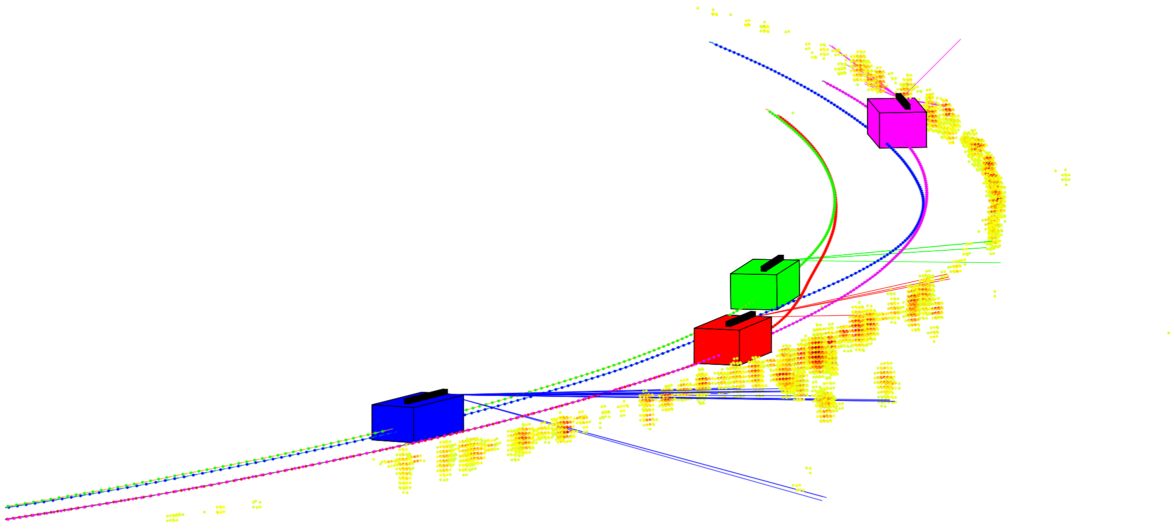
CHALMERS
UNIVERSITY OF TECHNOLOGY



3D Localization and Mapping using automotive radar

Ego-positioning using 3D Occupancy Grid Mapping

Master's thesis in Systems, Control and Mechatronics

Markus Hammarsten
Viktor Runemalm

Department of Signals and systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Master's thesis 2016:EX026

3D Localization and Mapping using automotive radar

Ego-positioning using 3D Occupancy Grid Mapping

Markus Hammarsten
Viktor Runemalm

Department of Signals and systems
Division of Signals Processing and Biomedical Engineering
Signal Processing Group
Chalmers University of Technology
Gothenburg, Sweden 2016

3D Localization and Mapping using automotive radar
Ego-positioning using 3D Occupancy Grid Mapping
Markus Hammarsten
Viktor Runemalm

Supervisor: Johan Degerman, SafeRadar Research Sweden AB
Examiner: Lennart Svensson, Department of Signals and systems

Master's Thesis 2016:EX026
Department of Signals and Systems
Division of Signals Processing and Biomedical Engineering
Signal Processing Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Visualization of an Occupancy Grid Map created from real data based on multiple test drives. The trajectories of the test drives are shown together with all grid point with probability of occupancy higher than 0.8.

Typeset in LaTeX
Gothenburg, Sweden 2016

3D Localization and Mapping using automotive radar
Ego-positioning using 3D Occupancy Grid Mapping
Markus Hammarsten, Viktor Runemalm
Department of Signals and systems
Chalmers University of Technology

Abstract

This thesis considers the problem of localization and mapping for a vehicle using front-looking radar. Using a reference vehicle equipped with a DGPS-system, measurements of the static environment are accumulated into an Occupancy Grid Map with a resolution of 0.2 meters. Since the radar has the ability to produce unambiguous measurements in both azimuth and elevation, we create the map in 3D. The method recursively updates the occupancy map of the static environment using an inverse sensor model. A vehicle driving through the same area can then be located by utilizing the information stored in the map. By fusing the inertial measurement from the vehicle with the radar measurements, together with the offline 3D map, the vehicle states are estimated in all six degrees of freedom (position and orientation). We employ two different methods for localization, one using a modified Rao-Blackwellized particle filter, and one based on registration utilizing tricubic interpolation and gradient search. Both proposed solutions are able to accurately estimate the ego-position of the car, within 0.3 meters of ground truth in most estimates, in both simulation and on real data. Cumulative errors, caused by only using inertial measurements, are successfully removed by both implementations. Radar's ability to distinguish dynamic from static targets, together with weather robustness, enables 3D radar to play a vital part for automotive positioning systems.

Keywords: Occupancy grid map, inverse sensor model, Binary Bayes' filter, sensor fusion, radar, 3D, EKF, tricubic interpolation, Rao-Blackwellized particle filter, vehicle localization.

Acknowledgements

Without questions, this could not have been possible without the tremendous assistance and dedication from the active safety team of DENSO Sales Sweden AB. We have received input, feedback, data and, perhaps most importantly, a warm welcome and friendly treatment from what during our time almost seemed like co-workers.
A special thanks goes to Johan Degerman, our supervisor at DENSO Sales Sweden AB, who has helped us tremendously, be it to provide a simulation platform or discuss the approaches with us.
Further, we want to extend our gratitude towards our academic supervisor and examiner Lennart Svensson. Despite a heavily packed schedule he found time to assist us and guide us when we felt stuck.
Last, but not least, a big thank you to our respective families whom have supported us through this journey.

Markus Hammarsten and Viktor Runemalm, Gothenburg, June 2016

Contents

List of Figures

List of Tables

1 Introduction

Autonomous drive and safety functions are hot topics in the automotive industry today. Various research, e.g. [25, 26], indicate that most accidents in traffic are related to human factors, which is why self-driving vehicles are gaining increasing interest. Environmental aspects such as decrease in fuel consumption also push the automation level of vehicles forward. In the area of autonomous drive, it is important that the vehicle has good knowledge about the proximity and its position in the world in order to make good predictions of other vehicles and accident risks. Thus, the need for robust sensors is inevitably also becoming more important. The internal sensors in a vehicle cannot alone obtain the global ego-position of the car, i.e. the position of the car in a global reference frame. They only measure e.g. angular rates, steering angle and velocities without any reference to position or rotation, and dead reckoning will result in bias errors very quickly. The global positioning system (GPS) is unreliable in urban environments due to occlusion and multipath. This calls for exteroceptive sensors, which are sensors that create measurements from external sources. A commonly used sensor for this purpose is lidar, which is a laser sensor. However, a robust system cannot rely only on optical systems such as camera or lidar, due to factors as rain and snow. Radar sensors are, compared to optical sensors, very robust to bad weather [11], dust and particles, vibrations and rough conditions in general. They also possess the ability to measure radial velocity instantly, which facilitates separation of moving and stationary environment. In addition, they are quite cheap as compared to scanning lasers. However, they usually carry disadvantages like interference problems and relatively low resolution in angles. This report focus on low-cost automotive radar sensors, and their ability to accurately map the environment in 3D and to robustly localize the ego-vehicle.

Mapping, in robotics, is a technique to store measurements of the environment to be used in a reference frame. The task of evaluating a position and orientation in that reference frame is called localization. Both localization and mapping are well studied fields of research, particularly the case were they are solved simultaneously, aka simultaneously localization and mapping (SLAM), and there exists vast number of publications. Most of them employ lidar sensors. SLAM using radar for mapping and ego-vehicle localization was performed back in 1999 [9], where they use landmarks to represent the environment. The landmark approach require association between measurements and landmarks to be correct all the time, or else the vehicle pose will get a bias error. To handle this problem there exist multihypothesis approaches such as FastSLAM [22]. FastSLAM uses particle filters to solve the estimation, and the problem of vehicle localization by particle filtering is widely researched, where [1, 8, 29] solve 2D state estimations by sensor fusion of varying sensor combinations. The association problem is particularly cumbersome for the radar, and thus it is common to circumvent the problem by representing the environment as a grid [24]. The occupancy grid map (OGM) is a frequently used map for localization purposes, and it is essentially a grid with spatial information of the environment.

The art of aligning two data sets is called registration. This can be utilized for localization, and is then often referred to as scan matching. When a map is utilized in scan matching, the objective is to find a pose that aligns measurements with the map. When the sensor is mechanically scanning a narrow lobe in 2D, scan matching works [32] but require a high rotation frequency (or low vehicle speed). A typical automotive radar is not mechanically scanning but instead digitally scanning, and uses direction-of-arrival techniques for angular estimation. Hence, we are referred to form a measurement scan using detected plots rather than signal down-range, and update the grid map with the OGM inverse sensor model. For localization purposes, radar measurements are being fused with the data from the inertial measurement unit (IMU) in order to create good estimations of the positioning. With radar measurements, a map can be built in which a vehicle can move around and use algorithms for ego-localization. A similar problem to this thesis has been previously tackled in [23], where they utilize 2D radar with the inverse sensor model and scan matching to solve the SLAM problem.

1.1 Purpose

As aforementioned, optical sensors are prone to fail under harsh conditions, but localization systems have to perform well regardless of environmental conditions. Thus, it is important to obtain high quality, yet inexpensive, robust systems as either replacements or supplements. The purpose of this thesis is to investigate how to approach the positioning and mapping problem with one front-looking 3D radar, determine how well it can perform and discuss radars place among exteroceptive sensors. We will investigate radars place amongst sensors used for localization, and determine if the purpose of radar can be extended beyond its regular use in today's industry. Today, there exist no known articles or papers which covers the combination of localization and mapping with 3D radar. This will be the focus of this thesis, and the feasibility of a solution to the problem will be investigated.

1.2 Objective

The objective of this thesis is to 1) create a 3D occupancy grid map of the static environment, by measurements from a front-looking radar mounted on a car, and 2) implement two localization algorithms and estimate the ego-position by sensor fusion of the IMU and the radar. The two implementations will be compared in order to evaluate which performs the best, given the prerequisites of this thesis. The goal is to enhance the ego-localization performance by dead reckoning, by including information from the map.

The algorithms will be implemented in MATLAB for simulation and validation. Important properties of the method is robustness, accuracy, and lastly computational efficiency. The thesis covers both simulated data as well as real data, but everything is processed offline and a real-time implementation will not be considered.

1.3 Contributions

In this thesis, the authors mainly contribute with three new ideas and modifications to existing methods. The first is the heuristic inverse sensor model, where we form a new model inspired by two previous implementations. The model is directly correlated to occupancy grid map creation, i.e. we have created a new model to recursively build maps. The second idea is the utilization of tricubic interpolation to smooth a discrete map, enabling optimization methods to operate on continuous functions and derivatives for likelihood evaluation used for ego-localization. The last idea are two simplifications to the Rao-Blackwellized particle filter, which decrease the computational speed but retain the accuracy. By assuming that there is no noise on the states solved by the particle filter, the true state of the propagated particles can be utilized in the second Kalman filter update. With our approach, a single Kalman filter can be used for the entire particle filter, instead of one Kalman filter for each particle.

1.4 Thesis outline

This section describes the layout of the report. The report will first cover a problem description alongside an explanation regarding the system setup, sensor specifications, the simulation environment and the test case on real data. Secondly, the report moves on with background theory and methods that are required by our approach. The theory covers basics of, Frequency Modulated Continuous Wave (FMCW) radar, signal processing of radar measurements, Bayesian theory with Kalman filters, Extended Kalman Filters (EKF), multiple models, and particle filters, mapping (occupancy grid), inverse sensor model and registration. An experienced reader within the theory subjects can skip the theory and move on to implementation directly, which are chapters 4, 5 and 6. The implementation chapters cover vehicle modelling, sensor modelling, map generation and filter designs. The results are presented in Chapter 7, together with evaluation and minor discussions. Lastly, a discussion and a conclusion regarding the thesis in its entirety finish the thesis.

2
Problem description and setup

This chapter will narrow down the specifics of the thesis, detailing the platform, sensors and data which we have worked with. The simulation environment details are explained, and a description of the real scenario is provided. A scope of the project is also presented, which encapsulate the thesis.

2.1 Scope

The scope of this thesis project is to investigate the capability of localization and mapping using a 77 GHz radar. We will not consider the *simultaneous* localization and mapping, aka SLAM, but instead keep the two problems separate. This means that our solution, in comparison to a more general case, will be unable to handle localization in unexplored areas. Further, we do not aim to develop the best ego-localization algorithm possible, instead we focus on a localization and mapping problem only utilizing a front-looking radar. We will also base the algorithms on good prior knowledge, i.e. the initial position of the car is in a close vicinity of the true position.

The vehicle will only move in, what we refer to as, a rich environment. To specify, we argue that it is required that the radar can obtain several detections each cycle, that objects in the environment gives consistent scatters, and that the detections are well spread out in space. The solution should be able to handle detections of dynamical objects, e.g. a passing car, while still being able to estimate the ego-position from the static environment

We will not deeply focus on IMU sensor modelling, but instead focus on the radar sensor model. The available IMU sensor data is already preprocessed to some degree, and the modelling will be done for the signals that are retrieved. There exist no data on measurement noise for the IMU, but we limit ourselves to not examine the sensor noise thoroughly.

2.2 System setup

For this thesis, DENSO Sales Sweden AB, DSSE, provides a Volvo S60 as a test vehicle. It is modified with some extra sensors, which are further described below. The exteroceptive sensor in this thesis is a front looking 77 GHz short range radar, developed by DSSE together with Qamcom Research & Technology. It is a first prototype, which has the unique feature of instantaneous 3D measurements. It has a range of approximately 30 meters, and a field of view of ± 70 degrees in both azimuth and elevation. It can, however, only make unambiguous detections up to ± 50 degrees, which introduces false, aliased, detections. This problem is due to the transmitter antenna. Based on Cramér-Rao lower bound, the lowest possible standard deviations of unbiased estimated parameters from data [7], the radar has a resolution of 0.5 degrees in azimuth, and 2 degrees in elevation. The radar has a measurement accuracy in range of approximately 0.3 meters.

The car has a built in IMU and sends signals on CAN, which are accessible. Important signals will be extracted and utilized in the data fusion. Examples of these signals are wheel speeds and yaw rate. The uncertainty of these signals will have to be examined during implementation. In order to evaluate the data, an Oxford RT 2000 is used as ground truth. It consists of three high accuracy accelerometers and gyroscopes, as well as a differential global positioning system (DGPS). It can compute all the desired states with high accuracy, position with 0.02 meter error, which makes it a good choice as a reference system. The sensor positions on the car can be seen in Figure 2.1.

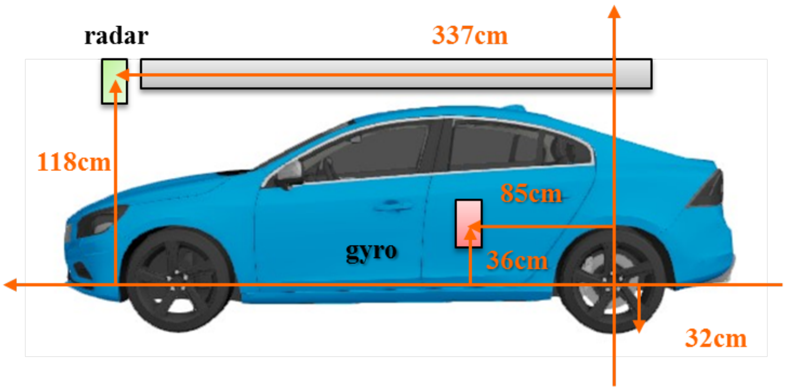
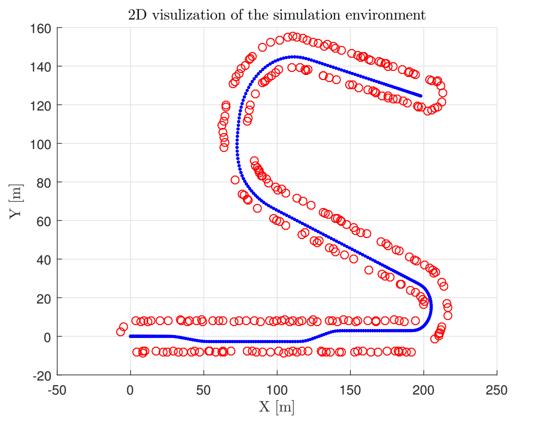


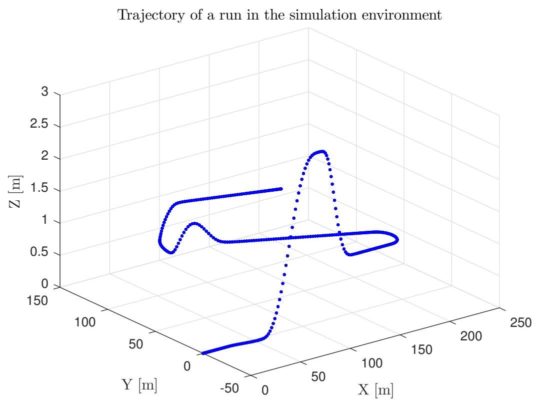
Figure 2.1: Visualization of the Volvo S60 and the sensors, where the orange box represents the Oxford RT 2000. The offsets are calculated based on the rear axis.

2.3 Simulation environment

A simulation environment in MATLAB is provided to support verification and evaluation of the implemented algorithms. Trajectories and scatterers (radar targets) can be created on demand, and it is a good platform to evaluate an implemented approach before moving on to real data. It can also be used to find good software configurations, i.e. balanced parameter settings. The scatterers correspond to real radar targets and have a probability of being detected while in the simulated field of view of the radar. In a controlled scene, it is easier to evaluate the impact of the algorithms since real data might be affected by external factors which cannot be compensated for, e.g. aliased detections. Figure 2.2a and 2.2b display the simulation environment, where the blue dots are the trajectory for one of many possible drives, and the amber markings around the trajectory are the scatterers. The trajectory consist of curves, lane changes, hills and even a velodrome curve in order to fully test the estimation of all considered dimensions.



(a) The simulation environment in MATLAB. The blue dots are the vehicle trajectory and the red circles are scatterers located at the side of the road.



(b) The simulation environment containing only the trajectory to capture the 3D aspect. To highlight the height of the map, the axes have different scaling.

Figure 2.2: Two images of the simulation environment in MATLAB to illustrate where the tests are carried out before moving on to real data.

2.4 Test scenario

The test team at DSSE has collected data at NEVS test track and provides a good test case, with several runs of the scenario with some slight variations. The vehicle travels beside a guard rail on its right, there are also four houses and some trees next to the road. An overview of the test area can be seen in Figure 2.4 and a picture from one of the scenarios can be seen in Figure 2.3, where the guard rail, some houses and another vehicle is present. In some of the test scenarios, the car either overtakes or is overtaken by another car. The other scenarios consider only the car and the static environment. The environment is not ideal, there are mainly detections on one side of the road. However, the detections are distinct and the guard rail produces consistent scatterers at the posts which could enable reliable ego-positioning. There are four usable drives where the measurements are synchronized and checked by the test team. It would have been preferable to have more data, i.e. more test runs, for the sake of evaluating the algorithms. In order to evaluate the generality of the proposed solutions, other test cases, in varying environments, are required.

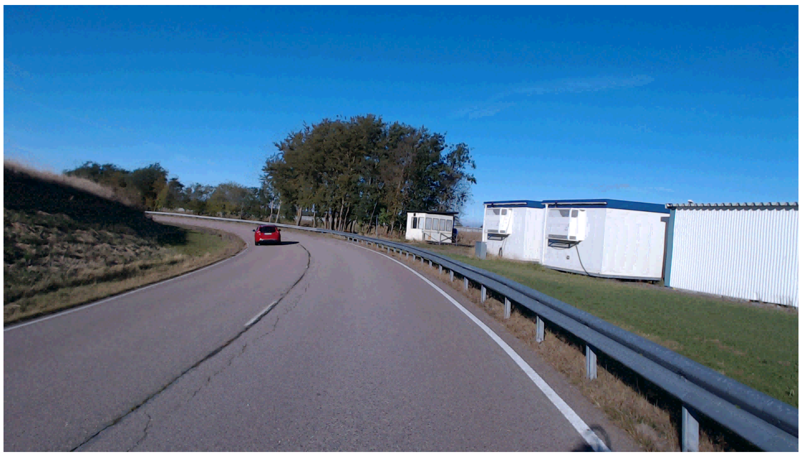


Figure 2.3: A snapshot from the guard rail case, which shows the guard rail, a few backsides of houses and another vehicle which is overtaken in the scenario. The camera is mounted inside of the vehicle looking out of the front window.

