely on any prior knowledge we may have, summarized in $p(\mathbf{x}_0)$.

The forward recursion step is then split in two parts: a prediction using the process model, and then an update using the measurement model. Assuming we have a solution for $p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})$ from the previous time instance, we can now express $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ in terms of already known densities

with the help of Dr. Kolmogorov and Reverend Bayes as

$$p(\mathbf{x}_k|\mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_{k-1}, \mathbf{x}_k|\mathbf{y}_{1:k-1})d\mathbf{x}_{k-1} \tag{4.5}$$

$$= \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{y}_{1:k-1})d\mathbf{x}_{k-1} \tag{4.6}$$

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k-1})}{p(\mathbf{y}_k|\mathbf{y}_{1:k-1})}, \tag{4.7}$$

where the denominator in (4.7) is constant with respect to the state variable $\mathbf{x}_k$.

As mentioned, smoothing can also be done recursively in two passes, where the first forward pass is identical to the filter recursion, and then a backward pass. The base case for the backward recursion is the final state at $k = T$, where the smoothing solution is identical to the filtering solution, $p(\mathbf{x}_T|\mathbf{y}_{1:T})$. Then assume that we have the smoothed solution at time $k+1$, $p(\mathbf{x}_{k+1}|\mathbf{y}_{1:T})$ and now the task is to express $p(\mathbf{x}_k|\mathbf{y}_{1:T})$ in already known terms,

$$p(\mathbf{x}_k|\mathbf{y}_{1:T}) = \int p(\mathbf{x}_k, \mathbf{x}_{k+1}|\mathbf{y}_{1:T})d\mathbf{x}_{k+1} \tag{4.8}$$

$$= \int p(\mathbf{x}_k|\mathbf{x}_{k+1}, \mathbf{y}_{1:T})p(\mathbf{x}_{k+1}|\mathbf{y}_{1:T})d\mathbf{x}_{k+1} \tag{4.9}$$

$$= \int p(\mathbf{x}_k|\mathbf{x}_{k+1}, \mathbf{y}_{1:k})p(\mathbf{x}_{k+1}|\mathbf{y}_{1:T})d\mathbf{x}_{k+1} \tag{4.10}$$

$$= \int \frac{p(\mathbf{x}_{k+1}|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{y}_{1:k})p(\mathbf{x}_{k+1}|\mathbf{y}_{1:T})}{p(\mathbf{x}_{k+1}|\mathbf{y}_{1:k})}d\mathbf{x}_{k+1}. \tag{4.11}$$

In (4.11) we see the motion model $p(\mathbf{x}_{k+1}|\mathbf{x}_k)$, the filtering density $p(\mathbf{x}_k|\mathbf{y}_{1:k})$, the smoothing density from the previous step $p(\mathbf{x}_{k+1}|\mathbf{y}_{1:T})$, and the filtering prediction density $p(\mathbf{x}_{k+1}|\mathbf{y}_{1:k})$, all of which are available from before.

2 Kalman filter

If the errors $\mathbf{q}_k$ and $\mathbf{r}_k$ from (4.3) and (4.4) are additive, normally distributed, and independent over time, and the models in themselves are linear, then we can solve the filtering and smoothing problems optimally, in a mean square error sense, using a Kalman filter [62]. All the assumptions regarding the process noise are seldom correct, but it is still a useful simplification which works often enough. Also, there is seldom a need for having different models at each time instance. Then the simplified models are

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \qquad \mathbf{q}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{4.12}$$

$$\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{r}_k, \qquad \mathbf{r}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \tag{4.13}$$

where $\mathbf{F}$ is the linear process model in matrix form, $\mathbf{H}$ is the linear measurement model in matrix form, and $\mathbf{Q}$ and $\mathbf{R}$ are the covariance matrices for the process noise and the measurement noise, respectively. Under these assumptions, the posterior distributions for both the filtering problem and the smoothing problem are also Gaussian, and can be exactly represented by their mean and covariance. The Kalman filter calculates mean, $\hat{\boldsymbol{\mu}}_{k|k}$, and covariance $\mathbf{P}_{k|k}$, of the posterior density for each time step, k, with the following recursion

