THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Long-Term Localization for Self-Driving Cars

ERIK STENBORG

CHALMERS

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2020

Long-Term Localization for Self-Driving Cars
ERIK STENBORG
ISBN 978-91-7905-377-2

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 4844
ISSN 0346-718X

Department of Electrical Engineering
Signal Processing Group
CHALMERS UNIVERSITY OF TECHNOLOGY
SE–412 96 Göteborg, Sweden

Typeset by the author using LaTeX.

Chalmers Digitaltryck
Göteborg, Sweden 2020

Abstract

Long-term localization is hard due to changing conditions, while relative localization within time sequences is much easier. To achieve long-term localization in a sequential setting, such as, for self-driving cars, relative localization should be used to the fullest extent, whenever possible.

This thesis presents solutions and insights both for long-term sequential visual localization, and localization using global navigational satellite systems (GNSS), that push us closer to the goal of accurate and reliable localization for self-driving cars. It addresses the question: How to achieve accurate and robust, yet cost-effective long-term localization for self-driving cars?

Starting in this question, the thesis explores how existing sensor suites for advanced driver-assistance systems (ADAS) can be used most efficiently, and how landmarks in maps can be recognized and used for localization even after severe changes in appearance. The findings show that:

- State-of-the-art ADAS sensors are insufficient to meet the requirements for localization of a self-driving car in less than ideal conditions. GNSS and visual localization are identified as areas to improve. (Paper I).
- Highly accurate relative localization with no convergence delay is possible by using time relative GNSS observations with a single band receiver, and no base stations. (Paper II).
- Sequential semantic localization is identified as a promising focus point for further research based on a benchmark study comparing state-of-the-art visual localization methods in challenging autonomous driving scenarios including day-to-night and seasonal changes. (Paper III).
- A novel sequential semantic localization algorithm improves accuracy while significantly reducing map size compared to traditional methods based on matching of local image features. (Paper IV).
- Improvements for semantic segmentation in challenging conditions can be made efficiently by automatically generating pixel correspondences between images from a multitude of conditions and enforcing a consistency constraint during training. (Paper V).
- A segmentation algorithm with automatically defined and more fine-grained classes improves localization performance. (Paper VI).
- The performance advantage seen in single image localization for modern local image features, when compared to traditional ones, is all but erased when considering sequential data with odometry, thus, encouraging to focus future research more on sequential localization, rather than pure single image localization. (Paper VII).

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 C. Toft, W. Maddern, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, T. Pajdla, F. Kahl, and T. Sattler, **"Long-Term Visual Localization Revisited"**, Manuscript submitted for review to IEEE Transactions on Pattern Analysis and Machine Intelligence

Paper IV E. Stenborg, C. Toft, and L. Hammarstrand, **"Semantic Maps and Long-Term Self-Localization for Self-Driving Cars using Image Segmentation"**, Manuscript to be submitted to IEEE Transactions on Robotics

Paper V M. Larsson, E. Stenborg, L. Hammarstrand, M. Pollefeys, T. Sattler, and F. Kahl, **"A Cross-Season Correspondence Dataset for Robust Semantic Segmentation"**, 2019 IEEE Conference on Computer Vision and Pattern Recognition, Proceedings pages 9532-9542

Paper VI M. Larsson, E. Stenborg, C. Toft, L. Hammarstrand, T. Sattler, and F. Kahl, **"Fine-grained segmentation networks: Self-supervised segmentation for improved long-term visual localization"**, 2019 IEEE International Conference on Computer Vision, Proceedings pages 31-41

Paper VII E. Stenborg, T. Sattler, and L. Hammarstrand, **"Using Image Sequences for Long-Term Visual Localization"**, Manuscript submitted to 2019 International Conference on 3D Vision

Preface

To get to this point where I'm now writing the last few words in my very own thesis, has been a special experience for me. I've been close to quitting numerous times, but something has pulled me back each time, and I don't know if it was a desire to learn more or simple stubbornness that was more to blame.

Anyway, I have learned a few things. I've learned not to be greedy and think that I have more than a 50-50 chance of getting the direction of rotation matrices right. I've learned, through extensive simulation on the computer cluster, that there are only two integers in the range 1 to 2. Thanks, Mikael @ C3SE, for helping me run things there. I've learned that I need more than 15 minutes to get from Göteborg to Södertälje, that booking a hotel room in the correct month can be a lot harder than it seems, and generally that making appointments is easier than keeping them. But maybe most of all, I've learned that the extent of my knowledge is quite limited. And when this limit is approaching, I've learned that I should ask people for advice. I was stubborn at first, but I think that I am now better at asking for help.

I want to thank my academic supervisors Lennart Svensson for guiding me through to the Licentiate, and Lars Hammarstrand for pulling me all the way here. Your advise, support, and help, that you are so generous with, has been essential for me to get here. A big thanks also to my industrial supervisor Joakim Sörstedt, my former manager Jonas Ekmark, and my current manager Mats Nordlund, for giving me this opportunity, believing in me, and supporting me through all these years. 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. Special thanks to Malin, for helping me getting started, to Carl, Måns, Torsten, and Fredrik for being such rocks and helping me the last few years, to Anders, Samuel, and Juliano for being fantastic office room mates. To all people who helped me producing this book, whether co-authoring or proof reading: Thank you all!

The Dudes, Magellanists, and 3D-recon people at Zenuity have all endured my silly questions at some point or another. Thank you for that! And, brace yourselves - you'll be seeing a lot more of me from now on.

Finally, also a warm thanks to family and friends, for pulling me out from work and reminding me of the world outside. Yoko, I'm very grateful that you are here with me. Lina och Hanna, jag är så lycklig att ni finns hos mig, och stolt över allt ni gör. Fortsätt så. Love you all!