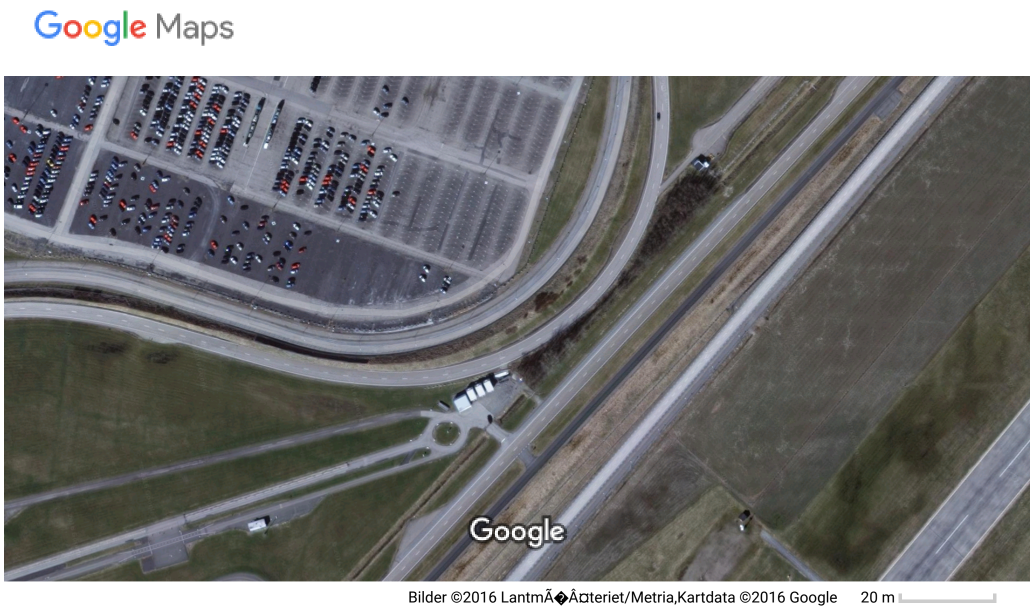


Figure 2.4: A Google Map view over the test area. All the test cases drive through the curve in the middle, beside the small white houses.

3
Background theory

To give the reader an understanding of the contents of the report, the underlying theory for the approaches, methods and algorithms are presented in this chapter. We cover filtering techniques, mapping (and more specifically occupancy grid mapping), registration, and motion modeling. This chapter also provides information regarding radar, as it is a central part of the thesis, and will, also, provide facts used in the discussion of radar's place among sensors in the automotive industry.

3.1 Radar

Radio detection and ranging, abbreviated radar, is a sensor which produces measurements from electromagnetic waves. Historically, radar has been widely used in military applications since its break through during World War II [44]. Since then, radar has entered civilian and commercial applications, such as weather forecasting and air traffic control. Today, one rapidly growing market for radar is the automotive industry, where radars are providing features as automatic cruise control (ACC) and autonomous emergency brake (AEB). There are several types of radar, but this report will only cover frequency modulated continuous wave (FMCW) radar which transmits continuous waves with frequency ramps.

Radar transmits a wave from the transmitter antenna, which bounces on objects and is reflected back to the receiver [11]. The receiver antenna uses binning for range detection, which are based on samples of the received waves, and more bins (higher sampling rate) implies a higher resolution of the range measurements. To achieve unambiguous range detections, the idea is to detect a short signal in the receiver to map it to few bins. However, the advantage of a longer signal duration is that it contains a higher signal energy. To solve the trade-off, one approach is pulse compression. Here, a long duration signal is transmitted, which allows for high signal energy, but the signal is a combination of many partially continuously increasing and/or decreasing frequencies. A full signal consists, thus, of partial signals which are known as chirps. In digital radars, a matched filter is used to map each transmitted signal to the received one in order to compress the response.

As radar, opposed to lidar, uses waves instead of rays, it can perform an instantaneous measurement of multiple objects in different directions. However, to extract target data, radar data have to be processed. The desired target data are the polar domain data, range and angles, but also the relative velocity and SNR. Radar has the ability to measure the Doppler effect, which can be used to compute the relative velocity of the target and the radar. Radar receives a mix of signals and a detector has to compute which parts of the input are from targets and which originate from noise. The next subsections will cover the range and Doppler computations, target/clutter estimation and direction of arrival, in that order. Figure 3.1 present the order of processing, where FFT^2 represents the range and Doppler computations, Section 3.1.1, CA-CFAR is the detector, Section 3.1.2, and MUSIC computes direction of arrival in Section 3.1.3.

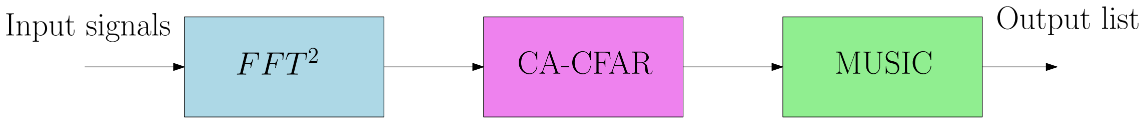


Figure 3.1: The flowchart describes the radar pipeline, from received input to finished processing. The output list is a list of detections with the computed attributes, e.g. range and SNR.

3.1.1 Signal processing to compute range and Doppler

To compute the range to a target, radars utilize that radio signal propagates at the speed of light. Given a static media, the round-trip time, t_{rt}, for a signal is

$$t_{rt} = \frac{2r}{c}, \tag{3.1}$$

where c is the speed of light and r is the distance to the target. The time lag can be computed by the matched filter which compares the transmitted and received signal. Now, equation (3.1) can be utilized to calculate the distance, r, to the reflector [44]. By further investigation of equation (3.1), the relation between maximum range and the chirp raise time is identified [27]. In order to unambiguously determine the ranges, all echoes have to arrive before the next chirp is transmitted. Otherwise, aliasing can occur, and faulty detections can be achieved. The rate of chirp transmissions is called pulse repetition frequency, PRF, and there are two key aspect regarding its magnitude. A high PRF means higher signal energy and greater range, but the negative aspect is the aforementioned ambiguity.

A key aspect of the radar is the ability to measure the relative radial velocity between the radar and an object. This is done by utilizing the Doppler effect, i.e. computing the frequency shift which originates from objects moving towards or away from the radar [44]. The Doppler shift, f_D, is calculated by

$$f_D = \frac{2 \cdot v_{relative}}{\lambda}, \tag{3.2}$$

where $v_{relative}$ is the relative velocity between the radar and the target, and λ is the wave length of the transmitted signal. To compute the range and relative velocity, the round-trip time and Doppler frequency have to be determined. In order to illustrate it, Figure 3.2 visualizes the identification by the distances in the transmitted signal, red, and the received signal, green, in both frequency and time.

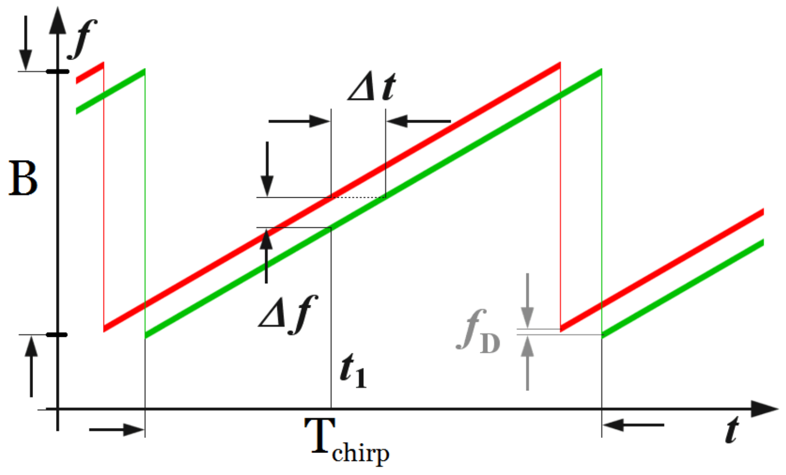


Figure 3.2: Illustration of how the time between a transmitted and received chirp, Δt, and the frequency difference between a transmitted and received chirp, Δf, are determined by a transmitted, red, and received, green, signal. The height of the triangle is the bandwidth, B, the distance between the two vertical lines is T_{chirp} and f_D is the Doppler frequency.

The variable Δf is known as beat frequency, and it is the difference in frequency between the transmitted and received signal. When computing the range, the Doppler frequency, f_D, is negligible when the transmission rate is large. In automotive industry, a common radar frequency is 77 GHz, which is significantly larger than the Doppler frequency. However, when computing the relative velocity, the signal is mixed down to a lower frequency, where the Doppler effect can be computed. From Figure 3.2, a relation between the frequencies and time can be constructed. The height of the triangle is the bandwidth, B, and the time in between two different saw tooth waves is T_{chirp}. Hence, the relationship between the aforementioned variables are formed in accordance to [44] as

$$\Delta t = \frac{\Delta f \cdot T_{chirp}}{B}. \tag{3.3}$$

By combining equation (3.1) and (3.3), the equation to calculate the range, r, becomes

$$r = \frac{c \cdot \Delta f \cdot T_{chirp}}{2B}. \tag{3.4}$$

Range binning is usually performed with fast Fourier transform (FFT) on the digitalized signal. After the A/D conversion, one FFT in fast time and one FFT in slow time, i.e. over one chirp and over all chirps forming a signal, are computed in order to calculate the range and relative velocity. When the range and velocity calculations are done, a so called range-Doppler map can be created. An example of how a range-Doppler map can look can be seen in Figure 3.3. The x-axis is relative velocity, the y-axis is range and the color scale describe the amplitude of the FFT spectra in decibel (dB). During the specified time instance, the radar is driving forward and detecting clutter, hence the yellow region at negative Doppler.

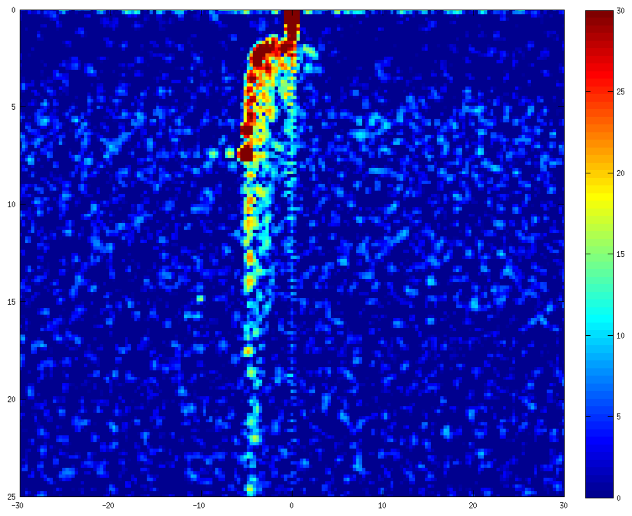


Figure 3.3: A range-Doppler map from a sequence of radar inputs where the radar is moving forward at approximately 5 m/s and detects clutter, which moves toward the car at - 5 m/s. The x-axis is relative velocity, the y-axis is range and the color scale describe the amplitude of the FFT spectra in decibel (dB).

3.1.2 Target detector

In order to determine which parts of the signal is noise, and which parts are desired target detections, a detector is required. Reflections, or echoes, bounce of targets which weakens the energy in the signal. The energy of the signal also lose power with respect to the distance raised to four, from transmitter to receiver [44, p.31]. Thus, a smart detector has to be applied in order to avoid inclusion of noise as detections. The considered detector in this thesis is the cell-averaging constant false-alarm rate (CA-CFAR).

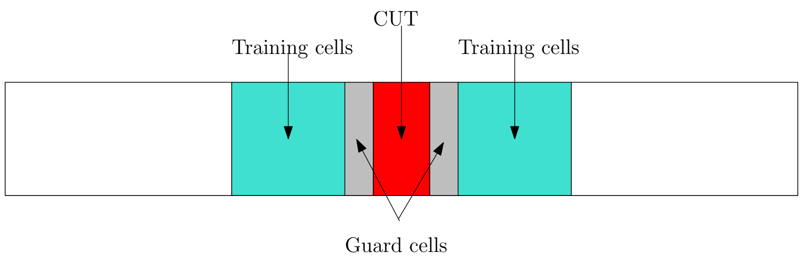


Figure 3.4: Image displaying the interpretation of one-dimensional CA-CFAR. The red segment is the cell under test (CUT) the grey segments are guard cells, the cyan segments are the training cells, and the white segments are the unused surrounding cells.

CA-CFAR employs an adaptive threshold to compute detections given a user-set probability of false alarm rate, P_{fa} [6, p.1]. CA-CFAR operate in the range-Doppler map, see Figure 3.3, and from each cell, defined by a combination of range and relative velocity in the map, the CA-CFAR algorithm will determine whether or not the cell is a target detection or not. A visualization of a one-dimensional CA-CFAR can be seen in Figure 3.4. The algorithm computes the interference from a window around the cell under test (CUT) by estimating the mean of the power in the window. To avoid any power-spill from the CUT, guard-cells are used in the direct neighbourhood. When the interference is computed, it is scaled by a constant which couples the threshold with the P_{fa}.

$$T = \alpha \frac{1}{N} \sum_{m=1}^{N} x_m \tag{3.5}$$

computes the threshold, T, where x_m is a sample from the training cells, N is the number of training cells, and α is computed by

$$\alpha = N(P_{fa}^{-1/N} - 1)[21]. \tag{3.6}$$

As the thresholds are computed for local cells, it changes depending on the CUT, but the constant P_{fa} is what connects the output when using the thresholds to determine if a detection originate from a target or from noise. CFAR can operate on multiple channels, i.e. multiple receiver antenna patches, but these computations consider only one channel.

SNR can tell us more than just what the ratio between signal and noise is, the value itself contain information how likely a detection is. Naturally, a higher SNR means that the detection is more probable. Based on the Swerling model 1, a connection between P_{fa}, SNR and probability of detection, P_d, is derived in [33, p.1173] as

$$P_d = P_{fa}^{\frac{1}{(1+SNR)}}. \tag{3.7}$$

This can be utilized when evaluating how good a detection is.

3.1.3 Direction of arrival

When the CA-CFAR detector has computed detections in range-Doppler, the remaining part is to estimate the direction of arrival (DOA). In 2D, this often correspond to azimuth and in 3D both azimuth and elevation. There exist different techniques for DOA estimation, but this report is limited to one approach called multiple signal characterization. This is because it is already implemented in the signal processing provided by DENSO Sales Sweden AB. Multiple signal characterization, abbreviated MUSIC, in radar applications is a technique used to estimate, among others, DOA of scatterers [34]. The considered general problem involves an antenna with M patches, located arbitrarily on the antenna plane, which receive waves from unknown directions. The covariance between the signals is also considered to be unknown and there is a background noise present. However, the amount of signal sources has to be known which is a major drawback for some applications.

MUSIC is an eigenspace method, and the idea is to distinguish the signal domain from the noise domain [43]. The approach utilizes that the eigenvectors of the autocorrelation matrix, $\mathbf{S}$, can distinguish the two domains from each other. Before computing $\mathbf{S}$, a model is required. The incident waveforms are modelled as a complex vector on the form

$$\mathbf{X} = \mathbf{AF} + \mathbf{W}, \tag{3.8}$$

which size depend on the number of array elements, M, on the antenna, where $\mathbf{A}$ is a matrix spanned by linearly combined functions, $\mathbf{a}_{ij}$, containing frequency information, $\mathbf{F}$ is a vector of amplitudes corresponding to the frequency components, and $\mathbf{W}$ is the noise which can originate from both the incoming signals and/or from the sensor itself [34]. A function $\mathbf{a}_{ij}$ is a function of the arrival angle at array i of signal j, and are known. The covariance matrix of equation (3.8) can now be computed as

$$\mathbf{S} = \overline{\mathbf{XX}^*} = \mathbf{A}\overline{\mathbf{FF}^*}\mathbf{A} + \overline{\mathbf{WW}^*}. \tag{3.9}$$

After computation and ordering the eigenvalues in $\mathbf{S}$, the largest eigenvalues represent the signal domain and the lowest represent the noise domain. Each eigenvalue represent an eigenvector, which together with the assumption that the noise and signals are uncorrelated makes the eigenvectors perpendicular to one another. This is the basis of the MUSIC estimator, which is expressed as

$$\mathbf{P}_{MU}(\theta) = \frac{1}{\mathbf{a}^*(\theta)\mathbf{E}_N\mathbf{E}_N^*\mathbf{a}(\theta)}, \tag{3.10}$$

where $\mathbf{E}_N$ is a matrix of the eigenvectors of the noise and $\mathbf{P}_{MU}(\theta)$ is the estimation function [43]. Due to no correlation between the signals and the noise, equation (3.10) will achieve peaks (infinitely large during ideal conditions where everything is known, including the noise) whenever the parameter θ belongs to the signal domain. In practise the result is instead steep peaks, for each signal. The complete derivation of equation (3.10) is presented in [43, p.11].

3.2 Bayesian filtering

A well-established framework, which models and estimates uncertainties, is Bayesian filtering [20, p.17-20]. As sensors are imperfect, they cannot measure deterministically, their measurements can be represented by stochastic variables. In fact, Bayesian filtering considers all unknown quantities as stochastic. Bayesian filtering allows efficient computation of said variables and is therefor a favourable approach to a sensor based problem. In a recursive manner, a state vector, $\mathbf{x}_k$, including desired parameters, is estimated based on sensor measurements. The desired output from each recursion is a posterior distribution of the state vector. One, or several, sensors produce measurements at times k, described by

$$\mathbf{z}_{1:K} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_K\} \ , k \in 1, \dots, K, \tag{3.11}$$

where $\mathbf{z}_k$ contains all measurements from the sensors at time k. These are used in order to compute a probability density function, PDF, of the posterior

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}). \tag{3.12}$$

An important modelling feature is the utilization of the first order Markov property. A state $\mathbf{x}_k$ is assumed to be only dependant on the state $\mathbf{x}_{k-1}$, and a measurement $\mathbf{z}_k$ is assumed to be only dependant of the measured state $\mathbf{x}_k$. This provides equations

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}, ..., \mathbf{x}_0) = p(\mathbf{x}_k|\mathbf{x}_{k-1}) \tag{3.13}$$

and

$$p(\mathbf{z}_k|\mathbf{x}_k, ..., \mathbf{x}_0) = p(\mathbf{z}_k|\mathbf{x}_k). \tag{3.14}$$

Bayesian filtering is constructed around Bayes' rule, and the following steps shows how the posterior can be formed by the combination of equations (3.13), (3.14) and Bayes' rule. The measurement dependencies in the posterior are first split up as

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) = p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{z}_{1:k-1}). \tag{3.15}$$

Bayes' rule is applied on (3.15) to describe the posterior as a combination of conditional probabilities,

$$p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{z}_{1:k-1}) = \frac{p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{z}_{1:k-1})p(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})} = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})}. \tag{3.16}$$

The second equality stems from the aforementioned assumption regarding the Markov chain property. The first term in the numerator of the fraction is known as the likelihood, while the second term in the numerator is a predicted density, also known as the prior. The denominator term is a normalization factor, which scales the integration of the posterior distribution to one.

By skipping the normalization factor, the posterior can be described as Posterior $\propto$ Prior $\cdot$ Likelihood. In order to find the prior distribution, the Chapman-Kolmogorov equation is used by marginalizing over the previous time step. The Chapman-Kolmogorov equation is

$$\begin{aligned} p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) &= \int p(\mathbf{x}_k, \mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}) d\mathbf{x}_{k-1} \\ &= \int p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_{1:k-1}) p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}) d\mathbf{x}_{k-1}. \\ &= \int p(\mathbf{x}_k|\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}) d\mathbf{x}_{k-1} \end{aligned} \tag{3.17}$$

The terms in the last equality are identified as the process model, equation (3.13), and the posterior density at the previous time step, $k-1$, respectively. The process model, or in the tracking case known as the motion model, describes how the states evolve over time. It is, thus, shown that given a prior, the posterior distribution can recursively be computed by the presented approach.