$$\hat{\boldsymbol{\mu}}_{k|k-1} = \mathbf{F}\hat{\boldsymbol{\mu}}_{k-1} \tag{4.14}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^\top + \mathbf{Q} \tag{4.15}$$

$$\hat{\mathbf{y}}_k = \mathbf{H}\hat{\boldsymbol{\mu}}_{k|k-1} \tag{4.16}$$

$$\mathbf{S}_k = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^\top + \mathbf{R} \tag{4.17}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^\top\mathbf{S}_k^{-1} \tag{4.18}$$

$$\hat{\boldsymbol{\mu}}_{k|k} = \hat{\boldsymbol{\mu}}_{k|k-1} + \mathbf{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k) \tag{4.19}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^\top. \tag{4.20}$$

Here $\hat{\boldsymbol{\mu}}_{k|k}$ is the estimated mean at time instance k using measurements up to, and including time k, while $\hat{\boldsymbol{\mu}}_{k|k-1}$ is the predicted mean at time instance k using measurements up to, and including time $k-1$, $\hat{\mathbf{y}}_k$ is the predicted measurement, $\mathbf{S}_k$ is the predicted measurement covariance, and $\mathbf{K}_k$ is the Kalman gain, all at time k.

The optimal smoothed solution can be achieved by first applying the Kalman filter, and then applying the Rauch-Tung-Striebel smoothing recursion [63], starting from $k = T - 1$ as

$$\mathbf{G}_k = \mathbf{P}_{k|k}\mathbf{F}^\top\mathbf{P}_{k+1|k}^{-1} \tag{4.21}$$

$$\hat{\boldsymbol{\mu}}_{k|T} = \hat{\boldsymbol{\mu}}_{k|k} + \mathbf{G}_k(\hat{\boldsymbol{\mu}}_{k+1|T} - \hat{\boldsymbol{\mu}}_{k+1|k}) \tag{4.22}$$

$$\mathbf{P}_{k|T} = \mathbf{P}_{k|k} + \mathbf{G}_k(\mathbf{P}_{k+1|T} - \mathbf{P}_{k+1|k})\mathbf{G}_k^\top. \tag{4.23}$$

3 Unscented Kalman Filter

In many problems either, or both, of the process and measurement models are non-linear, in which case there is usually no exact solution. One common solution, when the non-linearities are relatively mild, is to linearize

the functions using Taylor series expansion at the estimated mean. This results in the posterior being approximated as a normal distribution, and the algorithms to calculate it, are the Extended Kalman filter (EKF) [64, 65] and Extended Rauch-Tung-Striebel smoother (ERTS), respectively.

Another option, which is used in the appended papers, is the sigma point methods, such as the Unscented Kalman Filter (UKF) [66] and the Cubature Kalman Filter (CKF) [67]. These methods also approximate the posterior as a normal distribution, but in a slightly different way than through Taylor series expansion. One can think of the UKF and CKF in terms of numerical differences and view them as approximations to the EKF, but that is somewhat unfair, since both UKF and CKF generally approximate the densities involved better than what the EKF does [68]. The Unscented transform, which is the basis of the UKF, estimates the mean and covariance of $\mathbf{y} = \mathbf{h}(\mathbf{x})$ where $\mathbf{x}$ is a normally distributed random variable, by first selecting a set of sigma points, $\mathcal{X}^{(i)}$, around the mean of $\mathbf{x}$. These sigma points are then propagated through the non-linear function, resulting in another set of sigma points, $\mathcal{Y}^{(i)}$. Then, the mean and covariance of $\mathbf{y}$ can be approximated from $\mathcal{Y}^{(i)}$ as a weighted sum.