Erik Stenborg,
Göteborg, 2020

Contents

II Included Papers

Part I

Introductory Chapters

Chapter 1

Introduction

Autonomous vehicles have several advantages to vehicles that must be driven by humans, one being increased safety. Despite an increase in number of cars and total distance driven, traffic injuries are declining in Sweden [69] and the USA [49], from being the leading cause of death in age groups below 45, to "merely" being in the top ten [50]. 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 the one from NHTSA [62], 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 remaining control needs in the car, e.g. flow of fuel and air to the engine, are 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 that hu-

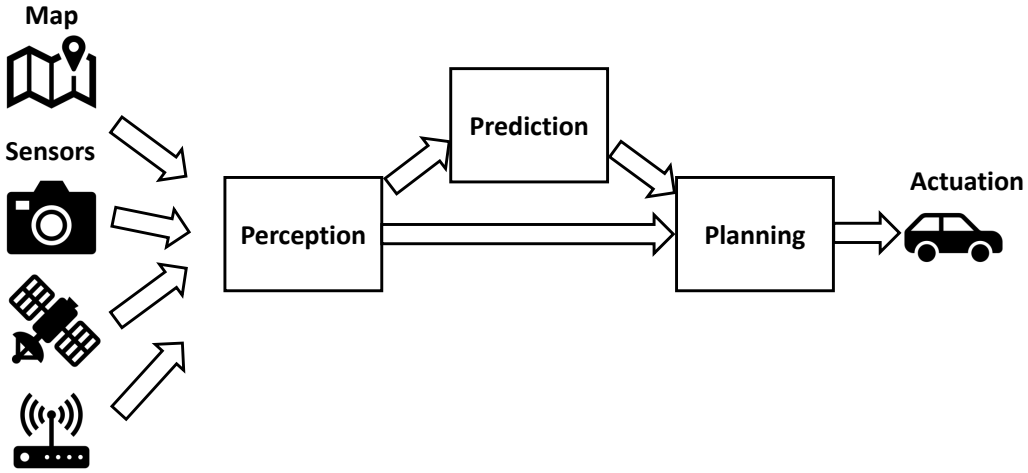

Figure 1.1: Flow of data and processing modules for autonomous driving. Localization is here considered part of the perception module.

mans 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, 45, 55], 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 divided into a few major modules, where Figure 1.1 shows one example of such a division. We have one module that is responsible for planning a trajectory and following it based on information from the other modules, another that interprets the sensor inputs into a simple structure that is meaningful for the driving task, and a third that 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 relative to the map. Any uncertainty in position and orientation will translate into uncertainty about the drivable area and in consequence also the planned path. Thus, we can see that accurate localization in relation to an accurate map of the close surroundings of the car, is an enabler for autonomous driving.

So, what is needed to solve effective localization for autonomous vehicles? Firstly, 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 been included in Chapter 2. Secondly, we need sensors capable of observing the landmarks, and sensor models that 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 can be achieved using different types of automotive class sensors. It further establishes a few missing pieces, particularly in the area of visual localization, but also relating to GPS, in reaching the desired performance, and proposes solutions for some of them. This is done in seven 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, and finish by having a look at what should go in the map.

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., [68, 70].

	Topological	Metric
Single shot	[10, 52, 68, 70]	[26, 37, 59, 60, 64]
Sequential	[2, 44, 48]	[12, 46, 61, 67, 72]

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

Specialized loop closure detectors also belong in this category, see e.g., [10, 52]. 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 focus on robust localization in conditions with large visual variations, see e.g., [44, 48]. 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 [2], 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., [37, 38, 59, 64], but also examples where a form of topological localization is performed first as an initial step, see e.g., [26, 60]. Most localization in robotics, with the purpose of providing navigational information to a robot, requires metric localization, with some examples in [12, 46, 67]. Localization using global navigational satellite 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., [10, 26, 37, 38, 59, 60, 64, 68, 70], and sometimes also when using sensors that are specialized for localization, such as the global navigation satellite systems, see e.g., [8].

Sequential localization means using a sequence of observations and some motion model to connect them. This is the type of observations available for on-line robot navigation, see e.g., [12, 44, 46, 48, 61, 66, 67, 71, 72]. 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.

In sequential metric localization, there is a possibility (and to some extent also a necessity) to use single shot localization towards a "global" reference to solve the problem. However, using sequences and motion models makes the problem easier to solve, compared to the single shot problem, and this should be taken advantage of for increased precision and robustness. For some sensors there is an obvious way of using them for localization. For example, global navigational satellite systems provide a global position which is well suited for

absolute single shot localization. But if looking a little under the hood, there is also a way to use it that provides high accuracy time relative localization, as is shown in Paper II.

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., [66], and the DARPA Urban Challenge in 2007, see e.g., [71]. 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. [3, 7, 11] for surveys 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 to keep it updated. In this thesis, the focus is on the localization problem, but for the experiments maps have been built, and then SLAM methods have been most useful. And in a longer perspective, to keep maps updated it may be useful to view also the localization problem as SLAM and thus incorporate updates to the map in the localization process when needed.

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. Occupancy grid maps discretize the world in a 2-D or 3-D grid, and measure how opaque [14, 53] or reflective [36] each pixel (or voxel in the 3-D case) is to the relevant sensor. Despite efforts to compress them [25], this dense type of maps requires a lot of storage.

For 3-D maps covering large areas or volumes, 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 in a global frame of reference, and hence, 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 are many different projections that minimize various distortions, 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 be able to store a 3-D representation of landmarks and the road, in any part of the world without discontinuities.

Arguably, 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 the 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 approximation, is sufficient.

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, there are also other sensors that can be quite useful for localization. Although primarily used for object detection, cameras and radars are examples of this type of sensors that are also useful for localization. Together with GNSS, and motion sensors such as accelerometers, gyroscopes and wheel speed sensors, they provide the measurements that we use for localization in this thesis.

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.

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.

All GNSS systems, including the most well known, GPS, use the same prin-

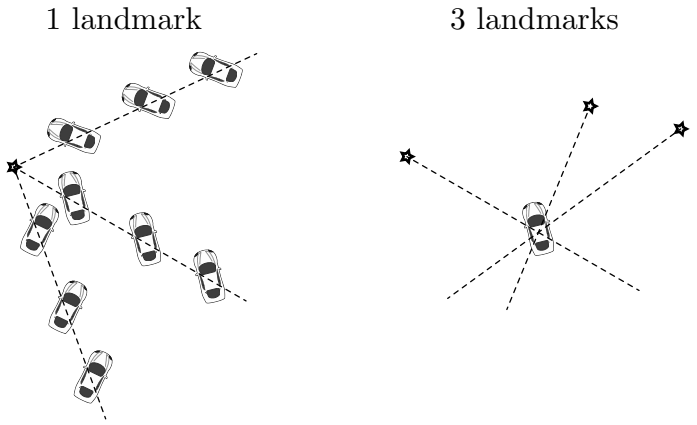

Figure 3.1: Triangulation from landmarks with known positions. With only one landmark, there is no unique solution, but with three landmarks we get a unique solution in 2-D.