3.2.1 Kalman filtering

The renowned Kalman filter, presented by R.E. Kalman and R.S. Bucy in [31], presents the optimal filter for stochastic linear processes with additive white Gaussian noise. In [15, p.39] they explain how given a Gaussian prior PDF, the posterior PDF will also be Gaussian. This is known as conjugate distribution, and it is a reason why the Kalman filter is optimal when recursively computing the posterior distribution for a system with Gaussian noise.

A discrete motion model, f_{k-1}, describes how a dynamical system transitions the states between time instances, while the likelihood describes how measurements are predicted by a given state vector [20, p.20-21]. Continuing with the discrete case, the mathematical expressions for motion and measurement models are

$$\begin{aligned} \mathbf{x}_k &= f_{k-1}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}) \\ \mathbf{z}_k &= h_k(\mathbf{x}_k, \mathbf{r}_k) \end{aligned} \tag{3.18}$$

The functions f_{k-1} and h_k belong to the nonlinear domain, but as linear functions are a subset of nonlinear functions they can possibly be linear. In the case they are linear, the functions can be described by

$$\begin{aligned} \mathbf{x}_k &= \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{q}_{k-1} \\ \mathbf{z}_k &= \mathbf{H}_k\mathbf{x}_k + \mathbf{r}_k, \end{aligned} \tag{3.19}$$

where the process and measurement noises are $\mathbf{q}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1})$ and $\mathbf{r}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$, respectively.

As a Gaussian distribution is modelled by the parameters mean value and covariance, the Kalman filter computes the optimal estimation for them. First and foremost, an estimated mean, $\hat{\mathbf{x}}_{k|k-1}$, and covariance, $\mathbf{P}_{k|k-1}$, of the motion model in equation

(3.19) are computed, and the result is

$$\begin{aligned}\hat{\mathbf{x}}_{k|k-1} &= \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1|k-1}\\ \mathbf{P}_{k|k-1} &= \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}.\end{aligned} \tag{3.20}$$

The subscripts $\{k-1|k-1\}$ and $\{k|k-1\}$ are used to denote the posterior of time instance $k-1$ and the prediction at time step k, conditioned on the previous time step, respectively. The variable $\hat{\mathbf{x}}$ is used to distinguish estimated values from true values $\mathbf{x}$. The result from equation (3.20) can now be used in the update step,

$$\begin{aligned}\mathbf{S}_k &= \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k\\ \mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{H}_k^T\mathbf{S}_k^{-1}\\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1})\\ \mathbf{P}_{k|k} &= \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{P}_{k|k-1}^T,\end{aligned} \tag{3.21}$$

where a posterior density is computed by mean value and covariance. For the derivation and explanation of the equations, see [31]. The resulting posterior distribution is presented by equation (3.22). This is, as shown in Section 3.2, used as a prior in the next recursion which progresses through all time instances k.

$$p(\hat{\mathbf{x}}_{k|k}|\mathbf{z}_{1:k}) = \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}) \tag{3.22}$$

3.2.2 Extended Kalman filter

The Kalman filter is only optimal for linear models, since linear filters retain the conjugacy of distributions for Gaussians. Nonlinear models can break this relation, and alternative solutions have to be found. One approach is to linearize the models before performing Kalman filtering, this is known as Extended Kalman filtering (EKF) [20, p.22]. The motion model is linearized via a first order Taylor expansion around the estimated state, $\hat{\mathbf{x}}_{k-1|k-1}$, at each time instance. As linearizations introduce errors, the EKF is only approximately optimal. For highly nonlinear models the EKF performs badly, but there have been suggested improvements to the algorithm, see [13], [2].

Consider the motion model presented in equation (3.18) where $\mathbf{q}_{k-1}$ is additive white Gaussian noise. The resulting Taylor expansion is then

$$\begin{aligned}\mathbf{x}_k &= f(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1}\\ &\approx f(\hat{\mathbf{x}}_{k-1|k-1}) + \tilde{\mathbf{F}}(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1|k-1}) + \mathbf{q}_{k-1},\end{aligned} \tag{3.23}$$

where $\tilde{\mathbf{F}}$ is defined as the Jacobian matrix evaluated at $\hat{\mathbf{x}}_{k-1|k-1}$. As equation (3.23) now is linear, a Kalman filter can be used to solve the problem. The mean value and covariance are computed by

$$\begin{aligned}\hat{\mathbf{x}}_{k|k-1} &= f(\hat{\mathbf{x}}_{k-1|k-1})\\ \mathbf{P}_{k|k-1} &= \tilde{\mathbf{F}}\mathbf{P}_{k-1|k-1}\tilde{\mathbf{F}}^T + \mathbf{Q}_{k-1},\end{aligned} \tag{3.24}$$

which are identified as the prediction step in the Kalman filter.

For a nonlinear measurement model, another Taylor expansion can be made. However, this time the point of linearization is the predicted mean instead of the prior. The Taylor approximation of $\mathbf{z}_k$ as in equation (3.18), considered with additive Gaussian noise, is computed by

$$\begin{aligned} \mathbf{z}_k &= h(\mathbf{x}_k) + \mathbf{r}_k \\ &\approx h(\hat{\mathbf{x}}_{k|k-1}) + \tilde{\mathbf{H}}(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1}) + \mathbf{r}_k, \end{aligned} \tag{3.25}$$

where $\tilde{\mathbf{H}}$ is the Jacobian matrix evaluated at $\hat{\mathbf{x}}_{k|k-1}$. This produces a slightly modified version of the update step in equation (3.21), where $\mathbf{H}$ is replaced with $\tilde{\mathbf{H}}$, and the innovation update is replaced with $h(\hat{\mathbf{x}}_{k|k-1})$.

3.2.3 Multiple models

When estimating the states of a process which behaves very differently depending on the situation, one model might be insufficient to describe its behaviour. If several models are introduced, there has to exist an approach on how to weight them. We begin by introducing the Gaussian mixture distribution at time step k as

$$p(\mathbf{x}_k|\mathbf{z}_k) = \sum_{j=1}^{M} p(\mathbf{x}_k|\zeta_k^j, \mathbf{z}_k) P(\zeta_k^j|\mathbf{z}_k), \tag{3.26}$$

where M is the amount of models, ζ_k^j is mode j at time instance k, $p(\mathbf{x}_k|\zeta_k^j, \mathbf{z}_k)$ is a Gaussian distribution associated with mode j and $P(\zeta_k^j|\mathbf{z}_k)$ is the probability of said mode condition on the measurements. The exact same steps used in section 3.2, to divide the posterior distribution by Bayes' rule and with marginalization of the Chapman-Kolmogorov integral, is used here to form a relation between the posterior and a prediction step and an update step. The posterior mode probability is

$$P(\zeta_k^j|\mathbf{z}_k) \propto p(\mathbf{z}_k|\zeta_k^j, \mathbf{z}_{k-1}) P(\zeta_k^j|\mathbf{z}_{k-1}), \tag{3.27}$$

where $p(\mathbf{z}_k|\zeta_k^j, \mathbf{z}_{k-1})$ is the likelihood for a certain mode j. For the full derivations, see [16, p.28-29]. As a Gaussian distribution is described by the two moments, mean value and covariance, they have to be computed.

By defining $w_{k|k}^j$ as the associated mode weight, the mean value and covariance are calculated as

$$\begin{aligned} \hat{\mathbf{x}}_{k|k} &= \sum_{j=1}^{M} w_{k|k}^j \hat{\mathbf{x}}_{k|k}^j \\ \mathbf{P}_{k|k} &= \sum_{j=1}^{M} w_{k|k}^j [\mathbf{P}_{k|k}^j + (\hat{\mathbf{x}}_{k|k} - \hat{\mathbf{x}}_{k|k}^j)(\hat{\mathbf{x}}_{k|k} - \hat{\mathbf{x}}_{k|k}^j)^T], \end{aligned} \tag{3.28}$$

where the weights are normalized. For a recursive method, the weights have to be updated accordingly. When the models are run in complete parallel, predictive steps have to be computed for each mode, but if we instead utilize the same prediction, the weight update is simply

$$w^j_{k|k} = \frac{p(\mathbf{z}_k|\zeta^j_k, \mathbf{z}_{k-1})w^j_{k|k-1}}{\sum_{i=1}^M p(\mathbf{z}_k|\zeta^i_k, \mathbf{z}_{k-1})w^i_{k|k-1}}[16]. \tag{3.29}$$

Figure 3.5 displays how a flow chart model with the same prediction works, the example uses two measurement updates which are unified to the posterior before moving on to the next iteration $k + 1$.

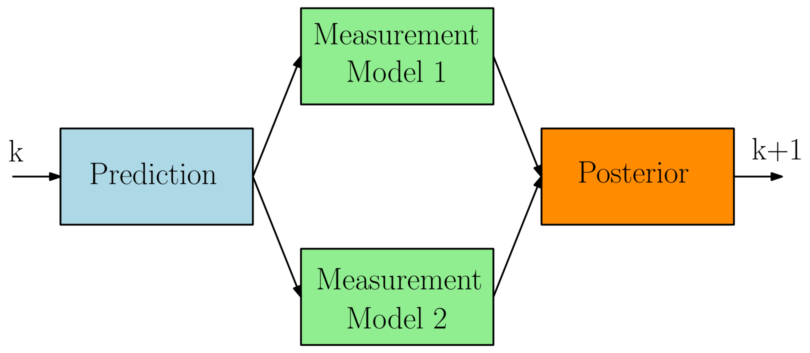


Figure 3.5: The unified prediction flow chart, where there is only one prediction and two measurement updates in each time cycle. The two measurement models merge by weighting of the likelihoods when forming a combined posterior.

3.2.4 Particle filters

The Kalman approach revolves around Gaussian distributions, or approximated Gaussian distributions for EKF. A non-parametric approach, which fit all types of models, i.e. nonlinear, multimodal etc, are particle filters. The posterior distribution is estimated by random samples from a proposed density, weighted according to how well the particle fit the measurements [20, p.24]. The particle filter posterior distribution is represented as

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^N \mathbf{w}^{(i)}_k \delta(\mathbf{x}_k - \mathbf{x}^{(i)}_k). \tag{3.30}$$

Here, $\mathbf{x}^{(i)}_k$ is a particle with the associated weight $\mathbf{w}^{(i)}_k$, at time instance k. N is the total amount of particles. The weights have to fulfil two criteria: the total sum of the weights is equal to one and all weights are non-negative. The criteria are mathematically interpreted as

$$\sum_{i=1}^N \mathbf{w}^{(i)}_k = 1 \ , \ \mathbf{w}^{(i)}_k \geq 0 \ \forall i \ , \ k = 1, ..., K. \tag{3.31}$$

Sometimes it is difficult to draw samples from a distribution $p(\mathbf{x})$, then a proposed density, $q(\mathbf{x})$, is introduced which should resemble the density which is to be estimated [37, p.8-9]. It is also required that the proposed density provide support for the density of interest. Importance-sampling is a Monte Carlo integration method where given a function of interest, $f(\mathbf{x})$, the expected value is approximated by

$$\mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})] \approx \frac{1}{N}\sum_{i=1}^{N} f_k(\mathbf{x}_{0:k}^{(i)})\frac{p(\mathbf{x}_{0:k}^{(i)}|\mathbf{z}_{1:k})}{q(\mathbf{x}_{0:k}^{(i)}|\mathbf{z}_{1:k})} = \sum_{i=1}^{N} f_k(\mathbf{x}_{0:k}^{(i)})\mathbf{w}_k^{(i)}. \tag{3.32}$$

The weights are normalized to still fulfil equation (3.31). Importance sampling is not a recursive method, and, thus, when new measurements are present the weights for the entire time sequence have to be recomputed. This is very unfavourable as time progresses, and it makes the algorithm intractable. Instead, a recursive version is utilized, which is known as sequential-importance sampling (SIS). SIS is an extension of importance sampling, where the proposed density is assumed to be

$$q(\mathbf{x}_{0:k}^{(i)}|\mathbf{z}_{1:k}) = q(\mathbf{x}_k^{(i)}|\mathbf{x}_{0:k-1}^{(i)}, \mathbf{z}_{1:k})q(\mathbf{x}_{0:k-1}^{(i)}|\mathbf{z}_{1:k-1}). \tag{3.33}$$

What equation (3.33) tells us is that the proposed density at time instance k is a marginal distribution for the proposed density at the previous time instance $k-1$. By combining the definition of the weights, according to equation (3.32), and (3.33), the recursive update for the weights can be expressed as

$$\mathbf{w}_k^{(i)} \propto \mathbf{w}_{k-1}^{(i)}\frac{p(\mathbf{z}_k|\mathbf{x}_k^{(i)})p(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)}|\mathbf{x}_{k-1}^{(i)}, \mathbf{z}_k)}. \tag{3.34}$$

There exist different algorithms for particle filters, but the principles are all based on SIS. The particles are propagated in a system over time, and the weights are updated whenever new measurements are available. It is common to let the particles propagate via a motion model, and then update the weights using the likelihood. Thus, this utilizes the prior and the particle propagation and weight update can be simplified to

$$\begin{aligned}\mathbf{x}_k^{(i)} &\sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^{(i)})\\ \mathbf{w}_k^{(i)} &\propto \mathbf{w}_{k-1}^{(i)}p(\mathbf{z}_k|\mathbf{x}_k^{(i)}).\end{aligned} \tag{3.35}$$

The weight update in equation (3.35) shows that a particle with a higher likelihood will get a higher weight in the next iteration and, thus, affect the posterior distribution more [20, p.24].

As time progresses, the SIS algorithm suffers from degeneracy, i.e. the posterior distribution will be incorrect due to few particles obtaining most of the weights. In order to avoid this problem, resampling is introduced. The idea of resampling is to revitalize particles with low weights, by mapping them to the particles with higher weights. Particles with lower weight represent the true state worse than particles with higher weight, and while these particles are important when evaluating the posterior, their contribution will be low due to the weight value. To avoid wasting computations on bad particles, binning low-weight particles to high-weight particles allow future importance sampling to be better. This can be done by, for example, sampling the particles from a uniform distribution where the probability of drawing a sample from particle is its associated weight, $\mathbf{w}_k^{(i)}$. The algorithm for the bootstrap particle filter is presented in [19, p.1173], which could be interpreted as sequential-importance resampling (SIR). In general it is disadvantageous to resample in every iteration, be it computational problems or propagating problems. The randomness in resampling also introduce an error, which unfold as an approximation error. Controlling this is essential, as resampling too often will result in loss of diversity of the estimator, while resampling to seldom will result in computations wasted on particles with unrepresentative states [41, p.85-86]. To solve this, some algorithms adapt to resample whenever a criterion is met [20, p.25].

3.2.5 Rao-Blackwellized particle filter

Particle filtering is a straight forward approach, but does, like many other implementations, have deficits. The main problem with particle filters are the rapid increase in computational complexity as the number of states increase. An approach to solve this problem for particle filters is described in [35]: marginalizing out linear states and combining each particle with a Kalman filter. By partitioning the state vector as

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_k^l \\ \mathbf{x}_k^n \end{bmatrix}, \tag{3.36}$$

the linear states, denoted with subscript l, are optimally estimated by Kalman filters, while the nonlinear states, denoted with subscript n, are estimated by the particle filter.

Algorithm 1 presents all the steps involved in the Rao-Blackwellized particle filter, which will be explained in more detail later. Firstly, an initialization of the particles and linear states are made, then the recursive part begins. The first step is to update the weights of the particles according to, e.g. a likelihood, and then normalize them. After this is complete, the particles are resampled. Now, Kalman filtering updates the linear states according to measurements of the linear states. Then, the particles are propagated and the final step, before reiterating the process, is to update the Kalman filter with the information from the particle filter prediction.

Algorithm 1 Rao-Blackwellized particle filter

```
1: Initialization: For $i = 1, ..., N$, initialize the particles according to $x_0^{n,(i)} \sim p_{x_0^n}(x_0^n)$, and set the linear states to $\{x_0^l, P_0\}$
2: For $i = 1, ..., N$, update the weights $w_k^{(i)} = p(y_k | X_k^{n,(i)}, Y_{k-1})$ and normalize the weights
       $$\tilde{w}_k^{(i)} = \frac{w_k^{(i)}}{\sum_{j=1}^{N} w_k^{(j)}}$$
3: Particle filter measurement update: Resample $N$ particles
       $$Pr\{x_{k|k}^{n,(i)} = x_{k|k-1}^{n,(j)}\} = \tilde{w}_k^{(j)}$$
4: Particle filter time update and Kalman filtering:
5:     a) Kalman filter measurement update, using equation (3.21)
6:     b) Particle filter time update: For $i = 1, ..., N$ propagate the particles.
       $$x_{k+1|k}^{n,(i)} \sim p(x_{k+1|k}^n | X_k^{n,(i)}, Y_k)$$
7: Kalman filter time update, using equation (3.40)
8: Set $k = k + 1$ and go to line 2.
```

By the state marginalization, it becomes evident that the models have to be revisited. By the partitioning, new equations are formed into a new model. Three different models are presented in [35], for three levels of generality. The second model is

$$\begin{aligned} \mathbf{x}_{k+1}^n &= f_k^n(\mathbf{x}_k^n) \quad + \mathbf{A}_k^n(\mathbf{x}_k^n)\mathbf{x}_k^l + \mathbf{w}_k^n \\ \mathbf{x}_{k+1}^l &= \qquad\qquad \mathbf{A}_k^l(\mathbf{x}_k^n)\mathbf{x}_k^l + \mathbf{w}_k^l \\ \mathbf{y}_k &= h_k(\mathbf{x}_k^n) \quad + \mathbf{H}_k(\mathbf{x}_k^n)\mathbf{x}_k^l + \mathbf{r}_k, \end{aligned} \tag{3.37}$$

which fits the problem in this thesis. The different matrices and functions in equation (3.37) connects the linear and nonlinear states, which can be utilized in, for example, the prediction step of the linear state variables. After the nonlinear prediction step, Algorithm 1 line 6, the information about the nonlinear states can be used to update the linear states. Mathematically interpreted, this is

$$\begin{aligned} \mathbf{x}_{k+1}^l &= \mathbf{A}_k^l(\mathbf{x}_k^n)\mathbf{x}_k^l + \mathbf{w}_k^l \\ \mathbf{z}_k &:= x_{k+1}^n - f_k^n = \mathbf{A}_k^n(\mathbf{x}_k^n)\mathbf{x}_k^l + \mathbf{w}_k^n, \end{aligned} \tag{3.38}$$

where $\mathbf{z}_k$ can be seen as a measurement accompanied by the measurement noise $\mathbf{w}_k^n$. Now, a part of last step of the Rao-Blackwellized approach can be performed, which is described by

$$\begin{aligned} \mathbf{S}_k &= \mathbf{A}_k^n \mathbf{P}_{k|k} (\mathbf{A}_k^n)^T + \mathbf{Q}_k^n \\ \mathbf{K}_k &= \mathbf{P}_{k|k} (\mathbf{A}_k^n)^T \mathbf{S}_k^{-1} \\ \hat{\mathbf{x}}_{k|k}^l &= \hat{\mathbf{x}}_{k|k}^l + \mathbf{K}_k(\mathbf{z}_k - \mathbf{A}_k^n \hat{\mathbf{x}}_k^l) \\ \mathbf{P}_{k|k} &= \mathbf{P}_{k|k} - \mathbf{K}_k \mathbf{S}_k (\mathbf{K}_k)^T \end{aligned} \tag{3.39}$$

as a Kalman measurement update. The update of the covariance matrix is computed in Joseph form, as the corresponding noise is not necessarily Gaussian and, thus, the Kalman filter cannot optimally compute the covariance with the standard approach. For linear models with additive Gaussian noise, Joseph form is simplified to the more commonly seen Kalman equation for posterior covariance computation. By combining the measurement update above with a prediction, the last step of Rao-Blackwellized particle filter is finalized as

$$\begin{aligned}
\mathbf{S}_k &= \mathbf{A}_k^n \mathbf{P}_{k|k} (\mathbf{A}_k^n)^T + \mathbf{Q}_k^n \\
\mathbf{K}_k &= \mathbf{A}_k^l \mathbf{P}_{k|k} (\mathbf{A}_k^n)^T \mathbf{S}_k^{-1} \\
\hat{\mathbf{x}}_{k+1|k}^l &= \mathbf{A}_k^l \hat{\mathbf{x}}_{k|k}^l + \mathbf{K}_k (\mathbf{z}_k - \mathbf{A}_k^n \hat{\mathbf{x}}_{k|k}^l) \\
\mathbf{P}_{k+1|k} &= \mathbf{A}_k^l \mathbf{P}_{k|k} (\mathbf{A}_k^l)^T + \mathbf{Q}_k^l - \mathbf{K}_k \mathbf{S}_k (\mathbf{K}_k)^T.
\end{aligned} \tag{3.40}$$

See [35, p.8-9] for the proof. Rao-Blackwellized can still be applied to systems with no linear relations [35, p.6]. Instead of standard Kalman filters, EKFs can be employed for the mild nonlinearities, while the particle filter estimates the strong nonlinearities and good estimations of the posteriors can be retrieved.