The UKF recursion which approximates the posterior density as a normal distribution, is then given in terms of its approximated mean, $\hat{\boldsymbol{\mu}}_k$, and covariance $\mathbf{P}_k$ for each time step k. First determine the sigma points,

$$\mathcal{X}_{k-1}^{(0)} = \hat{\boldsymbol{\mu}}_{k-1} \tag{4.24}$$

$$\mathcal{X}_{k-1}^{(i)} = \hat{\boldsymbol{\mu}}_{k-1} + \sqrt{n+\kappa}\left[\sqrt{\mathbf{P}_{k-1}}\right]_i \tag{4.25}$$

$$\mathcal{X}_{k-1}^{(i+n)} = \hat{\boldsymbol{\mu}}_{k-1} - \sqrt{n+\kappa}\left[\sqrt{\mathbf{P}_{k-1}}\right]_i, \tag{4.26}$$

then the prediction density,

$$\mathcal{X}_{k|k-1}^{(i)} = \mathbf{f}(\mathcal{X}_{k-1}^{(i)}) \tag{4.27}$$

$$\hat{\boldsymbol{\mu}}_{k|k-1} = \sum_{i=0}^{2n} W^{(i)} \mathcal{X}_{k|k-1}^{(i)} \tag{4.28}$$

$$\mathbf{P}_{k|k-1} = \sum_{i=0}^{2n} W^{(i)} (\mathcal{X}_{k|k-1}^{(i)} - \hat{\boldsymbol{\mu}}_{k|k-1})(\mathcal{X}_{k|k-1}^{(i)} - \hat{\boldsymbol{\mu}}_{k|k-1})^\top \tag{4.29}$$

then the posterior density,

$$\mathcal{Y}_k^{(i)} = \mathbf{h}(\mathcal{X}_{k|k-1}^{(i)}) \tag{4.30}$$

$$\hat{\mathbf{y}}_k = \sum_{i=0}^{2n} W^{(i)} \mathcal{Y}_k^{(i)} \tag{4.31}$$

$$\mathbf{S}_k = \sum_{i=0}^{2n} W^{(i)} (\mathcal{Y}_k^{(i)} - \hat{\mathbf{y}}_k)(\mathcal{Y}_k^{(i)} - \hat{\mathbf{y}}_k)^\top \tag{4.32}$$

$$\mathbf{C}_k = \sum_{i=0}^{2n} W^{(i)} (\mathcal{X}_{k|k-1}^{(i)} - \hat{\boldsymbol{\mu}}_{k|k-1})(\mathcal{Y}_k^{(i)} - \hat{\mathbf{y}}_k)^\top \tag{4.33}$$

$$\mathbf{K}_k = \mathbf{C}_k \mathbf{S}_k^{-1} \tag{4.34}$$

$$\hat{\boldsymbol{\mu}}_{k|k} = \hat{\boldsymbol{\mu}}_{k|k-1} + \mathbf{K}_k(\mathbf{y}_k - \hat{\mathbf{y}}_k) \tag{4.35}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^\top \tag{4.36}$$

all with weights as

$$W^{(0)} = \frac{\kappa}{n+\kappa} \tag{4.37}$$

$$W^{(i)} = \frac{1}{2(n+\kappa)}, \quad i = 1, \ldots, 2n. \tag{4.38}$$

Here n is the dimensionality of the state, κ is a tuning parameter, $\sqrt{\mathbf{P}}$ denotes any matrix square root such that $\sqrt{\mathbf{P}}\sqrt{\mathbf{P}}^\top = \mathbf{P}$, and $\left[\,\cdot\,\right]_i$ selects the i:th column of its argument. The special case when $\kappa = 0$ turns the UKF into CKF, which is used in Paper II.

In cases where the noise is not additive, it is possible to augment the state vector with noise states and use the same updates of the UKF or CKF filters [69].

4 Particle filters

Both the EKF and UKF approximates the posterior density as a normal density. Sometimes this approximation can be too crude. This may happen e.g., when the process model or the measurement model is highly non-linear, or it is known that the posterior density is multi-modal, and this needs to be reflected in the filtering solution. One possible solution when this is the case, is through multiple hypotheses filters [70–72], which describe the posterior as a sum of several normal distributions. Another popular solution is the particle filter, which makes no parametric approximation of the posterior density, but instead describes the posterior distribution as a weighted sum of many Dirac delta functions.