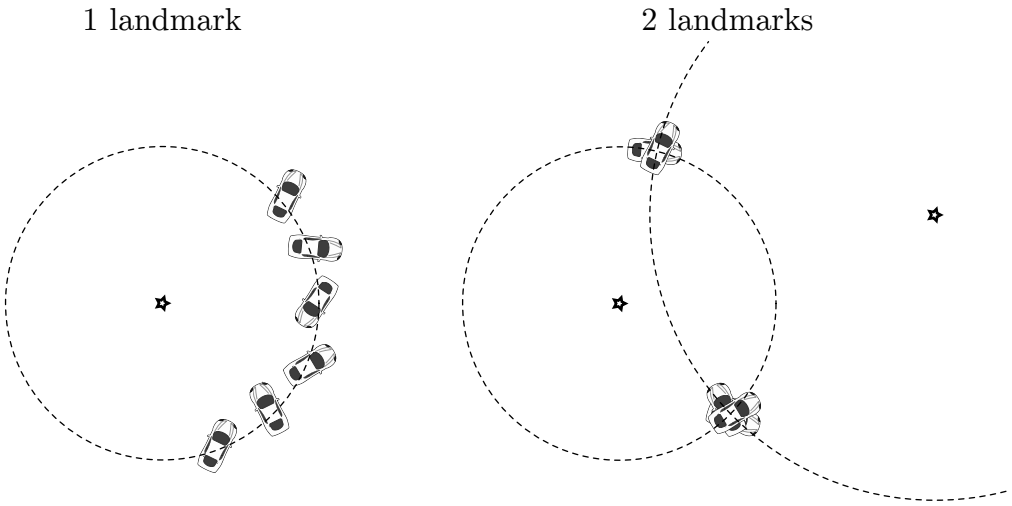

Figure 3.2: Trilateration from landmarks with known positions. With only one landmark, again there is no unique solution, and with two landmarks we get two possible positions in 2-D, but no information about orientation.

ciple of positioning. Kaplan and Hegarty describe this in detail in their book [29]. 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 lower 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 measures 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.

Accuracy of the estimated position depends on noise and bias in the signal. There is a noise floor of around 1% of the bit length in the C/A code, which is difficult to get below. The bit length expressed in meters (calculated as the bit length in seconds times the speed of light) is around 300 m, meaning that the receiver tracking loop introduces noise on the pseudo range measurements that has a standard deviation of about 3 m. However, since modern receivers can use some tricks, such as carrier phase smoothing [23], 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.

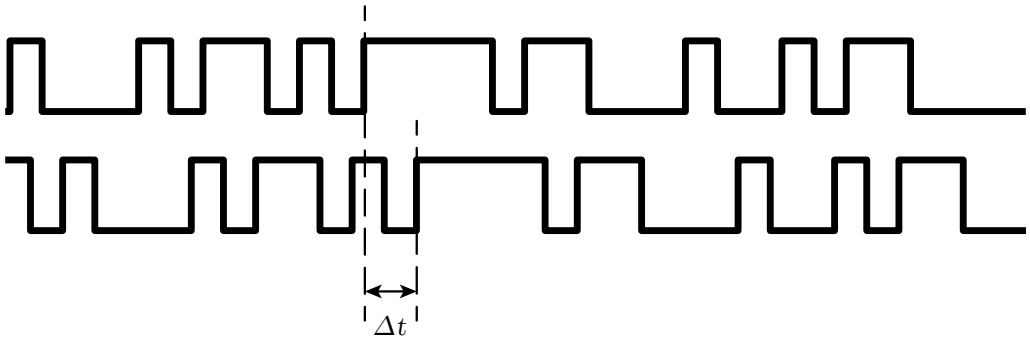

Figure 3.3: The time offset, Δt, when matching pseudo random numbers in the received C/A code signal to the internally generated sequence, is unique.

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.

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

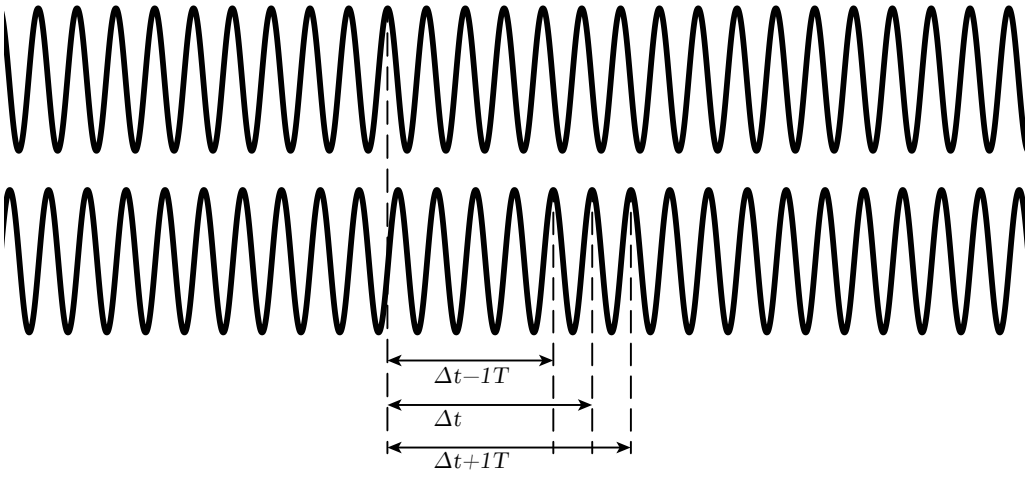

Figure 3.4: The true time offset, Δt, when matching the received carrier signal to the internal oscillator, is not uniquely observable.

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) [65], is one popular method to resolve the integer ambiguity.

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, 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, radiation from the Sun creates electrically charged particles that delay the signal. This

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
Multi-path	N/A	0.2	0.01
Total	N/A	7.1	0.2

Table 3.1: Standard deviation of user equivalent range errors in meters for a typical consumer receiver using one frequency band [29], and a high end, dual frequency receiver using corrections for satellite errors.

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. 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.

Finally, there is multi-path error, which is a local error caused by the signal reflecting from surfaces near the receiver. In contrast to the other error sources above, the multi-path error 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 [54].

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 sys-

tem, in the form of coefficients to Klobuchar's ionospheric model [31], 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, the IGS corrections tend to be used mostly 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 at 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 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

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.

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.

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.

Camera projection model

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. To be able to use camera images in our localization process, we need a model which describes how the camera measurement, comprising the image pixels, relates to angles of the incoming light from the observed objects. The geometric relation between a light source in 3-D and the pixel on the flat image sensor that captures the light, is modeled by the pinhole camera model in conjunction with a simple non-linear