3.3 Vehicle model

In Bayesian theory, see Section 3.2, it was shown how the posterior can recursively be computed by a motion model and a measurement model. The goal is that the mathematical model can describe the vehicle dynamics/kinematics, while still being computationally tractable, and, thus, a suitable model has to be chosen. Vehicle kinematics can be modelled by a numerous amount of motion models. Simpler ones include, but is not limited to, constant velocity and constant acceleration, where more complicated nonlinear models include coordinated turn, constant turn and acceleration etc. [36] One can consider far more complicated dynamical models, see [38], but these models require good knowledge about material parameters and properties of the car, which in this thesis, are unknown.

Kinematic models are formed by setting up differential equations which describe the behaviour of the process in question, e.g. vehicles or robots [40]. These are then discretized, either analytically or approximated by e.g. the Euler method, so that they can be implemented. The idea of discretization is to express the distribution of $\mathbf{x}(t+T)$ given $\mathbf{x}(t)$. With continuous models on the form

$$\dot{\mathbf{x}}(t) = a(\mathbf{x}(t)) + \mathbf{q}(t), \tag{3.41}$$

we want to compute its discrete counterpart

$$\mathbf{x}_k = a(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1}. \tag{3.42}$$

Analytical solutions for nonlinear problems are often hard (sometimes not applicable), but Euler approximation is easier. We will cover an Euler method discretization of a general nonlinear model on the form presented in equation (3.41). The Euler method use the approximation

$$\dot{\mathbf{x}}(t) \approx \frac{\mathbf{x}(t+T) - \mathbf{x}(t)}{T}, \tag{3.43}$$

where T is the time interval between t_1 and t_2, which can be utilized when approximating the next state as

$$\mathbf{x}(t+T) \approx \mathbf{x}(t) + Ta(\mathbf{x}(t)) + T\mathbf{q}(t). \tag{3.44}$$

By simple identification, the discretized model is formed as

$$\mathbf{x}_k = \mathbf{x}_{k-1} + Ta(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1}, \tag{3.45}$$

where the noise is assumed to be constant during the time interval T.

3.4 Mapping

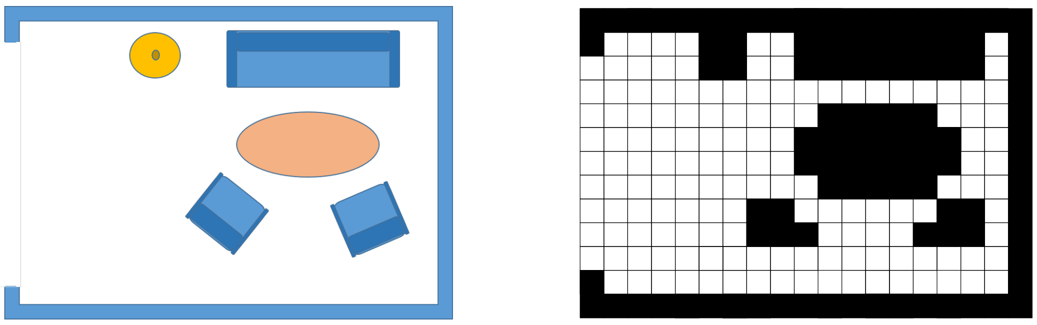
Mapping, in robotics, is a tool to store and fuse data from available exteroceptive sensors. A map of a dynamic environment will be time dependent as objects will move around between different time instances, but maps of the static environment are independent of time. This section will first cover the basics of static mapping, and then move on to occupancy grid mapping.

Mathematically, a map, $\mathbf{m}$, is a list of objects with properties that describe the environment, $\mathbf{m} = \{m_1, m_2, m_3, \ldots, m_N\}$ [41]. Maps are often used in localization and navigation. A map should not represent the physical environment as humans see it, but should instead represent the world as seen by the sensors. Mapping techniques thus depend upon which sensors that are used, but it also depend on the purpose of the map. This is intuitive, for example, a road map for cars with detailed information regarding roads, petrol vendors etc., does not look the same as a map used for orienteering, which has detailed information regarding topography and terrain. Maps are highly application dependant; the two aforementioned maps serve two different purposes, and contribute with different information to the user.

Five well known types of static environments mapping algorithms for robotics are mentioned in [14]: Kalman filter approaches, expectation maximization (EM) algorithms, hybrid approaches of the aforementioned two, object maps and occupancy grid maps. Mapping techniques can generally be divided into two subgroups depending on how they represent the environment, feature-based and location-based [41]. In feature-based methods, an object in the map, m_i where $i = 1, \ldots, N$, represents a landmark which contains properties such as position and extension. Common feature-based methods are Kalman filter approaches, where the landmark's extension in space is described by Gaussian probability distributions. In location-based methods, the map objects m_i represent positions in space and contain information about that location. Occupancy grid mapping is a common location-based method where space is divided into finite grids where each grid cell describe the probability of that space being occupied or free. The task in this thesis is to create maps used for localization purposes, and the occupancy grid map is a good framework for this. In accordance with our objective, we will only describe occupancy grid mapping more thoroughly.

3.4.1 Occupancy grid mapping

Occupancy grid mapping (OGM) is a commonly used mapping technique. It was first introduced by Elfes [10], where they employ sonar sensors to create a 2D map over an indoor environment. OGM segments the environment into grid cells with finite resolution, pixels in 2D and voxels in 3D, where every cell contains information regarding the probability of occupancy of that cell. A simple example of an OGM for an indoor environment is shown in Figure 3.6.



(a) Example of an indoor environment. **(b)** OGM of the indoor environment example.

Figure 3.6: A simple example of an OGM of an indoor environment. This OGM is represented by either occupied regions, black grid cells, or empty regions, white grid cells.

The OGM framework has both advantages and drawbacks. As a location-based method, it is able to not only describe object's position in the world, as feature-based methods, but also the absence of objects [41], usually denoted as empty or free space. This is favorable in navigation problems, as for example robots require knowledge of free spaces to calculate possible routes. In feature-based methods, an association between landmarks and measurements is needed, which is not required by OGM, but to the expense that OGM generally has a larger state vector to cover the whole space. OGM can also handle any type of sensor noise [14] and is a good framework for low level sensor fusion [30]. Feature-based methods have an advantage when creating maps with noisy data, especially pose uncertainty, because it is easier to adjust landmarks position in space. This is desirable when the map is created in real time, as in SLAM problems, and the landmarks need to be adjusted with new sensor measurements. For the same reason, feature-based methods are also more natural for dynamic maps. OGM also have a natural loss in accuracy due to discretization of the environment.

The static OGM is denoted with the matrix $\mathbf{m}$, and a certain cell in the map is denoted m_i. Each cell is a discrete random variable with two states, occupied (occ) and empty (emp). It is common that the states have the relation $P(m_i = occ) = (1 - P(m_i = emp))$, thus it is only necessary to store the information of one state in each cell, normally by $P(m_i = occ)$ or the ratio $\frac{P(m_i=occ)}{1-P(m_i=occ))}$. The variable $\mathbf{z}_{1:k}$ are measurements from the exteroceptive sensors up to time instance k, and the pose vector, $\mathbf{x}_{1:k}$, contains the position and orientation of the vehicle, which is assumed to be deterministically known. We denote distributions as $p(\cdot)$ and probabilities as $P(\cdot)$.

The goal of the OGM (and mapping algorithms in general) is to calculate the posterior of the map given the measurement and the state vector,

$$p(\mathbf{m}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}). \tag{3.46}$$

This posterior for OGM is generally intractable to compute due to the huge number of possible map combination; a map with the dimension of $\mathbb{R}^{i,j,k}$ has $2^{i \times j \times k}$ combinations. There are two main approaches to solve this [41], which we will refer to as *inverse sensor model* approach and *forward sensor model* approach.

The considered standard method, the inverse sensor model, makes an assumption that each grid cell is independent of other cells. The mapping problem is thus changed to calculate the posterior of

$$p(\mathbf{m}|\mathbf{z}_{1:k}, \mathbf{x}_{1:k}) = \prod_i P(m_i|\mathbf{z}_{1:k}, \mathbf{x}_{1:k}), \tag{3.47}$$

for all grid cells. Each cell can then be estimated with a binary Bayes' filter (see Section 3.4.2). The independency assumption is very, very strong and in many senses incorrect. An object usually covers multiple grid cells, so there is a physical dependency between them. However, as mentioned in [10], you could also argue that for a static map there are no causal relation between the occupancy state of different grids. Also, for many applications this assumption generate sufficient models which are very computationally efficient. Different methods have been proposed to solve this problem [10, 17], and the theory presented further in this report is based on the [41] which uses the heuristic inverse sensor model.

This method, as the name implies, builds upon the use of an inverse sensor model,

$$P(m_i|\mathbf{z}_k, \mathbf{x}_k). \tag{3.48}$$

It is hard to motivate the use of an inverse sensor model, because it uses effect to cause instead of cause to effect. This is not a statistically correct way to make conclusions, as multiple causes might cause an effect. As mentioned in [41], inverse sensor model is often used when the states are simple but the space of all possible measurements are huge. This is the case for the mapping problem where the grid cells states is simple, but the number of possible map configuration that could describe a measurement is huge. Inverse sensor models can also be motivated when the measurement uncertainty is low, as often for lidar, and the measurement directly maps the true environment.

As opposed to the inverse sensor model, there exist methods which do not require the assumption of independent grid cells. Instead of solving true posterior the idea is to maximize the likelihood. The methods use the forward sensor model,

$$p(\mathbf{z}_{1:k}|\mathbf{m}, \mathbf{x}_{1:k}), \tag{3.49}$$

which calculates the likelihood of the measurement conditioned on the map. For a specific map, likelihood functions are created for each detection up to time instance k. There are different methods that can solve this maximization problem. In [41] and [42] a forward sensor model method is used to create a map from sonar sensor measurements. Here, they are using expectation maximization (EM) to find a map that maximize a combined likelihood.

There are both pros and cons with forward sensor models for OGM. It is better than the inverse sensor model method to handle conflicting measurement, especially when the measurement in a region is sparse, as clearly shown in [42]. By keeping the dependency between the grid cells, more exact shape recovery of the environment is possible [41]. As can be seen in equation (3.49) the model cannot directly be implemented recursively due to the fact that the forward sensor model require all the measurement up to time instance k. This in combination with the use of search methods to find the posterior, makes this approach much more computational heavy than the standard method. The map created by the forward sensor model in [41] and [42] also has binary cell values, occ or emp, in comparison to inverse sensor model where the cells contains the log-odds ratio for occupancy (more details in section 3.4.2). In [42] the forward sensor model is also complemented with a method to estimate the uncertainty of the posterior.

3.4.2 Algorithm of the inverse sensor model approach for occupancy grid mapping

In this section the algorithm for the inverse sensor model approach is described. Each cell is assumed to be independent, and the posterior to calculate in accordance with equation (3.47). The method is based on a binary Bayes filter and a static map [41]. Before the calculation of the posterior, the idea of log-odds ratio is presented.

As previously mentioned, the two possible states in the OGM are each others complement, $P(m_i = occ) = (1 - P(m_i = emp))$, which will be denoted $P(m_i) = (1 - P(\neg m_i))$. One way of representing the state of the cell is by the ratio

$$\frac{P(m_i)}{P(\neg m_i)} = \frac{P(m_i)}{1 - P(m_i)}. \tag{3.50}$$

The log-odds ratio is then given by taking the logarithm of the ratio in equation (3.50), which yields

$$l_i = \log\left(\frac{P(m_i)}{1-P(m_i)}\right), \tag{3.51}$$

where l_i is the notation for the log-odds in grid i. We will show that the log-odds ratio is a computationally efficient method, and that it also avoids numerical truncation problems when the probabilities approach 0 and 1. The occupancy probability of state can easily be recovered by

$$P(m_i) = 1 - \frac{1}{1+e^{l_i}}. \tag{3.52}$$

Now that log-odds has been introduced, we switch focus to the problem of calculating the posterior in equation (3.47). Since the pose, $\mathbf{x}_{1:k}$, is deterministically known, it is omitted in the following equations. The posterior of one grid is then

$$P(m_i|\mathbf{z}_{1:k}). \tag{3.53}$$

Bayes rule is applied to equation (3.53), which yields

$$P(m_i|\mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k|m_i,\mathbf{z}_{1:k-1})P(m_i|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})}, \tag{3.54}$$

where the measurement model can be rewritten as

$$p(\mathbf{z}_k|m_i,\mathbf{z}_{1:k-1}) = p(\mathbf{z}_k|m_i), \tag{3.55}$$

due to the static world assumption. By applying Bayes rule a second time on the measurement model $p(\mathbf{z}_k|m_i)$ we get

$$P(m_i|\mathbf{z}_{1:k}) = \frac{P(m_i|\mathbf{z}_k)p(\mathbf{z}_k)P(m_i|\mathbf{z}_{1:k-1})}{P(m_i)p(\mathbf{z}_k|\mathbf{z}_{1:k-1})}. \tag{3.56}$$

In the same way as equation (3.54) - (3.56), the complemented posterior can be calculated as

$$P(\neg m_i|\mathbf{z}_{1:k}) = \frac{P(\neg m_i|\mathbf{z}_k)p(\mathbf{z}_k)P(\neg m_i|\mathbf{z}_{1:k-1})}{P(\neg m_i)p(\mathbf{z}_k|\mathbf{z}_{1:k-1})}. \tag{3.57}$$

By taking the odds ratio between the two posterior, a few factors will be cancelled and the fraction becomes

$$\begin{aligned} \frac{P(m_i|\mathbf{z}_{1:k})}{P(\neg m_i|\mathbf{z}_{1:k})} &= \frac{P(m_i|\mathbf{z}_k)}{P(\neg m_i|\mathbf{z}_k)}\frac{P(m_i|\mathbf{z}_{1:k-1})}{P(\neg m_i|\mathbf{z}_{1:k-1})}\frac{P(\neg m_i)}{P(m_i)} \\ &= \frac{P(m_i|\mathbf{z}_k)}{(1-P(m_i|\mathbf{z}_k))}\frac{P(m_i|\mathbf{z}_{1:k-1})}{(1-P(m_i|\mathbf{z}_{1:k-1}))}\frac{(1-P(m_i))}{P(m_i)}. \end{aligned} \tag{3.58}$$

Now, taking the logarithm of the odds ratio will form the posterior log-odds as

$$l_i^{1:k} = l_i^k + l_i^{1:(k-1)} - l_i^0. \tag{3.59}$$

The resulting posterior for each grid cell can thus be recursively computed for each new measurement. Only two terms needs to be defined: l_i^0 based on the initial prior $P(m_i)$, and l_i^k based on the inverse sensor model $P(m_i|\mathbf{z}_k)$. The initial prior $P(m_i)$ is usual set to 0.5, which implies no initial knowledge of the probability of occupancy for each grid cell. For the term l_i^0 in equation (3.59) this will yield

$$l_i^0 = \log \frac{P(m_i)}{(1 - P(m_i))} = \log \frac{0.5}{(1 - 0.5)} = 0, \tag{3.60}$$

and will hence not affect the equation. The grid cells will be updated by the inverse sensor model $P(M_i|\mathbf{z}_t)$ which will affect equation (3.59) by

$$l_i^k = \log \frac{P(m_i|\mathbf{z}_k)}{(1 - P(m_i|\mathbf{z}_k))} \begin{cases} = 0 & \text{if } P(m_i|\mathbf{z}_k) = 0.5 \\ > 0 & \text{if } P(m_i|\mathbf{z}_k) > 0.5 \\ < 0 & \text{if } P(m_i|\mathbf{z}_k) < 0.5 \end{cases}. \tag{3.61}$$

From equations (3.59) and (3.61) we note how the inverse sensor model recursively estimate static objects and free areas. Each update is computed by addition, which is very computationally efficient. Further, the logarithm ensures that we do not reach numerical problems when approaching probabilities near 0 and 1, which with multiplication of probabilities would create numbers with too many decimals to handle. It handles measurement from dynamic objects and conflicting measurement by simply adding up positive and negative evidence [41], which makes it sensitive to the ratio between the two types of evidence. Dynamic objects will, by definition, not accumulate many measurements for one position, which makes the log-odds ratio very capable of handling them. However, the method cannot resolve conflicting measurements of static objects well.

3.4.3 Building the inverse sensor model

Based on equation (3.61), the inverse sensor model for OGM is usually built around the idea that grid cells close to the measurement are possibly occupied, and are given values higher than 0.5. Grid cells that are between the sensor and the measurement are probably free and given values lower than 0.5 and the remaining grid cells, which the measurement do not provide any information about, gets a value of 0.5. For an ideal sensor, when the sensor directly describes the true environment, the inverse sensor model is completely valid. But as sensors are not ideal, the sensor noise have to be incorporated in the models. There are different approaches to incorporate sensor uncertainties in the model. In [41] they introduce a learning algorithm to estimate the inverse sensor model based on samples from a forward model, in [14] the inverse sensor model is built by convolution between the ideal sensor model and a Gaussian function describing the sensor noise, while in [30] the inverse sensor model is built by combining two separate functions describing occupied respective free space. In Figure 3.7, both an ideal and a noisy inverse sensor model for a one

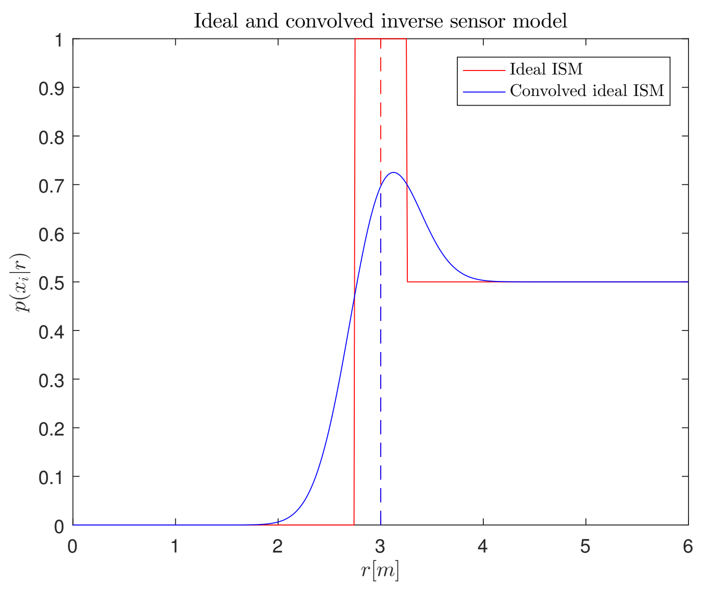


Figure 3.7: This plot shows an ideal and a convolved inverse sensor model for an one dimensional OGM based on the range reading $r = 3$ [m]. The convolution is between the ideal sensor model and a Gaussian with variance of 0.09 [m]. The convolved model is en example of how to incorporate sensor noise in the inverse sensor model.

dimensional OGM are shown. The inverse sensor model with measurement noise is created by the convolution method in [14].

As a last remark about the derived OGM, with inverse sensor model, is that the resulting map reminds of a discrete version of a likelihood field. Likelihood field is a heuristic method which introduce sensor noise in the map and enables the map to directly be incorporated as an likelihood [41]. The presented inverse sensor model incorporate sensor noise directly into the map in a similar way.