When using a particle filter, we draw N samples, $\mathbf{x}^{(i)}$, from a proposal distribution $q(\mathbf{x}_{0:k}|y_{1:k})$ and assign a weight $w^{(i)}$ to each sample, where

$$w_k^{(i)} = \frac{p(\mathbf{x}_{0:k}|y_{1:k})}{q(\mathbf{x}_{0:k}|y_{1:k})}. \tag{4.39}$$

Now the posterior distribution can be approximated as

$$p(\mathbf{x}_{0:k}|\mathbf{y}_{1:k}) \approx \frac{1}{N}\sum_{i=1}^{N} w_k^{(i)}\delta(\mathbf{x}_{0:k} - \mathbf{x}_{0:k}^{(i)}). \tag{4.40}$$

For the recursive step, assume that we have $\mathbf{x}_{0:k-1}^{(i)}$ and $w_{k-1}^{(i)}$ from the previous step. The new set of samples at step k is drawn from the proposal distribution and gets weights as

$$\mathbf{x}_k^{(i)} \sim q(\mathbf{x}_k|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k}) \tag{4.41}$$

$$w_k^{(i)} \propto \frac{p(\mathbf{y}_k|\mathbf{x}_k^{(i)})p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)}|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k})} w_{k-1}^{(i)}. \tag{4.42}$$

The version of particle filters that is used in Papers I and III, called the bootstrap particle filter [73], uses the motion model as proposal distribution, $q(\mathbf{x}_k^{(i)}|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{y}_{1:k}) = p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})$, which makes for a particularly easy recursion,

$$\mathbf{x}_k^{(i)} \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)}) \tag{4.43}$$

$$w_k^{(i)} \propto p(\mathbf{y}_k|\mathbf{x}_k^{(i)})w_{k-1}^{(i)}. \tag{4.44}$$

After the new samples are drawn and the weights are updated, the weights should be normalized such that $\sum_i w_i = 1$ over time. Due to the possibility of particle depletion, i.e. most of w_i are almost zero, the samples need to be resampled periodically. The resampling creates multiple copies of samples with large weights, and removes samples with near zero weight, and then resets the weights of the resampled particles.

5 Smoothing and mapping

Bayesian smoothing of a Markov process can be done optimally in a forward pass, followed by a backwards pass as seen in (4.11). However, for a typical mapping problem the Markov property does not hold, because when one returns to a previously visited place and recognizes this (loop closure), a direct dependence from that earlier time to now is created.

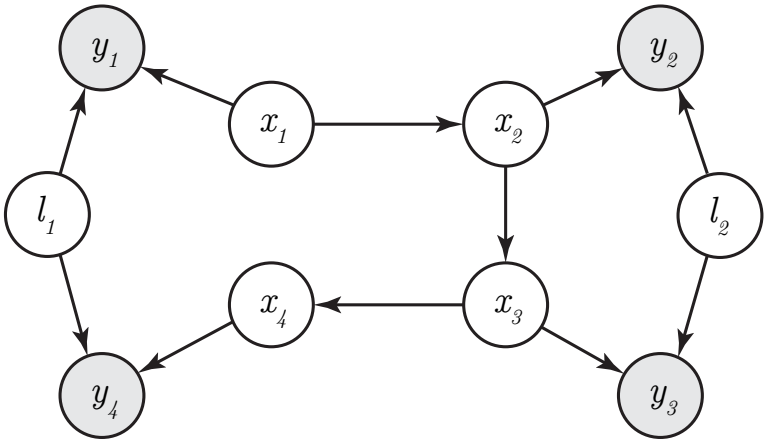

Figure 4.1: Bayes net representation of a small SLAM problem with 4 robot poses, 2 landmarks, and 4 observations. Unknown variables are white, while given measurements are marked gray.