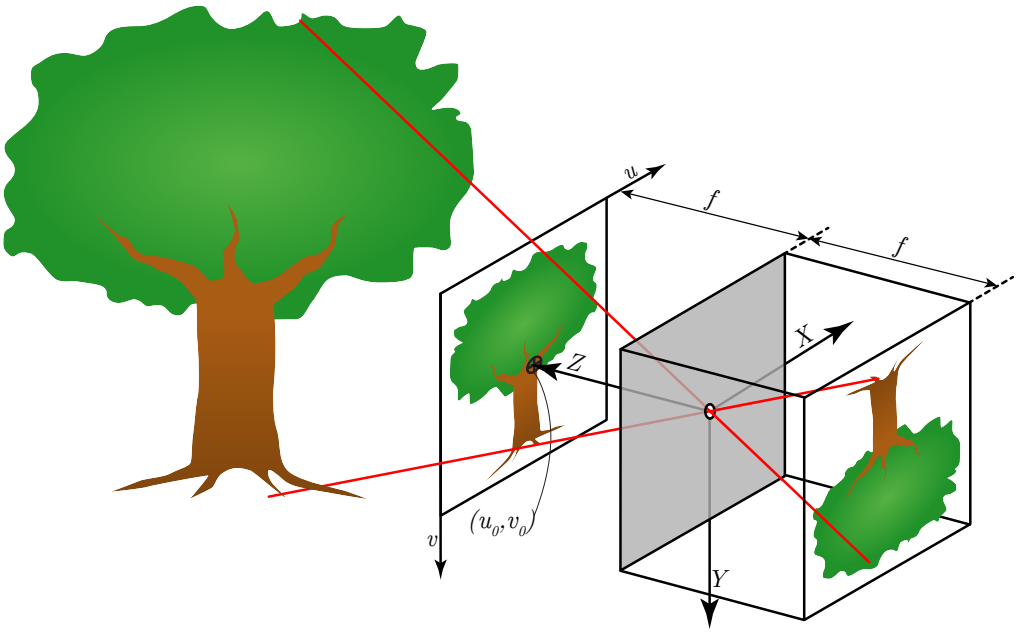

Figure 3.5: Pinhole camera projection of two points from an object, with image plane coordinate system (X, Y, Z), camera coordinate system (u, v), focal length (f), and principal point (u_0, v_0) marked.

model for distortion.

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 Figure 3.5. For convenience, we most often pretend that the image plane lies in front of the focal point, the leftmost projection in Figure 3.5, to get an image that is not upside down, and thus get rid of the negation, $[fX/Z, fY/Z]^\top$. The image coordinate frame of digital cameras 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 camera coordinates to world 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.

For real cameras using optical lenses, the pure 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. A very popular distortion model, and also the one used in this thesis, was described by Brown [6], comprising two components: a radial distortion (3.6), and a decentering distortion (3.7),

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

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

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

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

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

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

Here $\hat{\mathbf{u}}$ 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 [24, 75]. 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 Papers IV and VII.

Image features and semantic segmentation

With the geometric camera model, we have a modeled how light rays travel from objects in the world and shine on pixels in the camera. There are methods to solve the localization problem that directly apply this model to the intensity values of the pixels, in combination with a photometric calibration that relate the pixel values to brightness in the world [15, 18]. However, more often the raw image pixels are pre-processed into some type of image features. Most common feature extractors detect salient feature points, such as corners or local extrema in intensity. To enable matching of feature points between images, a small region of the image around each point is processed into a descriptor vector that describes its appearance. The idea with this vector is that when a feature of the same object is detected in two different images, their descriptor vectors will be very similar. The canonical example of such feature detector and descriptor is SIFT [40], but there are newer alternatives, e.g. ORB [58] that prioritizes speed, and learning based D2-Net [13] that prioritizes matching performance.

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.

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 approximation, and knowledge about the mounting position of the camera on the vehicle. In Paper I we use lane marker descriptions from a camera of this type.

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, as shown in Figure 3.6. For this type of feature, there is no easy way to define a measurement model. Instead, the problem is often defined as a supervised learning task. If Y is the observed image, and X is the underlying semantic class for each pixel in the image, then we want to find a function $f_\theta(Y) = X$.

Figure 3.6: (Left) Example of an image from a road environment and its semantic segmentation. (Right) The same image colorized according to class by a semantic segmentation algorithm.

This is typically a very flexible model with lots of model parameters, θ, that must be learned. To do that, supervised learning is often used, meaning that people manually annotate lots of example images with the correct class, and then give this as input to a a training process in which (locally) optimal parameters, $\hat{\theta}$, for that training set are found. More on how this training is done, comes in Section 4.

In 2015, the first successful semantic segmentation algorithms based on neural networks were demonstrated [39, 51, 74], and in the five years since then, the progress has been dramatic.

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, a detection from a radar contains 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., [17, 22, 41, 43].

Chapter 4

Parameter estimation

Parameter estimation is central to almost all scientific endeavors, and localization is no different. It 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 location of the self-driving car is continuously estimated in an iterative process that takes the latest measurements and weighs it against previous knowledge gotten from past measurements. This process, and various ways of implementing it, is described more in section 1 on Bayesian filtering and smoothing.

To build the maps that are needed for localization, the parameters for the map generally do not need to be estimated in real time. Some particular details on this process are described in more detail in section 2 on Smoothing and mapping.

In the examples given above, the unknown location of the car or landmarks were the parameters that we were interested in estimating. There are usually also a number of nuisance parameters that we are not really interested in knowing, but that we need to estimate nevertheless. For example, inside the sensor models that we use, there are parameters, and although we do not necessarily want to use them for the driving task, they are necessary to know for the sake of estimating location.

The method for determining those nuisance parameters varies depending on the nature of the problem. For example, the sensor model for the GNSS receiver contains a few parameters that are reasonably straight-forward for a human to estimate by doing certain experiments. However, some other sensor models, such as the model for semantic pixel-wise labeling of camera images, are very complex, and use millions of parameters. The process often used to determine these parameters is a recent development in parameter estimation called deep learning. A very brief introduction to deep learning and some related concepts that are used in this thesis are found in section 3.

1 Bayesian filtering and smoothing

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.

1.1 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.

1.2 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 [28]. 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 [56], starting from $k = T-1$ as

$$\mathbf{G}_k = \mathbf{P}_{k|k}\mathbf{F}^\top\mathbf{P}_{k+1|k}^{-1} \tag{4.21}$$
$$\hat{\boldsymbol{\mu}}_{k|T} = \hat{\boldsymbol{\mu}}_{k|k} + \mathbf{G}_k(\hat{\boldsymbol{\mu}}_{k+1|T} - \hat{\boldsymbol{\mu}}_{k+1|k}) \tag{4.22}$$
$$\mathbf{P}_{k|T} = \mathbf{P}_{k|k} + \mathbf{G}_k(\mathbf{P}_{k+1|T} - \mathbf{P}_{k+1|k})\mathbf{G}_k^\top. \tag{4.23}$$

1.3 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) [19, 32] 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) [27] and the Cubature Kalman Filter (CKF) [1]. 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 [21]. 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 [73].

1.4 Particle filters

Both the EKF and UKF approximates the posterior density as a normal density. Sometimes this approximation is 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 [4, 9, 57], 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 IV, called the bootstrap particle filter [20], 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.