3.5 Registration

The art of aligning two separate data sets is referred to as registration [3]. The idea of registration is to find the transformation between two data sets that minimize the difference between them. A simple example is shown in Figure 3.8, where two images contain matching objects. The two images can be aligned by a rotational and translational transformation. In [3] some of the most common registration techniques are summarized. Principal Component Analysis (PCA) is a method which align the biggest variance of two data sets. Singular Value Decomposition (SVD) is a method where point correspondences are made between the images and the sum of Euclidean distance between the points is minimized by solving a least squares problem. A popular method is Iterative Closest Point (ICP), which uses the SVD method but iteratively updates the point correspondence and removes outliers. Normal Distribution Transform (NDT) is a method where one one of two point clouds is transformed to a set of normal distributions and the optimal transformation is found by maximizing the likelihood between the two images [4].

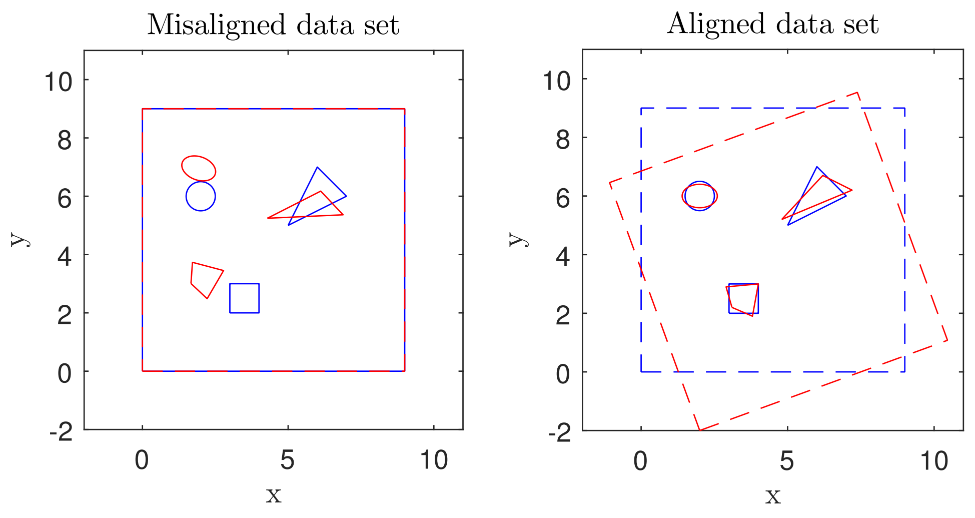


Figure 3.8: Example of a simple registration between two images, red and blue. There are similar objects (a triangle, a square and a circle) in the two images but at different positions in their local frames. The left figure show the two images on top of each other without any transformation between the two frames. In the right figure, the objects are aligned by a translational and rotational transformation between the two frames.

Different registration techniques are not applicable for all data sets, and this has to be taken into account when choosing the technique. Based on the data sets, the user also have to decide how the difference, or error, between the two sets should be calculated. For two point clouds, the error could be the distance between corresponding points. A requirement for a registration is that a transformation to align the data sets must be defined.

3.5.1 Registration for localization

If a registration should be made where the goal is to retrieve the position of an object, e.g. a robot or a vehicle, the alignment between a map and sensor measurements will be described by a rigid body transformation. Given a static world, the remaining error after alignment is the noise of the map or the measurements. The problem can be seen as finding the best local pose conditioned on the measurements, i.e. maximizing a likelihood [5]. The best found pose can then be used as input to a filter. Instead of computing a transformation between two data sets, an iterative search algorithm can be used which finds the optimal pose given a certain prior. This can be mathematically expressed as

$$\arg\max_{\mathbf{x}} f(\mathbf{x}), \tag{3.62}$$

where f is a chosen cost function, e.g. a likelihood.

3.5.2 Optimization steepest ascent

In unconstrained optimization, a gradient search algorithm which iteratively finds a local maximum is steepest ascent. In each iteration, the next point of the series is found by computing the gradient in the current point, and taking an α long step in that direction. Steepest descent is presented in [28, p.282-283], which is essentially the same as steepest ascent, with the exception of a negative sign used to find a minimum instead of a maximum.

Some approaches use steepest ascent as a line search, where the step length varies. However, a set step length can be set and used throughout all iterations. In order to decrease the effect of the function itself, the gradient can be normalized. This way, only the step length will determine how far the next step will travel.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \nabla f(\mathbf{x}_k) \tag{3.63}$$

is the iterative update formula for steepest ascent, where $\mathbf{x}_k$ are the state variables, α_k is the step length, and $\nabla f(\mathbf{x}_k)$ is the gradient evaluated at $\mathbf{x}_k$.

For unconstrained optimization, given a convex function the optimality criterion is fulfilled the gradient is equal to zero. [28, p.96] As this rarely happens in numerical approaches, a small interval, ϵ, can chosen to determine whether a local optima has been found or not. Further, just comparing the norm of the gradient with zero can be bad due to factors such as badly scaled variables, function values and gradients. Instead, one can introduce stopping criterion which takes this into consideration. [28, p.297]

4

Modelling of internal sensors and vehicle kinematics

This chapter describes the chosen vehicle model, detailing and motivating the choices behind it. The noise discretization for the motion model is derived afterwards. We also derive the measurement model for the internal sensors and the IMU, as well as describe some required computations with the reference system.

4.1 Vehicle model

When choosing the most appropriate vehicle model, we first take a step back and identify which parameters that should be estimated. This thesis considers vehicle kinematics and not vehicle dynamics, mainly because the main focus is not to control or steer the car by forces/torques. It is rather to predict the behaviour of the vehicle, and for this kinematics work well. A not so common practise for vehicle state estimation is 3D. This has to be taken into consideration since it implies six degrees of freedom for rigid body transformations. By investigating the CAN-bus information, to see which signals are available, and arguing that more simple models are insufficient for the required vehicle dynamics in this thesis, the desired states for the vehicle are chosen as presented in equation (4.1). The states, in corresponding order are, global positions in x, y and z, yaw, pitch, roll, yaw rate, pitch rate, roll rate, heading velocity and acceleration. From here on out, X, Y and Z are used when describing the position of the vehicle in the inertial frame.

$$\mathbf{x} = \begin{bmatrix} X & Y & Z & \psi & \theta & \phi & \dot{\psi} & \dot{\theta} & \dot{\phi} & v & a \end{bmatrix}^T \tag{4.1}$$

From [36] the constant turn rate and acceleration (CTRA) was deemed most appropriate as it can model turning and change in velocity despite a relatively low complexity. The velocity and acceleration are defined in heading direction. The resulting motion model is

$$\underbrace{\begin{bmatrix} X_k \\ Y_k \\ Z_k \\ \psi_k \\ \theta_k \\ \phi_k \\ \dot{\psi}_k \\ \dot{\theta}_k \\ \dot{\phi}_k \\ v_k \\ a_k \end{bmatrix}}_{\mathbf{x}_k} = \underbrace{\begin{bmatrix} X_{k-1} + Tv_{k-1}\cos\theta_{k-1}\cos\psi_{k-1} \\ Y_{k-1} + Tv_{k-1}\cos\theta_{k-1}\sin\psi_{k-1} \\ Z_{k-1} - Tv_{k-1}\sin\theta_{k-1} \\ \psi_{k-1} + T\dot{\psi}_{k-1} \\ \theta_{k-1} + T\dot{\theta}_{k-1} \\ \phi_{k-1} + T\dot{\phi}_{k-1} \\ \dot{\psi}_{k-1} \\ \dot{\theta}_{k-1} \\ \dot{\phi}_{k-1} \\ v_{k-1} + Ta_{k-1} \\ a_{k-1} \end{bmatrix}}_{f(\mathbf{x}_{k-1})} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ q^{\dot{\psi}}_{k-1} \\ q^{\dot{\theta}}_{k-1} \\ q^{\dot{\phi}}_{k-1} \\ q^{v}_{k-1} \\ q^{a}_{k-1} \end{bmatrix}}_{\Gamma q_{k-1}}, \tag{4.2}$$

where T is the time interval between predictions. It is derived by Euler discretization, where the cosine and sinusoidal terms enter via the rotational matrix mapping the local coordinate system of the car to a global frame. By Taylor expansion, the linearized motion model matrix,

$$\mathbf{F}(\mathbf{x}_k) = \begin{bmatrix} 1 & 0 & 0 & -Tv\cos\theta\sin\psi & -Tv\sin\theta\cos\psi & 0 & 0 & 0 & 0 & T\cos\theta\cos\psi & 0 \\ 0 & 1 & 0 & Tv\cos\theta\cos\psi & Tv\sin\theta\sin\psi & 0 & 0 & 0 & 0 & T\cos\theta\sin\psi & 0 \\ 0 & 0 & 1 & 0 & -Tv\cos\theta & 0 & 0 & 0 & 0 & -T\sin\theta & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & T & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & T & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & T & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.3}$$

is retrieved and used in the EKF equations. **F** is the Jacobian matrix, i.e. all first partial derivatives of each equation. In [12] they show that a discretized linearization, which is a discretization of the model before linearizing, produce less errors than a linearized discretization when polar velocity is used. These circumstances apply to our models, thus, we perform discretized linearization.

4.1.1 Noise discretization

Discretization of the model does not only apply to the kinematics, but also to the noise. We derive the sampling of the noise in accordance to [12]. The choice of the noise sampling is not as important as other factors in an EKF approach, linearization and discretization errors affect the resulting estimations more. We argue that

$$\hat{Q} = TFQF' \tag{4.4}$$

where Q is the covariance matrix, is a good choice. This means that the noise travel through the motion model, which means that correlating states will be effected faster by the noise than if $\hat{Q} = TQ$ is used. Since there exist no perfect models, a larger noise is preferable, which is a side-effect of using the model of our choice.

4.2 Sensor modelling of IMU

In this section we will describe the sensor model for the internal sensors, but also explain the reference system and how some modifications were made in order to retrieve the desired output from it. The sensor model for the IMU is the same for both localization approaches. The same measurement model is also used in the simulation environment to mimic the real case as close as possible.

4.2.1 Oxford RT 2000

In order to evaluate the performance of the state estimation, some means of generating ground truth data is required. The Oxford RT 2000 can provide accurate estimates for all the states that we consider, and is a convenient tool which we utilize. However, not all states are measured directly and have to be calculated. The three gyroscope sensors measure angular rates around fixed axes and have to be transformed into yaw, pitch and roll rates, as they are not equal to one another. The reason for the conversion is to retrieve the derivatives of the yaw, pitch and roll states that we are interested in. From the gyroscope, let ω_x, ω_y and ω_z be the outputs for angular velocity around x-, y- and z-axis respectively. [39] present the connection between Euler angles and body-axis rates, which forms the transformations as described by

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \omega_x + \omega_y \sin\phi \tan\theta + \omega_z \cos\phi \tan\theta \\ \omega_y \cos\phi - \omega_z \sin\phi \\ \omega_y \sin\phi \sec\theta + \omega_z \cos\phi \sec\theta \end{bmatrix} \tag{4.5}$$

and

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{\phi} - \dot{\psi} \sin\theta \\ \dot{\theta} \cos\phi + \dot{\psi} \sin\phi \cos\theta \\ -\dot{\theta} \sin\phi + \dot{\psi} \cos\phi \cos\theta \end{bmatrix}. \tag{4.6}$$

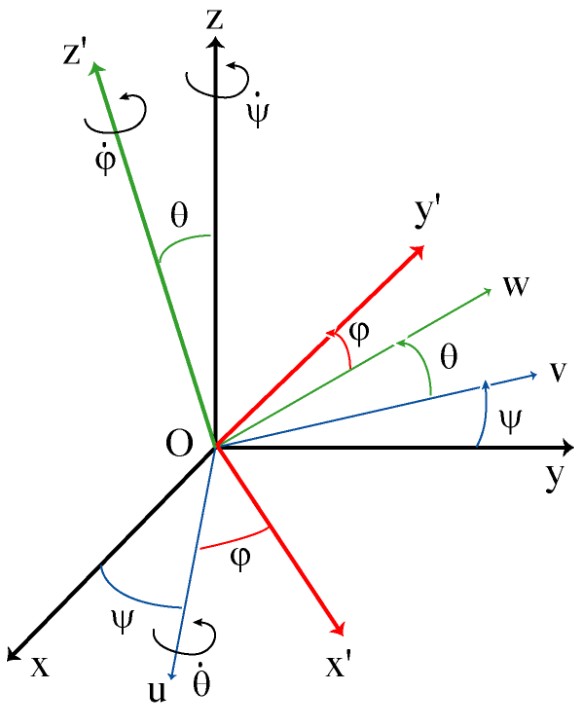


Figure 4.1: Depiction of the connection of the Euler angles.

These transformations are based on a set order of rotations for the Euler angles, which are interconnected (see Figure 4.1). In this approach, the rotations are first around z-axis, followed by rotation around the y-axis and lastly the x-axis. This is motivated by that we want to map the yaw angle, or heading, directly to the angle around body fixed z. This is the reason why ψ is not present in the transformation equations above.

4.2.2 Measurement model for IMU

The internal sensors in the car send their information on CAN, which can be accessed and read. We want to estimate three dimensional rotations and their rates, but the internal sensors do not measure pitch rate and roll rate. Instead, we utilize these two measurements from the Oxford RT 2000. We motivate this with two arguments. Firstly, to model vehicle kinematics, a constant pitch and roll is not reasonable. There exist hills, slopes and velodrome curves, which if we cannot model, will create erroneous maps when using the radar for example. Secondly, the pitch and roll rates are not large, they are in general small. Thus, they do not affect the kinematics too much, but still enable the vehicle to behave more realistic. Ideally, the vehicle that is used should measure these states, but given the prerequisites this is not applicable and we make the best of the situation.

With an Bayesian approach in consideration, a measurement model on the form $h(\mathbf{x}_k)$, is formed. From the internal sensors, the following measurements are retrieved: body fixed angular rates around x, y and z, velocity, and acceleration. The measurement equations for the angular rates originate from equation (4.6). The velocity measurements originate from the back wheels, and is thus assumed to always be in heading. Where the acceleration measurements are made is unknown, and we assume they are in heading. The mathematical model

$$h(\mathbf{x}_k) = \begin{bmatrix} \dot{\phi}_k - \dot{\psi}_k \sin\theta_k \\ \dot{\theta}_k \cos\phi_k + \dot{\psi}_k \sin\phi_k \cos\theta_k \\ -\dot{\theta}_k \sin\phi_k + \dot{\psi}_k \cos\phi_k \cos\theta_k \\ v_k \\ a_k \end{bmatrix}, \tag{4.7}$$

is nonlinear and the Jacobian,

$$H(\mathbf{x}_k) = \begin{bmatrix} \mathbf{0}_{5x4} & \begin{bmatrix} -\dot{\psi}_k \cos\theta_k & 0 & -\sin\theta & 0 & 1 & 0 & 0 \\ -\dot{\psi}_k \sin\phi_k \sin\theta_k & -\dot{\theta}_k \sin\phi_k + \dot{\psi}_k \cos\phi_k \cos\theta_k & \cos\theta \sin\phi & \cos\phi & 0 & 0 & 0 \\ -\dot{\psi}_k \cos\phi_k \sin\theta_k & -\dot{\theta}_k \cos\phi_k - \dot{\psi}_k \sin\phi_k \cos\theta_k & \cos\theta \cos\phi & -\sin\phi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{bmatrix}, \tag{4.8}$$

is used in the EKF measurement updates.

5
Map generation and radar sensor modelling

This chapter concerns our choice of radar sensor model and how we form the occupancy grid map, OGM. The chapter starts with coordination transformations between the polar domain and the Cartesian domain. It continues with how the map is formed, followed by the inverse sensor model (radar model). Thirdly, the grid cell update is presented and lastly we present how the likelihood can be formed by the OGM. To get a good overview of the map generation, a flow chart describing how the parts are connected is presented in Figure 5.1.

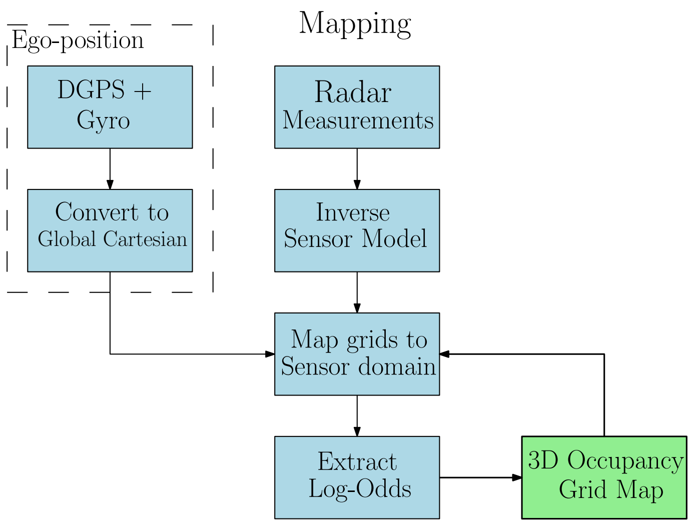


Figure 5.1: The flowchart describes how the different parts required to create the map are connected.

5.1 Radar domain and coordinate transformation

After the signal processing of the radar measurements, i.e. range-Doppler computations, CA-CFAR and MUSIC are done, we receive an array of radar detections of various attributes. The CA-CFAR provides information about target detections in range, r_{det}, Doppler and P_d, while MUSIC computes directional cosines with which the direction of arrival can be determined. The directional cosine relative the x-axis is u_{det}, while v_{det} is the directional cosine relative the y-axis. The antenna perspective is shown in Figure 5.2, where the black surface is the radar antenna, the red star is the detection and the cyan and green arcs are the directional angles for u_{det} and v_{det} respectively.

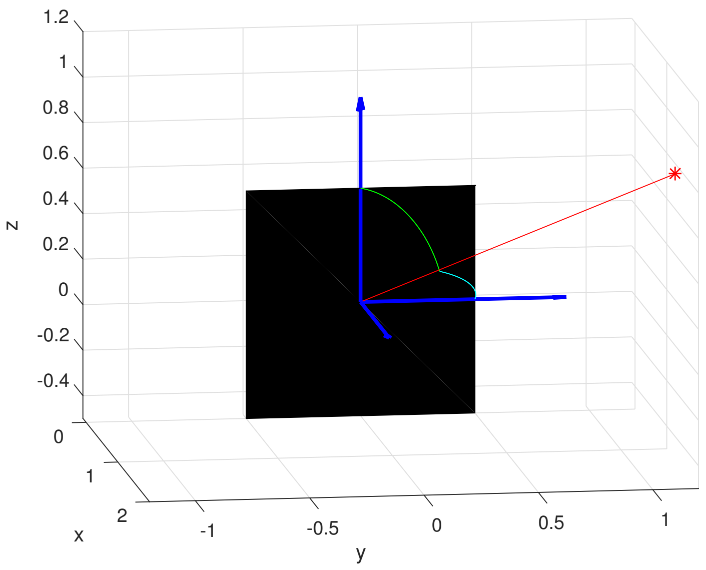


Figure 5.2: Radar antenna coordinate system, displaying how a detection at range r and directional angles for u (cyan) and v (green) are mapped to a local Cartesian frame. The red star is the detection and the black surface is the radar antenna.

A first screening can be done of the radar detections by thresholds on the radar data. Especially, dynamical objects can be filter out with the Doppler information. Doppler can be converted to relative velocity $v_{relative}$ and by comparing with the ego velocity, static targets are estimated by

$$\hat{v}_{target} = v_{relative} + v_{ego}\sqrt{1-u_{\text{det}}^2}\sqrt{1-v_{\text{det}}^2} + \dot{\psi}Lu_{det}\sqrt{1-v_{\text{det}}^2}, \tag{5.1}$$

where L is the distance between the radar position and the rear axis. Pitch rates are so low that they do not affect the equation, but can be included if necessary. If $\hat{v}_{target}$ is close to zero, it is assumed to be a static object.