Let us create a small example that shows the problem. Assume that we have two landmarks, l_1 and l_2, that we are interested in mapping. We drive near them with a robot, and make noisy observations with a sensor on the robot. Our problem now, is that we do not know exactly the positions and orientations of the robot during the measurement campaign. Let us assume that the robot has made four measurements, $y_1, \ldots, y_4$, of the landmarks from four different positions, $x_1, \ldots, x_4$. Let l_1 be seen from x_1 and x_4, and l_2 from x_2 and x_3. The corresponding Bayes network is shown in Figure 4.1. The relation between the posterior density of interest $p(l_1, l_2|y_{1:4})$ and the measurement models $p(y_1|x_1, l_1)$ and $p(y_2|x_2, l_2)$ can be seen by introducing the x:s in the posterior density,

$$p(l_1, l_2|y_{1:4}) = \int p(l_{1:2}, x_{1:4}|y_{1:4}) dx_{1:4}. \tag{4.45}$$

Now, the joint density inside the integral can be factorized in terms of motion model and measurement model.

$$\begin{aligned} p(l_{1:2}, x_{1:4}|y_{1:4}) \propto\ & p(y_1|x_1, l_1)p(x_2|x_1) \\ & p(y_2|x_2, l_2)p(x_3|x_2) \\ & p(y_3|x_3, l_2)p(x_4|x_3) \\ & p(y_4|x_4, l_1) \end{aligned} \tag{4.46}$$

The landmarks, especially l_1 which appears together with x_1 to x_4, prevent us from solving this with forward-backward smoothing. Therefore, we have to fall back to iterative methods. The iterative method could be for

example, loopy belief propagation [74], but often it is simply reformulated as a maximum á-posteriori (MAP) problem. When the noise in the models is additive and Gaussian, which often is the assumption, the MAP problem is equivalent to a weighted least squares problem,

$$\begin{aligned}\arg\max_{l_{1:2},x_{1:4}} p(l_{1:2},x_{1:4}|y_{1:4}) &= \arg\min_{l_{1:2},x_{1:4}} -\log p(l_{1:2},x_{1:4}|y_{1:4}) \\ &= \arg\min_{l_{1:2},x_{1:4}} ||y_1 - h(x_1,l_1)||_R^2 + \\ &\qquad ||x_2 - f(x_1)||_Q^2 + \ldots + \\ &\qquad ||y_4 - h(x_4,l_1)||_R^2, \end{aligned} \tag{4.47}$$

where $||x||_\Sigma^2 = x^\top \Sigma^{-1} x$. This least-squares problem can be solved using a general purpose optimizer, such as Levenberg-Marquardt. This formulation of the mapping problem is usually called Graph-SLAM, where SLAM stands for "simultaneous localization and mapping".

By solving the optimization problem in (4.47), we get the maximum á-posteriori solution of both poses and landmarks. As always with local optimization algorithms, there is a risk of getting stuck in local minimas, but if the initialization is good, this is less of a problem.

Chapter 5

Contributions and future work

In Paper I [75], we investigate localization performance when using a particular setup of low cost, automotive grade sensors found on a production vehicle from 2014. Performance when both camera and radar are functional and seeing landmarks, is found to be satisfactory, but when one or both is absent, performance quickly degrades past acceptable limits. To reach the desired performance, a few items are identified as candidates for further improvement. Information from the sensors is not used to its fullest extent, due to preprocessing in the sensors; e.g., the position estimates from the GPS system are not useful for anything more than a first rough initialization. The examined solution is sensitive to patches where landmarks are absent, so if error growth was much slower while landmarks are unavailable, the patches without observable landmarks could be handled better. The lane marker features provided by the vision system, are not sufficient for stand alone localization. Data association of measurements to the map is not trivial, and give rise to large errors if assumed to be correct when they are not. Modeling and implementation was a shared work between the first author and me, under the supervision of the two last authors.

In Paper II [76], we propose an alternative configuration of GNSS receiver. It uses only the basic state space corrections provided in the navigational message, and the carrier phase observations, to obtain a highly accurate relative positioning without the need for communication with base stations, or complex and time consuming ambiguity resolution. Together with a standard automotive grade IMU, and wheel speed sensors, this type of receiver would enable accurate positioning in open environments where landmarks may be scarce, but GNSS availability is high. Models and implementation for this alternative configuration was done by me under supervision of the second author.