1.5 Maximum á-posteriori filters

For both the Kalman filters and the particle filters the cost function they aim to minimize is the expected error of the estimate. This is often the most reasonable thing to aim for, but there are other popular alternatives. A notable example is the maximum likelihood estimation (MLE), or the maximum á-posteriori (MAP) estimation in case we are considering a prior distribution. These methods find the maximally likely solution, i.e. the mode of the distribution, instead of the mean as for previous filters. It is possible to construct degenerate densities that exhibit unwanted behaviour for either choice, see Fig. 4.1, but for nicely behaved problems with uni-modal densities, the two estimates are often rather similar.

The solution to the MAP filter is usually found through a local optimization procedure, where the prediction from the previous time step is used as an initial guess, and then an iterative local optimization algorithm is applied to the negative logarithm of the posterior density function. When the gradient of the density function is straight forward to compute, the Levenberg-Marquardt method is a popular choice, whereas a gradient free method, such as the simplex algorithm, may be used if the gradient is hard to compute [30].

A benefit of the iterative MAP filters over the direct solutions used in the non-linear Kalman filters, is that the linearization point is improved for each iteration and ends up at the ideal spot if the iteration converges, whereas it may lie arbitrarily far away for the Kalman filters [16]. This property is a major reason why MAP filters and smoothers are preferred over Kalman type filters to solve certain types of problems where large errors may be present in the initial estimates, e.g. the simultaneous localization and mapping (SLAM) problem. This is why a MAP smoother is employed in Paper VII.

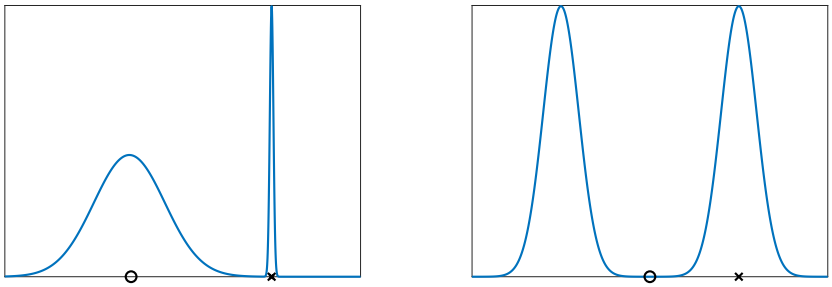

Figure 4.1: Examples of degenerate distributions. Minimum mean squared error estimate is marked with a circle, and maximum á-posteriori estimate with a cross. To the left 90% of the probability mass is around the left peak, yet, because the right peak reaches a higher maximum value, it is the MAP estimate. To the right the MMSE estimate ends up in a spot where there is almost no probability, yet it is the best trade-off with the smallest average squared error to the two peaks at the sides.

The MAP filter is also used in Paper IV, but here another feature plays a big role. The non-linear Kalman filters assume a measurement model that is reasonably easy to compute, and that the measurement covariance (4.17) and (4.32) can be formed. For very high dimensional measurements, such as a megapixel images, this not the case anymore. Yet, there may be efficient ways to evaluate the likelihood, and for particle filters and MAP filters that is the only requirement, which makes them suitable in such circumstances.

2 Smoothing and mapping

Bayesian smoothing of a Markov process can be done optimally in a forward pass followed by a backward pass. However, for a typical mapping problem or a localization problem that also does mapping, the Markov property does not hold. When a landmark is observed from multiple locations, those locations become related through the landmark, and it is no longer possible to say that the current position is only dependent on the previous (and next, in the case of smoothing) position and the current measurement.

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 observations, $y_1, \ldots, y_4$, of the landmarks from four different positions,

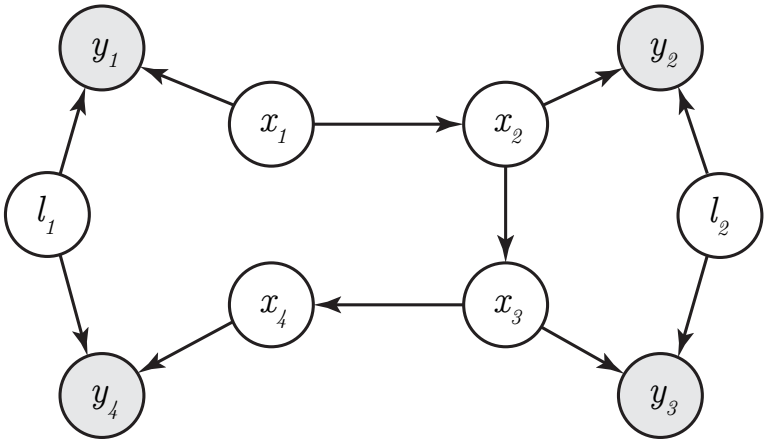

Figure 4.2: Bayes net representation of a small SLAM problem with 4 robot poses, 2 landmarks, and 4 observations. Unknown variables are white, while given measurements are marked gray.

$x_1, \ldots, x_4$. 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 4.2. 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. One way out of this is to use iterative methods such as loopy belief propagation [47] that makes many passes backwards and forwards. Another possibility is to reformulate it as a maximum á-posteriori (MAP) problem and consider the whole problem (or at least a much larger window than just the current time slot) at once. When the noise in the models is additive and Gaussian, which is an often used approximation, the MAP problem is equivalent to a weighted, non-linear, least squares problem,

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

where $||x||^2_\Sigma = x^\top \Sigma^{-1} x$. 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.

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 minima, but with good initialization, this is less of a problem.

3 Deep learning

Deep learning commonly refers to the practice of training and using artificial neural networks. Artificial neural networks are models made up of a set of layered functions (more than 3 layers to be considered "deep") with the goal of approximating a function that transforms an input into an output of some specified type. See review by LeCun et al. [35], for an overview of the field at the time. An example task, used frequently in this thesis, is to take an image as input and transform it into a semantic segmentation, i.e. each pixel in the image is transformed into a class label (look back at Figure 3.6 for an example).

The task at a high level is described as $f_\theta(Y) = X$, where Y is the observation, X is the output and θ are the model parameters. The whole network consists of multiple layers (or functions) where the output from an early layer is provided as input to a later layer. Each layer typically consists of a linear part where the input is treated as a vector and multiplied by a weight matrix and added with a bias vector to generate a linear combination of the input as intermediate output, Z_k. Before the output is passed on to the next layer, a non-linear function, $\rho()$, is applied to the intermediate vector.

$$Z_k = W_k X_{k-1} + b_k \tag{4.48}$$
$$X_k = \rho(Z_k) \tag{4.49}$$
$$X_0 = Y \tag{4.50}$$
$$X = X_n \tag{4.51}$$
$$\theta = (W_1, b_1, \ldots, W_n, b_n) \tag{4.52}$$

Without the non-linear function, $\rho()$, the whole system could be reduced to a single linear expression, which is too restrictive for the often highly non-linear functions that are desired. A common choice for the intermediate layers is

$$\rho(z_{k,i}) = \begin{cases} z_{k,i}, & \text{if } z_{k,i} > 0 \\ 0, & \text{otherwise,} \end{cases} \tag{4.53}$$

where $z_{k,i}$ denotes the i^{th} element in the intermediate vector, Z_k. However, for the last layer that produces the final output, often some other function is used. For regression tasks, the last non-linearity is often removed, and for for classification tasks a popular choice is to use the soft-max function,

$$\rho(z_{n,i}) = \frac{e^{z_{n,i}}}{\sum_j e^{z_{n,j}}}. \tag{4.54}$$

This way the output becomes a normalized vector that sometimes is interpreted as a probability mass function over the classes.