When trying to match detections to an existing map, the detections are mapped between the sensor domain and the global Cartesian frame (map domain). From the array data, directional cosines, which originate from the local coordinate frame of the radar, are together with range information converting the detections to a local vehicle frame by

$$\begin{aligned} x_{det} &= r_{det}\sqrt{1-u_{\text{det}}^2-v_{\text{det}}^2} + x_{radar} \\ y_{det} &= r_{det}u_{det} + y_{radar} \\ z_{det} &= r_{det}v_{det} + z_{radar} \end{aligned} \tag{5.2}$$

where x_{radar}, y_{radar} and z_{radar} are the offsets of the radar relative to the origin of the vehicle frame, which is defined in the rear axis of the vehicle. These local coordinates are then transformed to a global frame by the rotational matrix

$$R(\psi,\theta,\phi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \quad (5.3)$$

which correspond to three rotations in the order of yaw, pitch and lastly roll. $R(\psi,\theta,\phi)$ is multiplied by the local coordinates to get the global positions as

$$\begin{bmatrix} X_{det} \\ Y_{det} \\ Z_{det} \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + R(\psi,\theta,\phi) \begin{bmatrix} x_{det} \\ y_{det} \\ z_{det} \end{bmatrix}. \quad (5.4)$$

Thus, we have created a way to match a radar detection to the global map. By utilizing the orthogonality of rotational matrices, the transpose R^T can be utilized when the inverse transformation has to be done, which is used when creating the map. Section 5.2 will explain how the map is formed, and further clarify the transformational differences between creating a map and matching a detection to the map.

The uncertainty of the radar will be modeled as a multivariate normal distribution with uncorrelated states,

$$\mathbf{z}_{det} = \begin{bmatrix} r_{det} \\ u_{det} \\ v_{det} \end{bmatrix} + \mathbf{r}_{radar}, \quad (5.5)$$

where the noise, r_{radar}, is distributed as

$$\mathbf{r}_{radar} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_u^2 & 0 \\ 0 & 0 & \sigma_v^2 \end{bmatrix} \right). \quad (5.6)$$

This will be utilized when deriving the inverse sensor model in section 5.2.1.

5.2 Map generation

Occupancy grid mapping is a part of the scope of this thesis, but should of course be motivated as a valid choice for the problem. The map generated for this thesis should: cover the static environment, show the world as seen by the radar, and not perform object identification. The map is also generated with known pose of the radar. With the information given in Section 3.4.1, this motivate OGM as a valid choice for the map framework. The map is created as a 3D matrix, $\mathbf{m} \in \mathbb{R}^3$. One grid cell is denoted $m_{a,b,c}$, where each sub-index represent a global position in space. The distance between each grid cell is defined as the grid size Δm. The connection between index and global position is done by storing the position of the first index: X_0, Y_0 and Z_0, in the matrix and using the grid size to compute

$$\begin{aligned} a &= \lfloor \frac{X_i - X_0}{\Delta m} \rceil + 1 \qquad & X_i \geq X_0, \\ b &= \lfloor \frac{Y_i - Y_0}{\Delta m} \rceil + 1 \qquad & Y_i \geq Y_0, \\ c &= \lfloor \frac{Z_i - Z_0}{\Delta m} \rceil + 1 \qquad & Z_i \geq Z_0, \end{aligned} \tag{5.7}$$

where the symbols $\lfloor \cdot \rceil$ is a rounding to the closest integer. For simplicity, a specific index in the 3D map matrix will henceforth be described as earlier $m_{a,b,c} = m_i$.

The transformation between the polar domain and OGM Cartesian frame was derived in Section 5.1, and with those calculations we will explain the concept of how to map radar detections to the OGM, before going into more detail in the rest of this section. By the log-odds values, given by the sensor model, we want to update the OGM recursively. The sensor model is in the polar grid, and to avoid nonlinear mapping of uncertainties, the OGM grids will be mapped into the polar domain. Grid values are transformed into the sensor domain, where each Cartesian coordinate is mapped to a polar coordinate,

$$\{X_i, Y_i, Z_i\} \Longleftrightarrow \{r_i, u_i, v_i\}. \tag{5.8}$$

The log-odds value of each corresponding polar point is then extracted back to the Cartesian domain where the corresponding grid cells obtain the values. The mathematics behind equation (5.8) will be explained in Section 5.2.1 and Section 5.2.2.

How an algorithm forms the map is directly related to, and dependant on, the sensor model. The inverse sensor model (ISM) creates a map model with probabilistic representation of the grid cells, which suits the task of localization. The ISM creates a map where the sensor noise is incorporated, making a likelihood evaluation very simple (mathematically), which can be seen in section 5.3. Each cell contains the probability of occupancy, grid cells close to objects accumulate higher probability, and the probability decreases farther away from landmarks. This is a similar concept to likelihood fields [5], but here the map does not have to be transformed. Another advantage of using ISM is that, for localization, we do not have to make a hard decision regarding occupied or free space, making some information from the forward model obsolete. The forward sensor model approach is a physically correct way of creating an OGM, or any map for that matter, and will provide a better map. The forward sensor model gives deterministic information regarding the occupancy/empty space of grid cells, which would be favorable in path planning. Path planning is, however, not within the scope of this thesis. Uncertainties can be estimated from the binary map, but to create a likelihood for radar given a specific map configuration is quite complex compared to the inverse sensor model. The forward sensor model approach is also more computationally heavy due to the fact that it cannot be computed recursively. These arguments, together with the theory in Section 3.4.1 provide enough motivation to choose the inverse sensor model.

5.2.1 Inverse sensor model

The inverse sensor model, $P(m_i|\mathbf{z}_k)$, used in this thesis is mainly based on the method described in [30], which creates the inverse sensor model by combining two separate functions describing occupied respective free space, but also with influences from the convolution method in [14] which includes measurement noise in the ideal sensor model in a preferable way. The reason for not using the convolution method directly is that it has a physical drawback. Because of the free space in front of the detection, the top of the inverse sensor model will have an offset toward the unknown space and not peak around the detection. This can clearly be seen in Figure 3.7. A detection will only indirectly provide information of free space, hence it is important that the free space information do not interfere with the sensor uncertainty. The separate modelling in [30] allow us to model them without interference. The separate functions also allow a neat introduction of probability of detection, P_d, from equation (3.7). A low P_d imply that the there is little information about the detection, and should render the probability of both occupied and free space to 0.5. The inverse sensor model used in this thesis is then

$$P(m_i|\mathbf{z}_k) = \frac{1}{2}(1 + P_d f_{occ}(m_i, \mathbf{z}_k) - P_d f_{emp}(m_i, \mathbf{z}_k)), \quad P_d, f_{occ}, f_{emp} \in [0 \quad 1], \tag{5.9}$$

where $f_{occ}(m_i, \mathbf{z}_k)$ and $f_{emp}(m_i, \mathbf{z}_k)$ are functions describing occupied and empty space, which will be described by equations (5.12) and (5.13). This equation have the wanted properties from equation (3.58). The inverse sensor model turns toward 0.5 as P_d turn toward zero, i.e. when the there is no distinction between a target and noise. Grid points that are outside the region of both $f_{occ}(m_i, \mathbf{z}_k)$ and $f_{emp}(m_i, \mathbf{z}_k)$ will be updated with 0.5, i.e. no added value in log-odds. The inverse sensor model will be $P(m_i|z_k) > 0.5$ when $f_{occ}(m_i, \mathbf{z}_k) > f_{emp}(m_i, \mathbf{z}_k)$ and vice versa.

The occupancy function, $f_{occ}(\cdot)$, will be grid size dependant. Since the uncertainty of the radar detections are described in range and angles, equation (5.6), the number of grid cells covered by the Gaussian sensor uncertainty will increase with range (see Figure 5.4b). The approach is to calculate the probability that the detection has fallen into a certain grid cell by computing the cumulative distributed function (cdf), based on the sensor characteristics, in the interval covered by that grid cell. Because the sensor noise is independent between r, u and v, the total cdf can be calculated by the product of three separate one dimensional cdf, seen in equation (5.12). Given the detection r_{det}, the point r_i with the interval $\pm\Delta L_i^r$ contains the probability

$$\Phi_r(r_i + \Delta L_i^r) - \Phi_r(r_i - \Delta L_i^r) = \frac{1}{2}\,\mathrm{erf}\left(\frac{r_i + \Delta L_i^r - r_{det}}{\sigma_r\sqrt{2}}\right) - \frac{1}{2}\,\mathrm{erf}\left(\frac{r_i - \Delta L_i^r - r_{det}}{\sigma_r\sqrt{2}}\right), \tag{5.10}$$

where Φ is the cdf for a normal distribution. An example of what this mean can be seen in the upper right plot in Figure 5.3 where equation (5.10) calculates the area covered by the example grid cell. This is very similar to the convolution method in [14], and it is actually the result if the ideal sensor model is changed to have zero on both sides of the peek (see Figure 3.7). Radar operates with polar coordinates,

r, u and v, making it natural to model the sensor in the polar frame. The intervals, ΔL_i^r, ΔL_i^u, and ΔL_i^v, in the cdf will be calculated in the polar frame, but should represent the same physical size (in this case the size of a grid cell) in the Cartesian frame. The intervals are calculated as

$$\begin{aligned} \Delta L_i^r &= \sqrt{2}\Delta m, \\ \Delta L_i^u &= \frac{\sqrt{2}\Delta m}{r_i}, \\ \Delta L_i^v &= \frac{\sqrt{2}\Delta m}{r_i}, \end{aligned} \tag{5.11}$$

where each interval corresponds to approximately the diagonal of the grid cells. It should be noted that $r_i \in (0, r_{max})$, where r_{max} is the maximum range of the sensor.

The final equation for both range, azimuth and elevation is given by

$$f_{occ}(m_i, \mathbf{z}_k) = \prod_{j=\{r,u,v\}} \Big(\Phi_j(j_i + \Delta L_i^j) - \Phi_j(j_i - \Delta L_i^j)\Big), \tag{5.12}$$

where cdf interval is calculated as equation (5.10) for r, u and v. In summary: this equation will calculate the probability that a measurement originates from a specific grid cell. The middle left plot in Figure 5.3 shows a one dimensional example of this function.

The function for free space, $f_{emp}(\cdot)$, should affect the grid cells between the detection and sensor. In the polar coordinates this is simply the area with r lower than r_{det}. As mentioned earlier, it is not desirable to affect the area close to a detection. A Gaussian function with a scale factor of one will be used for the range (similar approach as in [30]). The standard deviation of the exponential is set to $\frac{1}{4}$ of the distance to the detection. For the angles, the same equation as in the occupied function is used. The same uncertainty factor in angle affects both functions the same. The final function of free space is

$$f_{emp}(m_i, \mathbf{z}_k) = \Big(e^{\frac{-r_i^2}{2(r_{det}/4)^2}} \prod_{j=\{u,v\}} \Big(\Phi_j(j_i + \Delta L_i^j) - \Phi_j(j_i - \Delta L_i^j)\Big)\Big). \tag{5.13}$$

The middle right plot in Figure 5.3 shows a one dimensional example of this function and the lower left shows the final inverse sensor model. An example of the inverse sensor model from equation (5.9) for a 2D OGM is shown in Figure 5.4b.

5.2.2 Update of occupancy grid cells

For each new radar measurement, and for each detection, the OGM is updated by equation (3.59). To minimize the computational complexity not all grid cells in the OGM will be updated for each detection. This is reasonable because only grid cells around the detection, and between the sensor and the detections, will gain any new values. This section will explain the mapping in equation (5.8), and how to filter out unnecessary grid points to save computations.

The most computationally expensive part of the OGM update is computing the error functions. Instead of doing it for all grids that overlap with the sensor field of view, the idea is to filter out which grids that the inverse sensor model affect. This is a lot easier to do in the polar domain than in the Cartesian, where the grids can be screened by a cuboid instead of a cone. Firstly, a cuboid is created in the Cartesian frame which contain the radar field of view plus some margin. An illustration of this in 2D is shown in Figure 5.4a. This cuboid contain excessive amounts of grids, but they will be filtered out in the polar domain. These grids are then converted to the polar domain by combining equation (5.2) and (5.4) and solving for u_{det}, and v_{det}, knowing that r_{det} is the Euclidean distance between the sensor and the detection. Because we transform grid cells location and not detections, the subscript *det* is changed to i,

$$\begin{aligned} r_i &= \sqrt{(X_i - X_{radar})^2 + (Y_i - Y_{radar})^2 + (Z_i - Z_{radar})^2}, \\ \begin{bmatrix} \sim \\ u_i \\ v_i \end{bmatrix} &= \frac{1}{r_i} R^T(\psi, \theta, \phi) \begin{bmatrix} X_i - X_{radar} \\ Y_i - Y_{radar} \\ Z_i - Z_{radar} \end{bmatrix}, \end{aligned} \tag{5.14}$$

where

$$\begin{bmatrix} X_{radar} \\ Y_{radar} \\ Z_{radar} \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + R(\psi, \theta, \phi) \begin{bmatrix} x_{radar} \\ y_{radar} \\ z_{radar} \end{bmatrix}. \tag{5.15}$$

The transformation of grid cells between the Cartesian frame and the polar frame is seen between Figure 5.4a and 5.4b.

The radar noise is modelled as a Gaussian, thus containing values from $\pm\infty$ in all dimensions. To avoid unnecessary computations, only the grid cells within the third standard deviation region, $0 < r_i \leq (r_{det} + 3\sigma_r)$, $(u_{det} - 3\sigma_u) \leq u_i \leq (u_{det} + 3\sigma_u)$ and $(v_{det} - 3\sigma_v) \leq v_i \leq (v_{det} + 3\sigma_v)$, are calculated with the inverse sensor model in equation (5.9). The third sigma region contains almost all information in a Gaussian, (99.73%), making the loss negligible. The yellow region in Figure 5.4b shows this area in a 2D inverse sensor model. The map is then updated according to equation (3.59). The process from an initial OGM to an updated OGM after an radar detection is described in Figure 5.3 and Figure 5.4.

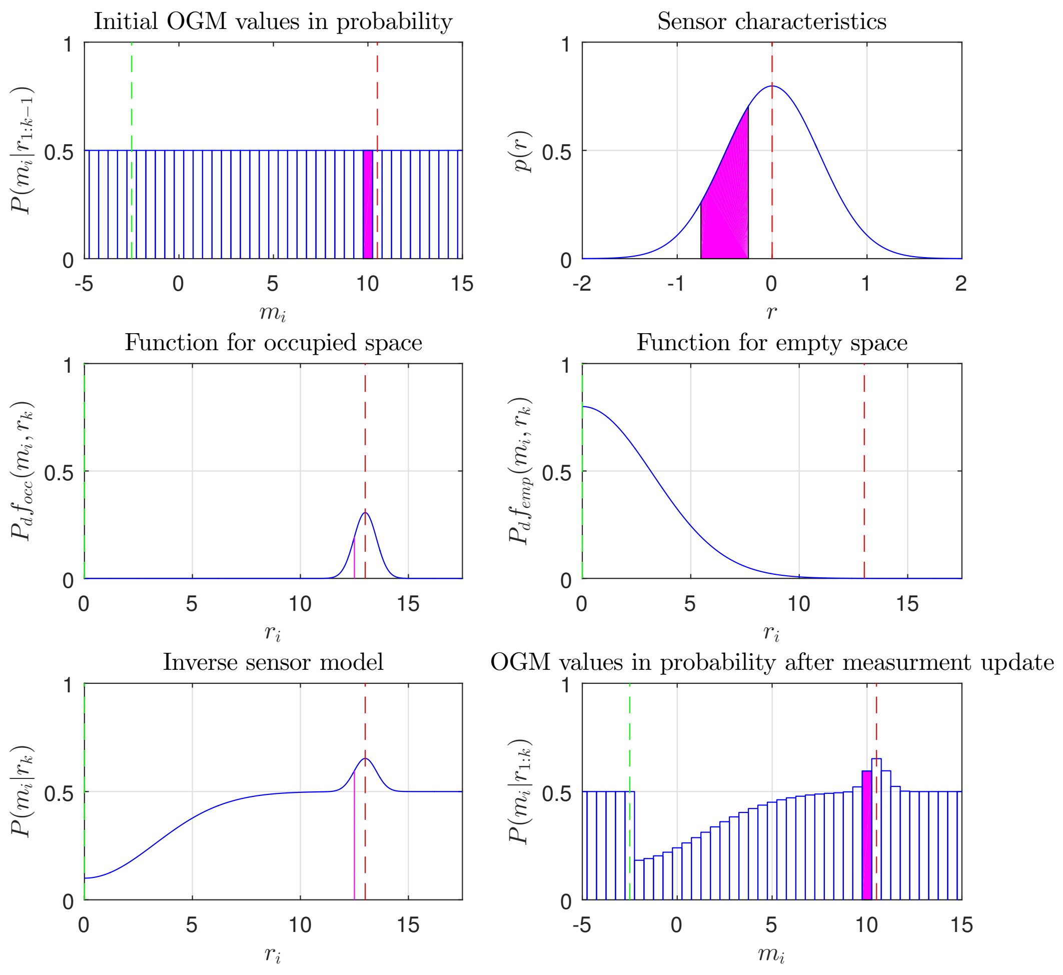
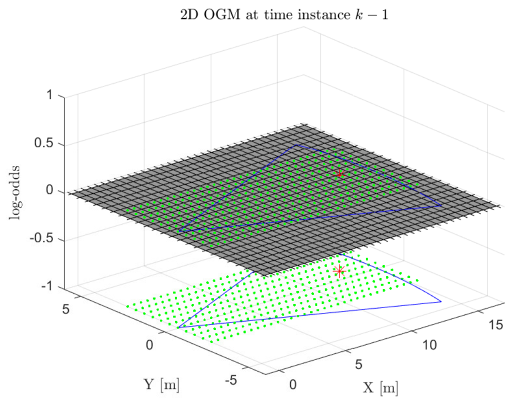
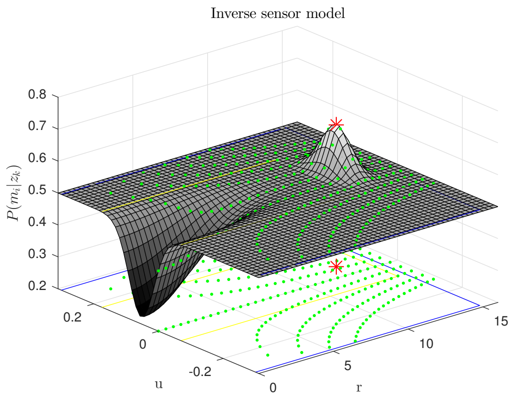


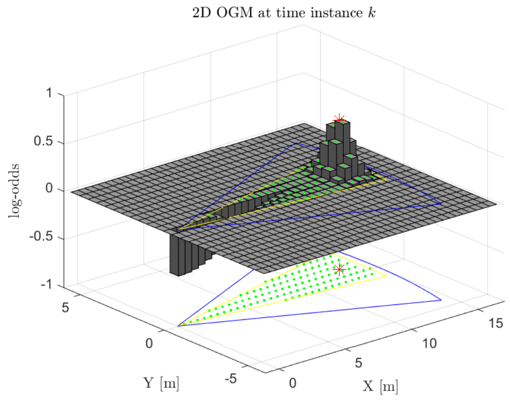
Figure 5.3: A sequence showing the basics of our approach to OGM mapping using an inverse sensor model for a one dimensional OGM. The plots are referred to as upper left (UL), middle right (MR), lower left (LL) etc. The green dashed line represent the sensor position, and the red dashed line is a range measurement which has 80% probability of detection. The magenta colored area/line represent a specific grid point, used as a reference which can be followed between the images. UL represent the OGM at time instance $k-1$. UR shows the range characteristics of the sensor, and the occupied function in ML is calculated as the area in an interval of this function. ML and MR show the two functions that represent occupied and empty space, which result in the the inverse sensor model shown in LL. LR shows the resulting OGM after the log odds update at time instance k.



(a) Initial 2D OGM before it has been updated with a radar detection.



(b) The inverse sensor model for a radar detection.



(c) 2D OGM after it has been updated with a radar detection.

Figure 5.4: A sequence showing the basics of our approach to occupancy grid mapping, using the inverse sensor model for a two dimensional OGM. The case is similar to the one dimensional case illustrated in Figure 5.3, but here we can also illustrate angular uncertainty. The initial OGM is shown in Figure 5.4a with no initial information regarding the map, i.e all grid cells have log-odds ratio of 0. The blue area describes the field of view, the red cross the radar detection, and the green dots represents grid cells that are transformed to polar coordinates for the inverse sensor model. The inverse sensor model is shown in Figure 5.4b, and the green dots represent the grid cells that has been transformed. Only the grid cells inside the yellow area are calculated with the inverse sensor model. The third illustration, Figure 5.4c, shows the updated OGM for the given radar measurement. The green grid cells were the grid cells that were updated by the inverse sensor model.