In Paper III [77], we aim to increase use of the camera beyond the simplest features, such as lane markings, while still being robust towards

the large changes in visual appearance that occur naturally due to different seasons and lighting conditions. Aiming for lower dependence towards the visual appearance of landmarks in the map representation, a model for localization in semantically labeled point clouds is proposed, implemented, and evaluated against a reference method based on traditional image features. Although the intended resilience towards changing appearance was not achieved, it was shown that visual localization using point clouds can be done with far less informative descriptors where matching of feature points from image to map is infeasible, and yet achieve comparable performance. Idea, localization model, and implementation was done primarily by me under supervision of the last author. Reference localization was done primarily by the second author. Mapping of environment and obtaining ground truth poses was shared work between last author and me.

Future work

Below are some thoughts about future work that may follow this thesis.

- **Derive accuracy and reliability requirements**
  All the papers in this thesis are missing a good definition of what the required localization accuracy and reliability requirements are. This ought to be properly defined and motivated, preferably through a derivation as hinted at in Chapter 2.

- **Increase accuracy and scope of semantic localization**
  The accuracy of the solution in Paper III is not satisfactory. Perhaps using more advanced map primitives may help in increasing accuracy. One could imagine that line or surface descriptors would provide richer information, and thus also better performance for localization. Also, visual changes from day to night are in some sense bigger than changes over the seasons, but they were not in the scope in Paper III, since it was neither included in the data set used for training the semantic segmenter, nor in the dataset used for testing localization performance.

- **Train semantically aware image feature detector/descriptor**
  Even though data size of the maps produced in Paper III is considerably smaller when comparing to a point cloud map using SIFT descriptors, it is still rather large, and may need further compression. One possibility could be to train a better feature detector, possibly also together with a feature descriptor. This detector and descriptor should be semantically away such that only the most relevant points that are known to be constant over time and easy to recognize in various conditions, are detected and included in the map.

Bibliography

[1] Trafikanalys, "Vägtrafikskador 2011", Tech. Rep.

[2] National Highway Traffic Safety Administration, "Traffic safety facts 2015", Tech. Rep.

[3] National Highway Traffic Safety Association, "Motor vehicle traffic crashes as a leading cause of death in the united states, 2012-2014", Tech. Rep., 2016.

[4] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey", Tech. Rep., 2015.

[5] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network", in *Advances in neural information processing systems*, 1989, pp. 305–313.

[6] U. Muller *et al.*, "Off-road obstacle avoidance through end-to-end learning", in *Advances in neural information processing systems*, 2006, pp. 739–746.

[7] M. Bojarski *et al.*, "End to end learning for self-driving cars", *ArXiv preprint arXiv:1604.07316*, 2016.

[8] I. Ulrich and I. Nourbakhsh, "Appearance-based place recognition for topological localization", in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, Ieee, vol. 2, 2000, pp. 1023–1029.

[9] A. Torralba *et al.*, "Context-based vision system for place and object recognition", in *Proceedings of the Ninth IEEE International Conference on Computer Vision-Volume 2*, IEEE Computer Society, 2003, p. 273.

[10] E. Olson, "Recognizing places using spectrally clustered local matches", *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1157–1172, 2009.

[11] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance", *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[12] M. J. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights", in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 1643–1649.

[13] T. Naseer *et al.*, "Robust visual robot localization across seasons using network flows", in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[14] H. Badino, D. Huber, and T. Kanade, "Visual topometric localization", in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, IEEE, 2011, pp. 794–799.

[15] L. Svarm *et al.*, "Accurate localization and pose estimation for large 3d models", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 532–539.

[16] T. Sattler, B. Leibe, and L. Kobbelt, "Efficient & effective prioritized matching for large-scale image-based localization", *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 9, pp. 1744–1756, 2017.

[17] Y. Li, N. Snavely, and D. P. Huttenlocher, "Location recognition using prioritized feature matching", in *European conference on computer vision*, Springer, 2010, pp. 791–804.

[18] Y. Li *et al.*, "Worldwide pose estimation using 3d point clouds", in *Large-Scale Visual Geo-Localization*, Springer, 2016, pp. 147–163.