We can see already at this level of abstraction, that beside the main parameters, θ, there are a number of hyper-parameters that have to do with number of layers, dimensionality of each layer, and similar things. These parameters determine the architecture of the network, and ought to be optimized over as well, but partly due to the time it takes to train the regular parameters and partly due to their discrete nature, it is impractical to run an outer optimization loop over hyper parameters.

3.1 Convolutional neural networks

There is a wide variety of ways to connect the layers above. For certain problems, simply connecting multiple layers where each output in each layer is affected by all outputs from the previous layer ("dense network"), is appropriate. For recursive tasks, there exist designs where layers connect back in a recursive manner.

However, for image inputs, where the pixels are regularly ordered in a grid, and also quite numerous, it is common to use a particular type of artificial neural network called "convolutional neural network" (CNN). There are two justifications for using CNN:s: First, for megapixel images, the huge number of pixels used as individual inputs, make a deep network untractable on current hardware. Second, the idea that we want the net to react in a similar way to the same subpattern in the image regardless of its location in the image. These two problems led to this particular type of network.

Instead of feeding the whole input into one big network, the idea is to use a smaller network and apply it to a small window of the image, and then sweep

this window over the whole image. The output from each window is stored in what is called "activation maps", which have the same spatial (x,y) shape as the input. One can think of it in terms of a filter bank of weight kernels that the input image is convoluted with. This design solves both the size problem and the spatial covariance problems, as each small network kernel is reused for the whole image, and thus the weights are both much fewer and also used in the same way in all corners of the input image.

A common design of CNN for image segmentation is to layer several of these convolutional layers after each other, and between some of them put a special type of layer that resamples the feature maps to either decrease or increase their spatial dimensions. Between the first few layers, the size is reduced, and between the last few layers, the size is increased back to the original size. This lets the network work on both fine details in the first and last layers, and at a higher, more coarse level in the middle layers. See, for example, the works from 2015 [39, 51, 74], when neural networks started to be successfully used for semantic segmentation.

3.2 Loss function and minimization

With a determined architecture, estimation of the model parameters, θ, in supervised training, when there are annotated examples to learn from, is done by minimizing a loss function defined from the training examples

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}(\mathbf{X}, \mathbf{Y}, \theta) \tag{4.55}$$

$$\mathcal{L}(\mathbf{X}, \mathbf{Y}, \theta) = \frac{1}{N} \sum_{i=1}^{N} l(X_i, f_\theta(Y_i)). \tag{4.56}$$

Here, $\mathbf{X} = \{X_i\}_{i=1}^N$ denotes the set of all labels in the training set, and $\mathbf{Y} = \{Y_i\}_{i=1}^N$ denotes the set of all the input data corresponding to $\mathbf{X}$, meaning the parameters are estimated by minimizing the (empirical) mean loss of the training examples.

Depending on the task at hand, the loss function for each sample is chosen accordingly. For regression tasks, the squared error is a popular choice of loss function, and for classification tasks, such as semantic segmentation, where the last layer is a soft-max layer, a popular choice of loss function is the cross-entropy loss

$$l(X, f_\theta(Y)) = -X^T \log f_\theta(Y). \tag{4.57}$$

The cross entropy loss emerges when trying to get the distribution $p(f_\theta(Y)|Y)$ to be as close to that of $p(X|Y)$ as possible.

This minimization problem is usually solved by local optimization using a version of gradient descent, that approximates the full gradient, which is dependent on all samples from the training data, by an approximation calculated from only a few samples. The approximated gradient is much faster to compute, but the price for that is that it is noisy. The noise, however, can be handled by gradually decreasing the step length of the optimizer.

3.3 Overfitting

From trying to fit a polynomial curve through a number of data points, it is obvious that by increasing the number of parameters (i.e. degree of the polynomial), the curve gradually fits the data better and better. When increasing the number of model parameters such that the degrees of freedom of the curve equals or is greater than the degrees of freedom of the data points, the fit is perfect with no residual error. However, generalization of the model to unseen data tends to be quite poor in that case, compared to if fewer model parameters are used. This phenomena of overfitting model parameters to data would seem to be a huge problem if the model contains millions or even billions of parameters.

Indeed, it is a problem for deep learning, and there are various counter measures for it. The first one is to detect when it happens. This is usually done by reserving some of the manually annotated data as validation data that is not used in the optimization of $\hat{\theta}$. By comparing model error on the training data and on the validation data, it is possible to detect when over fitting occurs. If detected, there are various heuristics to counter the effect, which are often described as different types of regularization that artificially decrease the degrees of freedom of the model. However, the best way to combat overfitting is to get more training data.

Since manual annotation of huge amounts of data can be rather costly, often one tries to expand the training data set by some automatic process. For example when using images, a common strategy is to slightly modify the input image by cropping, flipping, rotating, adjusting colors, adding noise, etc, and assuming that after applying the same cropping etc on the labels, they should still match the image content and be usable for training. In Paper V, we present another idea on how to easily expand the training data set with automatically generated data. This idea applies specifically to the problem of image segmentation in changing conditions.

Chapter 5

Contributions

Paper I: Vehicle self-localization using off-the-shelf sensors and a detailed map [42]

We investigate localization performance when using a particular setup of low cost, automotive grade sensors for ADAS 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 pre-processing 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 gives rise to large errors if assumed to be correct when they are not.

All work for this paper was shared between the first author and me, under the supervision of the two last authors. The first author was responsible for the radar sensor model, while I was responsible for the camera sensor, and motion models.

Paper II: Using a single band GNSS receiver to improve relative positioning in autonomous cars [63]

An alternative configuration of GNSS receiver is proposed. 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.

All work, including models, implementation and writing for this article was done by me under supervision of the second author.