5.3 Likelihood from occupancy grid map

The end goal of the map is to be able to utilize it for the localization of the vehicle. If we can utilize the map to create a likelihood for the detections, conditioned on the vehicle pose and the map, it can easily be incorporated in the Bayesian filtering framework. The goal is to calculate the likelihood

$$\mathcal{L}(\mathbf{m}, \mathbf{x}|\mathbf{z}). \tag{5.16}$$

Earlier, in the mapping theory, we omitted the states, $\mathbf{x}$, because they are deterministically known, but we include the term again to clearly show that the likelihood is state dependent. The likelihood $\mathcal{L}(\mathbf{m}, \mathbf{x}|\mathbf{z})$ considers only one detection, but when more detections are present, which is almost always the case, the different detections have to be combined to form the likelihood. The likelihood for several detections is

$$\mathcal{L}(\mathbf{m}, \mathbf{x}|\mathbf{z}) = p(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N|\mathbf{m}, \mathbf{x}). \tag{5.17}$$

This can be complicated to compute, given that we have no forward model, instead we consider the detections independent from each other and we can form the likelihood as

$$\mathcal{L}(\mathbf{m}, \mathbf{x}|\mathbf{z}) = \prod_i^N p(\mathbf{z}_i|\mathbf{m}, \mathbf{x}) = \prod_i^N \mathcal{L}(\mathbf{m}, \mathbf{x}|\mathbf{z}_i). \tag{5.18}$$

We now want to calculate the terms $\mathcal{L}(\mathbf{m}, \mathbf{x}|\mathbf{z}_i)$.

As described in Section 5.2, we use the inverse sensor model to create the OGM. We will utilize that the sensor noise is incorporated in the map to directly use the OGM as a heuristic likelihood function. What this means is that the likelihood for a radar detection can directly be computed from the point in the map where the detection is located (see Section 5.1), without considering the sensor noise. Since the map is discrete, there will be a problem when retrieving the grid values for detections. Interpolation would compute a good representation of the point, but binning can also be sufficient. In the gradient search approach, tricubic interpolation is utilized whilst the particle filter approach uses trilinear interpolation. We will further describe these procedures in their respective sections, 6.1 and 6.2. The values of the extracted grids are converted back to probability from log-odds by equation (3.52).

Besides the map, we also want to incorporate the detection probability, P_d, when forming the likelihood. P_d should have the same impact as when it was used to create the map. For a grid cell in the map with probability of occupancy $P(m_i)$, and a detection in that cell with detection probability P_d, the likelihood should have the following characteristics

- High $P(m_i)$ and high P_d should yield a high $\mathcal{L}(\mathbf{m}, \mathbf{x}|\mathbf{z}_i)$.
- High $P(m_i)$ and low P_d should yield an uncertain $\mathcal{L}(\mathbf{m}, \mathbf{x}|\mathbf{z}_i)$.
- Low $P(m_i)$ and high P_d should yield a low $\mathcal{L}(\mathbf{m}, \mathbf{x}|\mathbf{z}_i)$.

- Low $P(m_i)$ and low P_d should yield an uncertain $\mathcal{L}(\mathbf{m}, \mathbf{x}|\mathbf{z}_i)$.

Since the likelihood will be formed as a probability, an uncertain likelihood correspond to a probability of 0.5. Equation (5.9), which creates the inverse sensor model, have the given properties and can be used again to create the likelihood for one detection. By replacing the occupancy function with the probability of occupancy for the grid cell, $f_{occ} \rightarrow P(m_i)$, and the function of empty space to the probability of empty space for the grid cell, $f_{emp} \rightarrow (1 - P(m_i))$, we retrieve the desired equation

$$\begin{aligned} \mathcal{L}(\mathbf{m}, \mathbf{x}|\mathbf{z}_i) &= \frac{1}{2}(1 + P_d P(m_i) - P_d(1 - P(m_i)) \\ &= \frac{1}{2} + P_d(P(m_i) - \frac{1}{2}). \end{aligned} \tag{5.19}$$

To give an example of how equation (5.19) works, Table 5.1 below shows the output P_d given combinations of high and low SNR and map values. The equation outputs the desired performance. The final likelihood is formed by calculating equation (5.19) for each detection and combining them with equation (5.18).

$P_d \backslash P(m_i)$	0.2	0.9
0.2	0.44	0.58
0.9	0.23	0.86

Table 5.1: The table present examples of likelihoods, $\mathcal{L}(\mathbf{m}, \mathbf{x}|\mathbf{z}_i)$, computed by equation (5.19), for combinations of P_d and $P(m_i)$. The output coincide with the desired attributes of the equation.

6

Localization filter designs

We employ two different approaches to solve the localization problem, where the impementations in chapter 4 and 5 are equal for both cases. The first approach, 6.1, use registration to find position and orientation measurements for an EKF. The second implementation, 6.2, use a Rao-Blackwellized particle filter, which combines a particle filter with Kalman filters, to solve the localization. Both approaches use EKF, but there exist other filters capable of dealing with non-linear models, e.g. sigma point filters. They were discarded when compared to the EKF, because the state vector is quite large (and they scale badly with cardinality) and there are mild non-linearities in the models which imply simple linearizations.

6.1 Ego-position estimation based on registration

In this approach, we use an EKF to solve the state estimation completely. Signals from the IMU are utilized to update the velocity, acceleration and angular rates. A second measurement update is performed where the measurement is the pose estimate from registration between the OGM and radar measurements. The registration is solved by a gradient search algorithm that tries to find the pose that maximizes a local likelihood,

$$\arg\max_{\mathbf{x}_k} \mathcal{L}(\mathbf{m}, \mathbf{x}_k | \mathbf{z}_k) \quad \forall k \in 1, \ldots, K. \tag{6.1}$$

The registration algorithm is thus a function of the pose variables: X, Y, Z, yaw, pitch and roll and the OGM. When the best pose of the aforementioned variables is found, it is used as a measurement to the EKF by

$$\mathbf{z}_k^{MAP} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{H_k^{MAP}} \begin{bmatrix} X \\ Y \\ Z \\ \psi \\ \theta \\ \phi \end{bmatrix}. \tag{6.2}$$

A multiple model approach is chosen to evaluate the input from the gradient search, which aims to determine the uncertainty of the measurement depending on weights based on likelihoods. To easier understand how the filter is assembled, Figure 6.1 visualize the filter components and how they are connected.

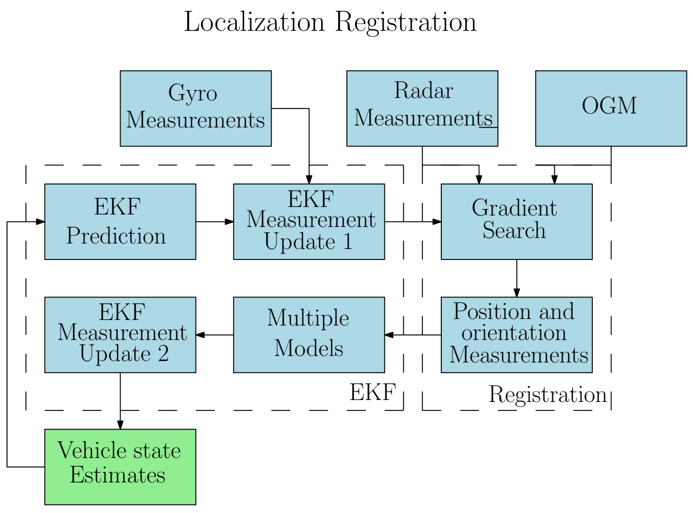


Figure 6.1: The image display the interconnections and structure of the gradient search approach components.

6.1.1 Likelihood maximization by gradient search

We use gradient search, see Section 3.5.2, to solve the registration. For each cycle of available radar data, the algorithm tries to reach a local optima for the ego pose by maximizing the likelihood for the detections in the OGM. To expedite the rate of convergence, an adaptive step length is employed, which starts large and decreases if no increase in function value is found. This is also known as line search. This approach helps a lot near narrow edges and when moving on top of a steep ascent. A set max number of iterations is used to make the algorithm tractable. In situations when the gradient is very low but not low enough, the progression will decrease severely, see equation (3.63). To solve this, the gradient is normalized.

Gradient search has a major flaw when combined with an OGM: it is only valid for continuous variables, and the grid is discrete. For any continuous point, the likelihood and the corresponding derivatives have to be estimated. They use difference approximation in [23] to numerically find a local derivative of the likelihood, but how the detections are matched to the grids is not mentioned. We solve the problem by interpolation, which makes it possible to get an continuous estimate of the likelihood and also an analytic derivative. Figure 6.2 highlights the issue of forming a likelihood from a 2D discrete grid without any smoothing. The red crosses are located in the middle of each grid.

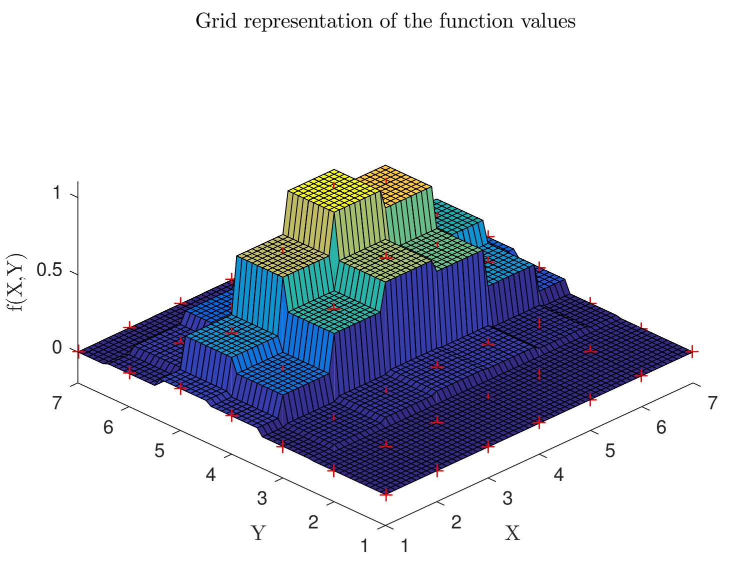


Figure 6.2: Illustration of a 2D likelihood formed by a grid map without interpolation. The red crosses correspond to the center of a grid.

6.1.2 Tricubic interpolation of occupancy grid map

To circumvent the issue of a discrete map, the idea is to smooth the map by cubic interpolation and retrieve a continuous function around the detections. In a three dimensional case, interpolation creates a continuous function between the eight function values, f_{xyz}, that surrounds a point. In our case, these correspond to likelihoods from equation (5.19). Linear interpolation would produce a continuous likelihood, but cubic interpolation utilize several orders of derivatives, and cross-derivatives, to create a function that also has a continuous gradient. This is favourable when using gradient search. A continuous derivative is achieved by conditioning that the gradients have to be the same when approaching a point from two adjacent grids. As we want to use the gradients in the search algorithm, and MATLAB's own cubic interpolation does not output them, we form our own implementation and extract them so that they can be utilized. The tricubic interpolation for detection point X_{det}, Y_{det} and Z_{det} is computed by

$$f(X_{det}, Y_{det}, Z_{det}) = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} a_{ijk}\tilde{X}_{det}^{i}\tilde{Y}_{det}^{j}\tilde{Z}_{det}^{k}, \tag{6.3}$$

where $\tilde{X}$, $\tilde{Y}$, and $\tilde{Z}$ represent the relative value between the detection and the grid points and grid size [18]. Note that $f(X_{det}, Y_{det}, Z_{det})$ is the same as the likelihood for a detection described in equation (5.19). The parameter a is a vector of cardinality 64, given by

$$a = B^{-1}b, \tag{6.4}$$

where

$$b = \left[f_{xyz}, \frac{\partial f_{xyz}}{\partial X}, \frac{\partial f_{xyz}}{\partial Y}, \frac{\partial f_{xyz}}{\partial Z}, \frac{\partial^2 f_{xyz}}{\partial X \partial Y}, \frac{\partial^2 f_{xyz}}{\partial X \partial Z}, \frac{\partial^2 f_{xyz}}{\partial Y \partial Z}, \frac{\partial^3 f_{xyz}}{\partial X \partial Y \partial Z} \right]^T . \tag{6.5}$$

The vector b has $8 \times 8 = 64$ coefficients, containing the values and several orders of derivatives of the eight corner grids. B^{-1} is a 64×64 matrix containing the coefficients of the equation between b and a, which is based on the cubic Hermite spline. The value of f_{xyz} in b is given directly by the calculating the likelihood from the eight surrounding grid points in the map. To get a continuous derivative between grid points, calculating all derivatives in equation (6.5) with finite difference ensures that the derivatives are equal between two adjacent grids. An example of finite differences is calculated by

$$\frac{\partial f_{x,y,z}}{\partial X} = \frac{f_{x+1,y,z} - f_{x-1,y,z}}{2}. \tag{6.6}$$

A useful aspect of equations (6.4) and (6.5) is that the coefficients of a don't have to be recalculated for a detection as long as the detection stays in the same grid during the gradient search. The 2D grid in Figure 6.2 is interpolated by both linear and cubic interpolation, which are shown in Figure 6.3 and Figure 6.4 in order to showcase the difference between the smoothness. It is apparent that the cubic interpolation smooth a lot better than its linear counterpart.

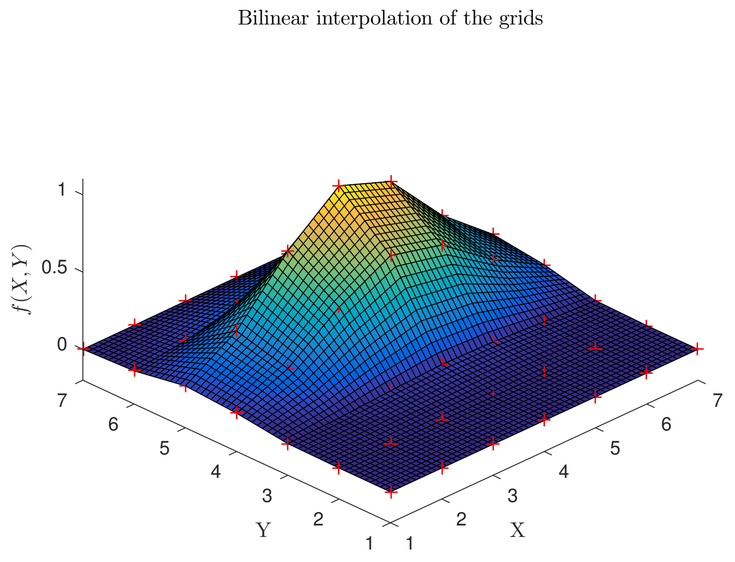


Figure 6.3: Bilinear interpolation of the 2D likelihood in Figure 6.2. The red crosses correspond to the center of a grid.

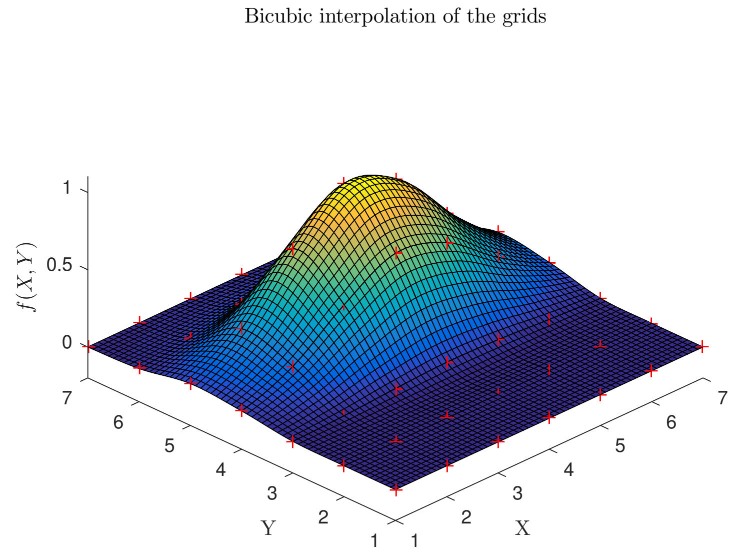


Figure 6.4: Bicubic interpolation of the 2D likelihood in Figure 6.2. The red crosses correspond to the center of a grid.

So far the function value, $f(X_{det}, Y_{det}, Z_{det})$, has been calculated, and the next step is to derive the gradient. A nice aspect of cubic interpolation is that the partial derivatives of equation (6.3) can be calculated analytically and is continuous between grids (if implemented as mentioned). Since the goal is to find the local pose that maximize the likelihood of the radar detections, the detections should be derived with respect to the pose states: X, Y, Z, yaw, pitch and roll. Equation (5.4) showed the relation between the detections and the pose states. Let s be an arbitrary pose state, then the partial derivative of the tricubic interpolation based on that state is

$$\frac{\partial f(X_{det}, Y_{det}, Z_{det})}{\partial s} = \\ \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} a_{ijk}\Big(i\frac{\partial \tilde{X}_{det}}{\partial s}\tilde{X}_{det}^{i-1}\tilde{Y}_{det}^{j}\tilde{Z}_{det}^{k} + j\frac{\partial \tilde{Y}_{det}}{\partial s}\tilde{X}_{det}^{i}\tilde{Y}_{det}^{j-1}\tilde{Z}_{det}^{k} + k\frac{\partial \tilde{Z}_{det}}{\partial s}\tilde{X}_{det}^{i}\tilde{Y}_{det}^{j}\tilde{Z}_{det}^{k-1}\Big), \tag{6.7}$$

where the argument s is droped from $X_{det}(s)$ etc. for simplicity. In Figure 6.5 and 6.6, the partial derivatives of the bicubic interpolation in Figure 6.4 are shown.

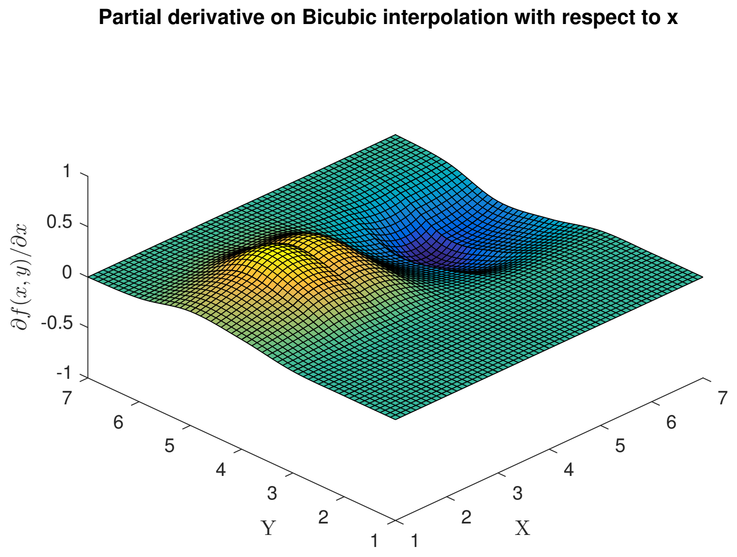


Figure 6.5: Partial derivative of the bicubic interpolation in Figure 6.4 with respect to x. As can bee seen the derivative is continuous but not smooth between the grids.

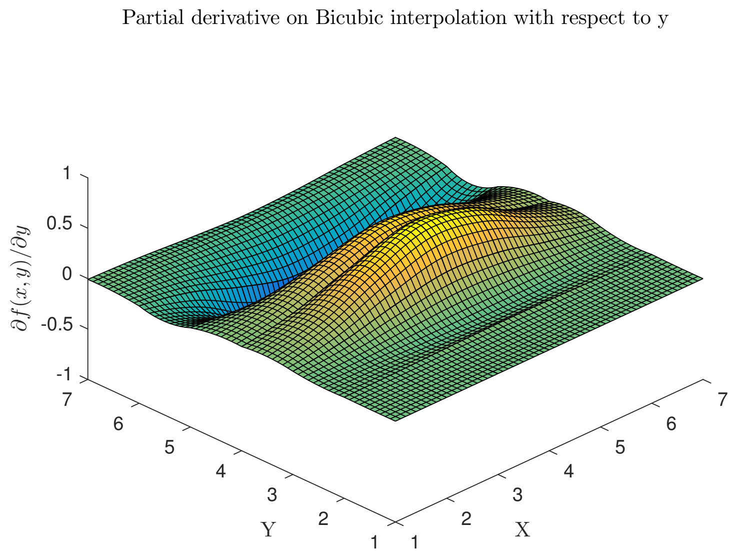


Figure 6.6: Partial derivative of the bicubic interpolation in Figure 6.4 with respect to y. As can bee seen the derivative is continuous but not smooth between the grids.