[19] G. Schindler, M. Brown, and R. Szeliski, "City-scale location recognition", in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, IEEE, 2007, pp. 1–7.

[20] A. Irschara *et al.*, "From structure-from-motion point clouds to fast location recognition", in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, IEEE, 2009, pp. 2599–2606.

[21] H. Durrant-Whyte, D. Rye, and E. Nebot, "Localization of autonomous guided vehicles", in *Robotics Research*, Springer, 1996, pp. 613–625.

[22] S. Thrun *et al.*, "Robust monte carlo localization for mobile robots", *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[23] A. C. Murillo, J. J. Guerrero, and C. Sagues, "Surf features for efficient robot localization with omnidirectional images", in *Robotics and Automation, 2007 IEEE International Conference on*, IEEE, 2007, pp. 3901–3907.

[24] S. Corbett and P. Cross, "Gps single epoch ambiguity resolution", *Survey Review*, vol. 33, no. 257, pp. 149–160, 1995.

[25] S. Thrun *et al.*, "Stanley: The robot that won the darpa grand challenge", *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[26] C. Urmson *et al.*, "Autonomous driving in traffic: Boss and the urban challenge", *AI magazine*, vol. 30, no. 2, p. 17, 2009.

[27] A. Vu, J. A. Farrell, and M. Barth, "Centimeter-accuracy smoothed vehicle trajectory estimation", *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 4, pp. 121–135, 2013.

[28] M. Schreiber, C. Knöppel, and U. Franke, "Laneloc: Lane marking based localization using highly accurate maps", in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, IEEE, 2013, pp. 449–454.

[29] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i", *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[30] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): Part ii", *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[31] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics", in *Autonomous robot vehicles*, Springer, 1990, pp. 167–193.

[32] J. J. Leonard and H. F. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot", in *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*, Ieee, 1991, pp. 1442–1447.

[33] M. Montemerlo *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem", in *Aaai/iaai*, 2002, pp. 593–598.

[34] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures", *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[35] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing", *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[36] M. Kaess, A. Ranganathan, and F. Dellaert, "Isam: Incremental smoothing and mapping", *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[37] D. Cremers, "Direct methods for 3d reconstruction and visual slam", in *Machine Vision Applications (MVA), 2017 Fifteenth IAPR International Conference on*, IEEE, 2017, pp. 34–38.

[38] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras", *IEEE Transactions on Robotics*, 2017.

[39] B. Triggs *et al.*, "Bundle adjustment-a modern synthesis", in *International workshop on vision algorithms*, Springer, 1999, pp. 298–372.

[40] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Second Edition. Cambridge University Press, ISBN: 0521540518, 2004.

[41] A. Elfes, "Sonar-based real-world mapping and navigation", *IEEE Journal on Robotics and Automation*, vol. 3, no. 3, pp. 249–265, 1987.

[42] D. Pagac, E. M. Nebot, and H. Durrant-Whyte, "An evidential approach to map-building for autonomous vehicles", *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 623–629, 1998.

[43] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps", in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, 2010, pp. 4372–4378.

[44] A. Hornung *et al.*, "Octomap: An efficient probabilistic 3d mapping framework based on octrees", *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[45] E. Kaplan and C. Hegarty, *Understanding GPS: Principles and Applications, Second Edition*, ser. Artech House mobile communications series. Artech House, 2005.

[46] R. Hatch, "The synergism of gps code and carrier measurements", in *International geodetic symposium on satellite doppler positioning*, vol. 1, 1983, pp. 1213–1231.

[47] P. J. Teunissen, "The least-squares ambiguity decorrelation adjustment: A method for fast gps integer ambiguity estimation", *Journal of geodesy*, vol. 70, no. 1, pp. 65–82, 1995.

[48] M. Petovello, "The differences in differencing", *Inside GNSS*, pp. 28–32, Sep. 2011.

[49] J. Klobuchar *et al.*, "Ionospheric time-delay algorithm for single-frequency gps users", *Aerospace and Electronic Systems, IEEE Transactions on*, no. 3, pp. 325–331, 1987.