Paper III: Long-Term Visual Localization Revisited

We present a benchmark with three different data sets to evaluate performance of visual localization algorithms. A number of both historic and state of the art algorithms are evaluated and compared to each other. This article was written when the benchmark had been available for over a year, and we can see the development in the area from when the benchmark was first released. The overall conclusion is that, although there has been great progress in visual localization, the problem is not considered solved. One thing we note is that localization using several images in sequence has great potential to improve results, yet there has been surprisingly little research in this direction. The CMU dataset is larger, in terms of number of images, than the other two data sets combined, and has higher estimated accuracy in the ground truth poses. Of the three, it is also the only dataset that can be used to evaluate sequential localization over longer sequences.

My contribution was to produce the ground truth poses for and write parts about the CMU Seasons dataset, and evaluation and writing for the sequential localization algorithms.

Paper IV: Semantic Maps and Long-Term Self-Localization for Self-Driving Cars using Image Segmentation

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. Resilience towards changing appearance was partially achieved, and it is 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, models, implementation, and writing was primarily by me with good help from second author for the idea and evaluation, under supervision of the last author.

Paper V: A Cross-Season Correspondence Dataset for Robust Semantic Segmentation [34]

Another use for the ground truth poses from the localization benchmark data sets is found. We show that this data can be used to train consistency over different conditions, and thus greatly reduce the amount of manual labor needed for state of the art segmentation performance.

My contribution includes creating one of the underlying data sets and contributing to the idea of using it for consistency training. Most work on models and implementation was done by the first author. I contributed some parts to the implementation and evaluation phases, e.g. the patch interpolation, additional ground truth segmentations for our data set, code for efficiently training and evaluating on a computer cluster. Writing was primarily done by the first author.

Paper VI: Fine-Grained Segmentation Networks: Self-Supervised Segmentation for Improved Long-Term Visual Localization [33]

From the earlier conclusion in Paper IV, that more classes are needed in some areas to provide better performance, a solution is proposed here. We choose to drop the semantic meaning of the predefined classes, and to instead use a variable number of classes without semantic meaning, but that are still consistent over long time periods. The main contribution is how to train these types of networks and to show that it does indeed lead to improved performance for localization algorithms based on image segmentations.

My work from Paper IV clearly pointed towards the need for more fine grained classes, and I was involved from an early stage in the motivations for the work and was involved in the discussions on how to realize it along the way. In the later stages I trained and evaluated the different networks to find the one that performs best for sequential localization. I wrote parts about visual localization from introduction to evaluation.

Paper VII: VoLoc: Image sequences beat single images every day (and night)

We pick up the thread from Paper III, about using multiple images in sequence to improve localization performance. Instead of using semantic description of the environment, we here use more traditional local image features that give rise to point correspondences between image and map. We show that if decent odometry data is available, this significantly improves performance to the point where the advantage that modern visual localization algorithms have over a method based on traditional local image features in single image performance, is all but

erased.

I have been involved in the discussion that led into this avenue of research, responsible for the implementation of the coarse localization smoother, the odometry based smoothing, and making modifications in the SLAM-solver, as well as evaluating results, and most of the writing.

Bibliography

[1] Ienkaran Arasaratnam and Simon Haykin. “Cubature kalman filters”. In: *IEEE Transactions on automatic control* 54.6 (2009), pp. 1254–1269.

[2] Hernán Badino, D Huber, and Takeo Kanade. “Visual topometric localization”. In: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE. 2011, pp. 794–799.

[3] Tim Bailey and Hugh Durrant-Whyte. “Simultaneous localization and mapping (SLAM): Part II”. In: *IEEE Robotics & Automation Magazine* 13.3 (2006), pp. 108–117.

[4] Samuel S Blackman. “Multiple hypothesis tracking for multiple target tracking”. In: *IEEE Aerospace and Electronic Systems Magazine* 19.1 (2004), pp. 5–18.

[5] Mariusz Bojarski et al. “End to end learning for self-driving cars”. In: *arXiv preprint arXiv:1604.07316* (2016).

[6] Duane C. Brown. “Close-range camera calibration”. In: *Photogramm. Eng* 37.8 (1971), pp. 855–866.

[7] Cesar Cadena et al. “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age”. In: *IEEE Transactions on robotics* 32.6 (2016), pp. 1309–1332.

[8] SJ Corbett and PA Cross. “GPS single epoch ambiguity resolution”. In: *Survey Review* 33.257 (1995), pp. 149–160.

[9] Ingemar J. Cox and Sunita L. Hingorani. “An efficient implementation of Reid’s multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking”. In: *IEEE Transactions on pattern analysis and machine intelligence* 18.2 (1996), pp. 138–150.

[10] Mark Cummins and Paul Newman. “FAB-MAP: Probabilistic localization and mapping in the space of appearance”. In: *The International Journal of Robotics Research* 27.6 (2008), pp. 647–665.

[11] Hugh Durrant-Whyte and Tim Bailey. “Simultaneous localization and mapping: part I”. In: *IEEE robotics & automation magazine* 13.2 (2006), pp. 99–110.

[12] Hugh Durrant-Whyte, David Rye, and Eduardo Nebot. “Localization of autonomous guided vehicles”. In: *Robotics Research*. Springer, 1996, pp. 613–625.

[13] Mihai Dusmanu et al. “D2-Net: A Trainable CNN for Joint Description and Detection of Local Features”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8092–8101.

[14] Alberto Elfes. “Sonar-based real-world mapping and navigation”. In: *IEEE Journal on Robotics and Automation* 3.3 (1987), pp. 249–265.

[15] Jakob Engel, Thomas Schöps, and Daniel Cremers. “LSD-SLAM: Large-scale direct monocular SLAM”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 834–849.

[16] Maryam Fatemi et al. “A study of MAP estimation techniques for non-linear filtering”. In: *2012 15th International Conference on Information Fusion*. IEEE. 2012, pp. 1058–1065.

[17] Maryam 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.

[18] Xiang Gao et al. “LDSO: Direct sparse odometry with loop closure”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 2198–2204.

[19] Arthur Gelb. *Applied optimal estimation*. MIT press, 1974.

[20] Neil J Gordon, David J Salmond, and Adrian FM Smith. “Novel approach to nonlinear/non-Gaussian Bayesian state estimation”. In: *IEE Proceedings F (Radar and Signal Processing)*. Vol. 140. 2. IET. 1993, pp. 107–113.

[21] Fredrik Gustafsson and Gustaf Hendeby. “Some relations between extended and unscented Kalman filters”. In: *IEEE Transactions on Signal Processing* 60.2 (2012), pp. 545–555.

[22] Lars Hammarstrand, Malin Lundgren, and Lennart Svensson. “Adaptive radar sensor model for tracking structured extended objects”. In: *IEEE Transactions on Aerospace and Electronic Systems* 48.3 (2012), pp. 1975–1995.