[50] D. C. Brown, "Close-range camera calibration", *Photogramm. Eng*, vol. 37, no. 8, pp. 855–866, 1971.

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

[52] Z. Zhang, "A flexible new technique for camera calibration", *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[53] T. Tuytelaars, K. Mikolajczyk, *et al.*, "Local invariant feature detectors: A survey", *Foundations and trends® in computer graphics and vision*, vol. 3, no. 3, pp. 177–280, 2008.

[54] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[55] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features", *Computer vision–ECCV 2006*, pp. 404–417, 2006.

[56] M. Calonder *et al.*, "Brief: Binary robust independent elementary features", *Computer Vision–ECCV 2010*, pp. 778–792, 2010.

[57] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[58] C. Lundquist, L. Hammarstrand, and F. Gustafsson, "Road intensity based mapping using radar measurements with a probability hypothesis density filter", *IEEE Transactions on Signal Processing*, vol. 59, no. 4, pp. 1397–1408, 2011.

[59] L. Hammarstrand, M. Lundgren, and L. Svensson, "Adaptive radar sensor model for tracking structured extended objects", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 3, pp. 1975–1995, 2012.

[60] M. Fatemi *et al.*, "Variational bayesian em for slam", in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2015 IEEE 6th International Workshop on*, IEEE, 2015, pp. 501–504.

[61] M. Lundgren, L. Svensson, and L. Hammarstrand, "Variational bayesian expectation maximization for radar map estimation.", *IEEE Trans. Signal Processing*, vol. 64, no. 6, pp. 1391–1404, 2016.

[62] R. E. Kalman *et al.*, "A new approach to linear filtering and prediction problems", *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[63] H. E. Rauch, F. Tung, C. T. Striebel, *et al.*, "Maximum likelihood estimates of linear dynamic systems", *AIAA journal*, vol. 3, no. 8, pp. 1445–1450, 1965.

[64] R. E. Kopp and R. J. Orford, "Linear regression applied to system identification for adaptive control systems", *Aiaa Journal*, vol. 1, no. 10, pp. 2300–2306, 1963.

[65] A. Gelb, *Applied optimal estimation*. MIT press, 1974.

[66] S. Julier and J. Uhlmann, "A new extension of the kalman filter to nonlinear systems", in *The Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls, Multi Sensor Fusion, Tracking and Resource Management II*, 1997.

[67] I. Arasaratnam and S. Haykin, "Cubature kalman filters", *Automatic Control, IEEE Transactions on*, vol. 54, no. 6, pp. 1254–1269, 2009.

[68] F. Gustafsson and G. Hendeby, "Some relations between extended and unscented kalman filters", *IEEE Transactions on Signal Processing*, vol. 60, no. 2, pp. 545–555, 2012.

[69] E. Wan and R. V. D. Merwe, "The unscented kalman filter for nonlinear estimation", *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, 2000.

[70] D. Reid, "An algorithm for tracking multiple targets", *IEEE transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[71] I. J. Cox and S. L. Hingorani, "An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking", *IEEE Transactions on pattern analysis and machine intelligence*, vol. 18, no. 2, pp. 138–150, 1996.

[72] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking", *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, no. 1, pp. 5–18, 2004.

[73] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation", in *IEE Proceedings F (Radar and Signal Processing)*, IET, vol. 140, 1993, pp. 107–113.

[74] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study", in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, Morgan Kaufmann Publishers Inc., 1999, pp. 467–475.

[75] M. Lundgren *et al.*, "Vehicle self-localization using off-the-shelf sensors and a detailed map", in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2014, pp. 522–528.

[76] E. Stenborg and L. Hammarstrand, "Using a single band gnss receiver to improve relative positioning in autonomous cars", in *Intelligent Vehicles Symposium (IV), 2016 IEEE*, IEEE, 2016, pp. 921–926.

[77] E. Stenborg, C. Toft, and L. Hammarstrand, "Long-term visual localization using semantically segmented images", *Unpublished*,