[23] Ron Hatch. “The synergism of GPS code and carrier measurements”. In: *International geodetic symposium on satellite doppler positioning*. Vol. 1. 1983, pp. 1213–1231.

[24] Janne Heikkila and Olli 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.

[25] Armin Hornung et al. “OctoMap: An efficient probabilistic 3D mapping framework based on octrees”. In: *Autonomous Robots* 34.3 (2013), pp. 189–206.

[26] Arnold 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.

[27] 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.

[28] Rudolph Emil Kalman et al. “A new approach to linear filtering and prediction problems”. In: *Journal of basic Engineering* 82.1 (1960), pp. 35–45.

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

[30] Carl T Kelley. *Iterative methods for optimization*. SIAM, 1999.

[31] John Klobuchar et al. “Ionospheric time-delay algorithm for single-frequency GPS users”. In: *Aerospace and Electronic Systems, IEEE Transactions on* 3 (1987), pp. 325–331.

[32] Richard E Kopp and Richard J Orford. “Linear regression applied to system identification for adaptive control systems”. In: *Aiaa Journal* 1.10 (1963), pp. 2300–2306.

[33] Mans Larsson et al. “Fine-grained segmentation networks: Self-supervised segmentation for improved long-term visual localization”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 31–41.

[34] Måns Larsson et al. “A Cross-Season Correspondence Dataset for Robust Semantic Segmentation”. In: *CVPR*. 2019.

[35] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[36] Jesse Levinson and Sebastian 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.

[37] Yunpeng Li, Noah Snavely, and Daniel P Huttenlocher. “Location recognition using prioritized feature matching”. In: *European conference on computer vision*. Springer. 2010, pp. 791–804.

[38] Yunpeng Li et al. “Worldwide pose estimation using 3d point clouds”. In: *Large-Scale Visual Geo-Localization*. Springer, 2016, pp. 147–163.

[39] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.

[40] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[41] Malin Lundgren, Lennart Svensson, and Lars Hammarstrand. “Variational Bayesian Expectation Maximization for Radar Map Estimation.” In: *IEEE Trans. Signal Processing* 64.6 (2016), pp. 1391–1404.

[42] Malin 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.

[43] Christian Lundquist, Lars Hammarstrand, and Fredrik Gustafsson. “Road intensity based mapping using radar measurements with a probability hypothesis density filter”. In: *IEEE Transactions on Signal Processing* 59.4 (2011), pp. 1397–1408.

[44] Michael J Milford and Gordon 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.

[45] Urs Muller et al. “Off-road obstacle avoidance through end-to-end learning”. In: *Advances in neural information processing systems*. 2006, pp. 739–746.

[46] Ana Cris Murillo, José Jesús 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.

[47] Kevin P Murphy, Yair Weiss, and Michael 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.

[48] Tayyab Naseer et al. "Robust visual robot localization across seasons using network flows". In: *Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014.

[49] National Highway Traffic Safety Administration. *Traffic Safety Facts 2015*. Tech. rep.

[50] National Highway Traffic Safety Association. *Motor Vehicle Traffic Crashes as a Leading Cause of Death in the United States, 2012-2014*. Tech. rep. 2016.

[51] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. "Learning deconvolution network for semantic segmentation". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1520–1528.

[52] Edwin Olson. "Recognizing places using spectrally clustered local matches". In: *Robotics and Autonomous Systems* 57.12 (2009), pp. 1157–1172.

[53] Daniel Pagac, Eduardo Mario Nebot, and Hugh Durrant-Whyte. "An evidential approach to map-building for autonomous vehicles". In: *IEEE Transactions on Robotics and Automation* 14.4 (1998), pp. 623–629.

[54] Mark Petovello. "The differences in differencing". In: *Inside GNSS* (Sept. 2011), pp. 28–32.

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

[56] Herbert E Rauch, F Tung, Charlotte T Striebel, et al. "Maximum likelihood estimates of linear dynamic systems". In: *AIAA journal* 3.8 (1965), pp. 1445–1450.

[57] Donald Reid. "An algorithm for tracking multiple targets". In: *IEEE transactions on Automatic Control* 24.6 (1979), pp. 843–854.

[58] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF." In: *ICCV*. Vol. 11. 1. Citeseer. 2011, p. 2.

[59] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. "Efficient & effective prioritized matching for large-scale image-based localization". In: *IEEE transactions on pattern analysis and machine intelligence* 39.9 (2017), pp. 1744–1756.

[60] Grant Schindler, Matthew Brown, and Richard Szeliski. “City-scale location recognition”. In: *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*. IEEE. 2007, pp. 1–7.

[61] Markus Schreiber, Carsten Knöppel, and Uwe Franke. “Laneloc: Lane marking based localization using highly accurate maps”. In: *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE. 2013, pp. 449–454.

[62] Santokh Singh. *Critical reasons for crashes investigated in the national motor vehicle crash causation survey*. Tech. rep. 2015.

[63] Erik Stenborg and Lars 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.

[64] Linus 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.

[65] Peter JG Teunissen. “The least-squares ambiguity decorrelation adjustment: a method for fast GPS integer ambiguity estimation”. In: *Journal of geodesy* 70.1 (1995), pp. 65–82.

[66] Sebastian Thrun et al. “Stanley: The robot that won the DARPA Grand Challenge”. In: *Journal of field Robotics* 23.9 (2006), pp. 661–692.

[67] Sebastian Thrun et al. “Robust Monte Carlo localization for mobile robots”. In: *Artificial intelligence* 128.1-2 (2001), pp. 99–141.

[68] Antonio 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.

[69] Trafikanalys. *Vägtrafikskador 2011*. Tech. rep.

[70] Iwan Ulrich and Illah Nourbakhsh. “Appearance-based place recognition for topological localization”. In: *Robotics and Automation, 2000. Proceedings. ICRA’00. IEEE International Conference on*. Vol. 2. Ieee. 2000, pp. 1023–1029.

[71] Chris Urmson et al. “Autonomous driving in traffic: Boss and the urban challenge”. In: *AI magazine* 30.2 (2009), p. 17.

[72] Anh Vu, Jay A Farrell, and Matthew Barth. “Centimeter-accuracy smoothed vehicle trajectory estimation”. In: *IEEE Intelligent Transportation Systems Magazine* 5.4 (2013), pp. 121–135.

[73] E.A. Wan and R. Van Der Merwe. “The unscented Kalman filter for nonlinear estimation”. In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium* (2000).

[74] Fisher Yu and Vladlen Koltun. “Multi-scale context aggregation by dilated convolutions”. In: *arXiv preprint arXiv:1511.07122* (2015).

[75] Zhengyou Zhang. “A flexible new technique for camera calibration”. In: *IEEE Transactions on pattern analysis and machine intelligence* 22.11 (2000), pp. 1330–1334.