During the gradient search, equation (6.3) and (6.7) are calculated for each detection. As mentioned, the function is the same as the likelihood for a specific detections $f(X_{det,i}, Y_{det,i}, Z_{det,i}) = \mathcal{L}(\mathbf{m}, \mathbf{x}|\mathbf{z}_i)$. As was described in equation (5.18) the final likelihood is formed by the product of each detections likelihood. The derivative of the final likelihood is then calculated by

$$\frac{\partial \mathcal{L}(\mathbf{m}, \mathbf{x}|\mathbf{z}_i)}{\partial s} = \sum_{j=1}^{N} \Big(\frac{\partial f_j(X_{det}, Y_{det}, Z_{det})}{\partial s} \prod_{k \neq j} (f_k(X_{det}, Y_{det}, Z_{det})) \Big). \tag{6.8}$$

s is as before an arbitrary pose state and all the terms on the right hand side is calculated from equation (6.3) and (6.7). The product term $\prod_{k \neq j}$ is the product between all function values from 1 to N, except $f_j(X_{det}, Y_{det}, Z_{det})$.

6.1.3 Multiple model for weighting of the gradient search input to EKF

When dealing with optimization problems, not all solutions are global optima. Depending on non-linearities and constraints, several local optima might obstruct an algorithm in the search for the global solution. Gradient search is prone to this, and will get stuck in a local optima if it is reached. The likelihood is dependant on the estimated pose, and different combinations of position and rotation can create several local optima in the likelihood, which means that a slight offset can force the gradient search away from the global optima. By our approach when forming the map, and the fact that we have good knowledge of initial position and a well performing vehicle model, we know that if the gradient search moves the position long distances, an incorrect optimum has been found. To handle this problem, we employ a multiple model technique where two models, with different covariance matrices, are weighted against each other to form the posterior distribution. The purpose of the multiple models is to provide information regarding the uncertainty from the innovation in the EKF. The purpose implies that the multiple models should have different likelihoods, hence the predictive models can be the same. To avoid unnecessary computation, a joint prediction is utilized. This implies a joint posterior distribution, which aligns well with our approach. In fact, it fits very well as the point is to estimate one posterior. The weights are formed by computing two different likelihoods, which are compared against each other. The weights are then used to compute the posterior according to equation (3.28).

When working with multiple models, the weights are usually inherited over time. This way, the choice of model is dependant on previous data and is more resilient to switching. We do not desire this behaviour, thus the weights in each iteration are independent of previous weights. This stems from how and why the multiple models are used, namely to evaluate radar detections to a map. As radar detections are volatile, we want to be able to switch between the two models immediately without any delay from previous weights. Hence the weights are computed as

$$w^j_{k|k} = \frac{L^j_k}{\sum_{i=1}^{2} L^i_k} \quad j = 1, 2, \tag{6.9}$$

where L^j_k is the likelihood of mode j at time instance k. Desirably, the model with higher uncertainty will receive a higher weight whenever the gradient search has moved a longer distance. This means that the posterior distribution will rely more on the prediction and internal sensor measurement update compared to the gradient search. For a better understanding of what is happening with multiple likelihoods, we present a one-dimensional case in Figure 6.7. Here, both likelihoods are centered around a measurement, and the blue likelihood has a lower uncertainty whilst the green has a high uncertainty. By inserting the predicted measurement, the likelihoods are evaluated and used in equation (6.9).

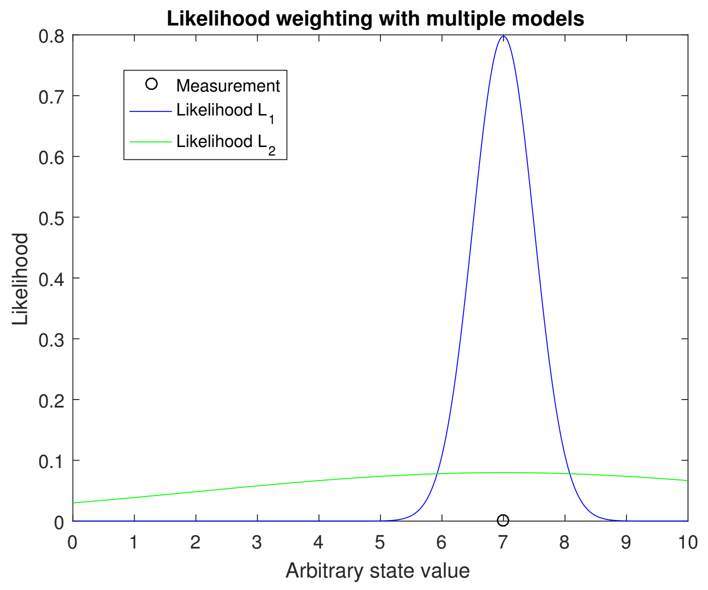


Figure 6.7: Illustration of two different likelihoods, one with low covariance (blue) and one with higher covariance (green), both centered around an arbitrary measurement (black dot). Depending on the predicted measurement value, values are extracted from both likelihoods and used in equation (6.9).

To conclude, the point of the multiple models is to evaluate the input of the gradient search to the EKF. The goal is to compute the posterior, equation (3.28), and the weights decide how much each model's parameters are used in the computation. For example, a high weight for the model with low uncertainty means that the algorithm trust the information from the gradient search. This should correspond to the gradient search finding the correct local optima. We utilize a joint prediction since the purpose of the multiple models is to evaluate the measurement models and not to evaluate the posteriors.

6.2 Ego-position estimation based on particle filter

There exist more ways than registration to solve the problem of utilizing a map for localization. An idea might be to just put out an enormous amount of possible car positions in the OGM and see which ones that can match the map the best, given radar measurements. With a large map this would of course be incredibly computationally heavy, but there actually exist a filtering technique which does this in a smart way: particle filters (Section 3.2.4). But the standard particle filter becomes intractable with a large state vector, and when modeling vehicle kinematics in 3D, the state vector becomes large and the approach is not desirable. Instead, the Rao-Blackwellized particle filter is considered (Section 3.2.5), and the amount of particles can drastically be decreased. Figure 6.8 present an overview of how the Rao-Blackwellized particle filter estimates the posterior in an iterative manner.

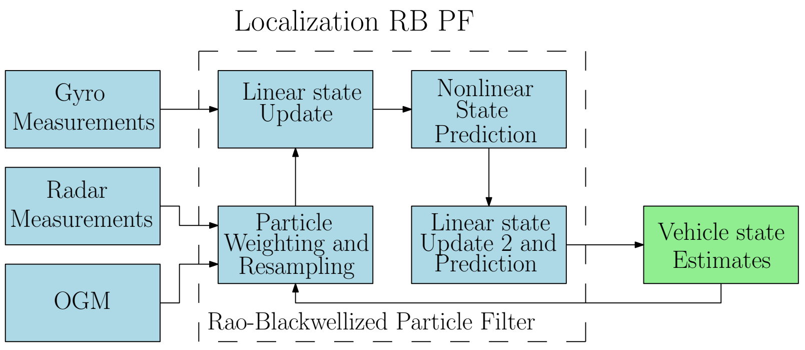

Localization RB PF
Gyro Measurements
Radar Measurements
OGM
Linear state Update
Nonlinear State Prediction
Particle Weighting and Resampling
Linear state Update 2 and Prediction
Vehicle state Estimates
Rao-Blackwellized Particle Filter

Figure 6.8: The figure describe the interconnection between the subsystems, and how they together form the filter.

The implementation requires a partitioning of the states, according to equation (3.36), which we base on the degree of non-linearity. As the velocity and acceleration enter linearly, they can be optimally estimated by Kalman filters. The angular rates are not linear in the measurement update, but this can still be approximated by an EKF. The position och orientation states will be estimated with the likelihood described in equation (5.18) and (5.19), based on radar measurements, the state vector and the OGM. This function is highly nonlinear, and the position and orientation states will thus be estimated by the particles. The partitioning gives us the state vectors

$$\mathbf{x}_k^n = \begin{bmatrix} X_k \\ Y_k \\ Z_k \\ \psi_k \\ \theta_k \\ \phi_k \end{bmatrix} \qquad \mathbf{x}_k^l = \begin{bmatrix} \dot{\psi}_k \\ \dot{\theta}_k \\ \dot{\phi}_k \\ v_k \\ a_k \end{bmatrix}, \tag{6.10}$$

where n represent the states solved by the particle filter and l represent the states solved by (Extended) Kalman filters. We implement the filtering according to Algorithm 1 in Section 3.2.5, but there are two modifications which alter the equations in the approach. These two changes are explained in Section 6.2.1. As we describe in Section 3.2.5, the motion models in equations (4.2) and (4.3), and measurement models in equations (4.7) and (4.8) have to be rewritten as

$$
\begin{aligned}
f_k^n(\mathbf{x}_k^n) &= \begin{bmatrix} X_k \\ Y_k \\ Z_k \\ \psi_k \\ \theta_k \\ \phi_k \end{bmatrix}, &
\mathbf{A}_k^n(\mathbf{x}_k^n) &= \begin{bmatrix} 0 & 0 & 0 & T\cos\theta\cos\psi & 0 \\ 0 & 0 & 0 & T\cos\theta\sin\psi & 0 \\ 0 & 0 & 0 & -T\sin\theta & 0 \\ T & 0 & 0 & 0 & 0 \\ 0 & T & 0 & 0 & 0 \\ 0 & 0 & T & 0 & 0 \end{bmatrix}, \\
\mathbf{A}_k^l(\mathbf{x}_k^n) &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, &
\mathbf{H}_k(\mathbf{x}_k^n) &= \begin{bmatrix} -\sin\theta & 0 & 1 & 0 & 0 \\ \cos\theta\sin\phi & \cos\phi & 0 & 0 & 0 \\ \cos\theta\cos\phi & -\sin\phi & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned}
\tag{6.11}
$$

These are the matrices required in equation (3.37) where the left out matrices are zero.

In a standard particle filter, the particles propagate according only to the motion model, but in the Rao-Blackwellized method the particles propagate after the measurement update of the linear states. Thus, the prediction incorporate more information and can follow the posterior better. The particle filter can solve the ego-positioning by comparing radar measurements and the map for each particle. For each time step, each particle project the detections to the map and retrieve a likelihood value based on equation (5.18). The likelihood updates the particles weights according to equation (3.35). Particles that get low weights have probably diverged from the true posterior and will be removed by re-sampling. In this thesis the states of the vehicle will be estimated as the weighted mean of the particles, i.e.

$$
\mathbb{E}[p(\mathbf{x}_k|\mathbf{Z}_{1:k})] = \sum_{i=1}^{N} \mathbf{x}_k^{(i)} w_k^{(i)}. \tag{6.12}
$$

where the weights are already normalized. This is the estimation we use when comparing to ground truth, when the filter performance is evaluated. This is representable enough as long as the posterior is not multimodal. An example of the posterior from real data is shown in Figure 6.9.

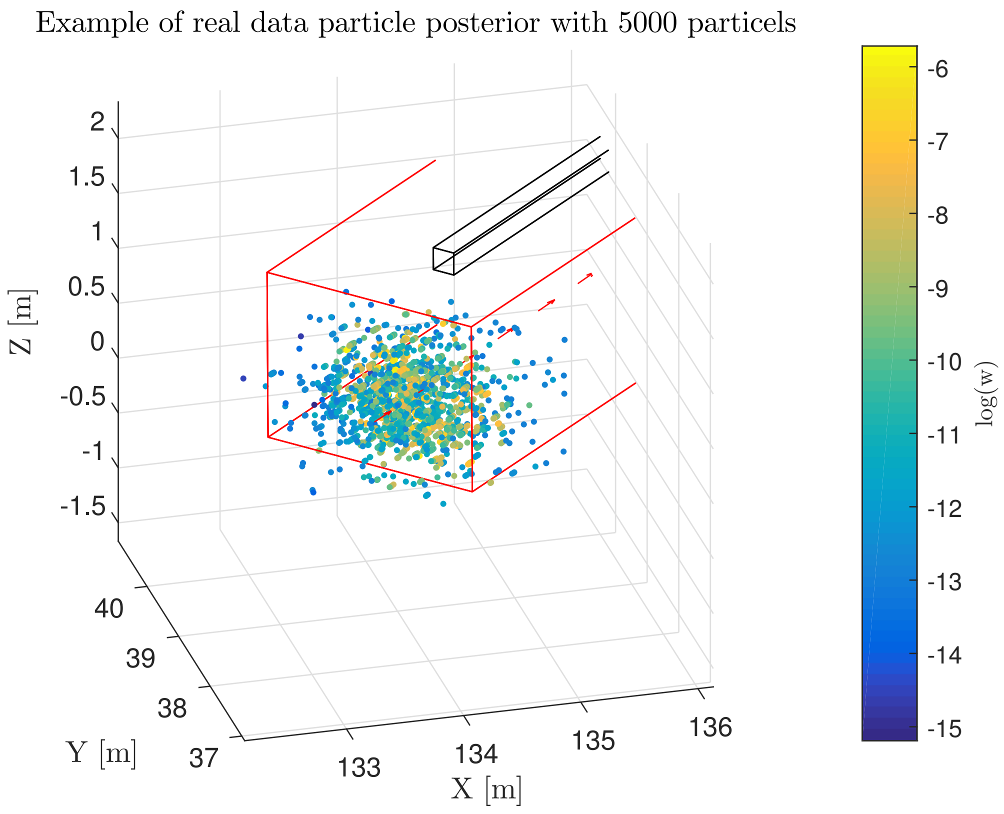


Figure 6.9: The MATLAB plot shows a distribution of 5000 particles around the rear axis of the vehicle, in one time instance. The red box represent the car, and the black box is the conduit on which the radar is mounted. For a better illustration, the particles have only propagated a few cycles from the initial prior and has, thus, a bigger spread than after converging. Particles in the center have, in general, higher weight where particles with lower weight near the center have bad orientation estimates. 3D shapes are hard to evaluate in an image, but it shows that the distribution is not multimodal.

6.2.1 Modifications of Rao-Blackwellized

The modifications originate from an idea to reduce the computational complexity. Instead of assigning one Kalman filter to each particle, the idea is to use the same Kalman gain, innovation covariance and posterior covariance matrix for all particles, since they are the most computationally demanding parameters in the Kalman filter. This simplification can only be motivated if the elements in the states that affects the mentioned matrices are not spread out. Important to note is that each particle still has its own state vector and computed innovation. To estimate the aforementioned matrices, a mean of all particles has to be estimated. With the same approach as for evaluating the posterior, a weighted mean is computed for all of the states, with which we form one of each matrix required in equation (3.40). This simplification is applicable in the first measurement update in Algorithm 1, point 5, that is when the velocity measurements from the IMU updates the particles' velocities. The Jacobian of the measurement model in equation (6.11), $\mathbf{H}_k(\mathbf{x}_k^n)$, depends only on pitch and roll, which are assumed to have a low spread between the particles. This is realistic due to kinematic constraints on the car, enforced by the road.

For the time update in Algorithm 1, point 7, the proposed simplification is not valid. The motion model in equation (6.11), $\mathbf{A}_k^n(\mathbf{x}_k^n)$, also depends on yaw, which is harder to assume to have low spread between the particles. A single Kalman gain, innovation covariance and posterior covariance matrix in equation (3.40) can't be computed for all particles. The way we solve this problem is by saying that we have no noise on the nonlinear states, $\mathbf{Q}_k^n = 0$. This can be motivated by the fact that no noise enters directly on the pose states but enters through the velocities. Thus each particle has exact knowledge of its own position and orientation. With this assumption, we can make a huge simplification to equation (3.40), which we derive in Appendix A. The result of the derivation is

$$\hat{\mathbf{x}}_{k+1|k}^l = \mathbf{A}_k^l \mathbf{x}_{k|k}^l, \tag{6.13}$$

$$\mathbf{P}_{k+1|k} = \mathbf{Q}_k^l, \tag{6.14}$$

i.e. the predicted linear states are the same as the true propagated linear state values times the linear mapping matrix, and covariance of the next step is solely based on the process noise for the linear states. This is also intuitive, if the position of the particles are known prior and after the propagation, we can exactly tell how much velocity that correspond to (as the time interval T is known). Since the velocities used in the propagation of the particles is known, this doesn't even have to be calculated. The noise is also intuitive; if the velocity at time instance k was known, the uncertainty of the predicted velocity is the same as the uncertainty of the motion model.

7
Results

This chapter presents the results of the thesis, evaluating both the map and the two localization algorithms. Firstly, the map is presented and evaluated. Secondly, we present the results from the simulation environment, where localization performance of an EKF with only the IMU is shown before presenting the results of the two proposed solutions with map utilization. This is because we want to show how much of the drift, originating from cumulative errors, that can be removed by including the radar and the OGM. Then, with the same outline as the simulation environment, we present the results based on real data. The Rao-Blackwellized modification is then compared with the standard Rao-Blackwellized implementation in order to evaluate any changes in performance. Lastly, the multiple models for the gradient search approach are also evaluated, and we investigate if it behaves like intended.

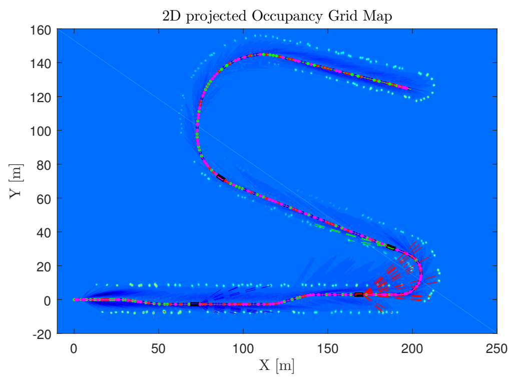
The important metrics for localization are the positions, X, Y and Z and the orientation ψ, θ and ϕ. By knowing these estimates, the position for any point in the car can be computed and not just the center of the rear axis. However, the results focus only on the estimation of said position. As we do not have access to a reference map, a mathematical evaluation of the map is not possible.

The simulations are constructed of iterations of data, and the estimates are plotted over what we refer to as a cycle. Each cycle represent the time instance when the radar has made a measurement, which has an update rate of 12.5 Hz in the guard rail scenario. The simulation environment cycles are made so that they correspond to the update rate of the real radar.

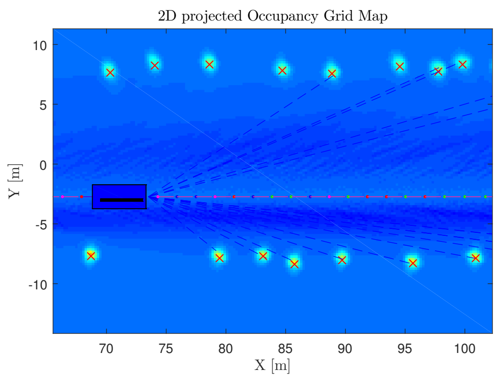
All three filters use the same motion model, see Section 4.1, but the simulation environment does not include acceleration measurements, and the aforementioned state is removed from the state estimation. This is inconvenient, but it should not affect the results too much. We use the same process noise in both simulation and on real data, except that the motion model does not contain acceleration in simulation, but there is noise on the velocity state as well as on the acceleration. We base the measurement noise covariance on evaluating the measurements from real data, and use them both on real data and in the simulation environment.

7.1 Map evaluation

Figure 7.1a shows the OGM created by four different runs of the simulation, with varying uncertainty of the scatterers. By using several runs, we recursively build up the map, but are also able to capture the effect of probability of detection, P_d. The inverse sensor model is successful in creating the map around the true position of the scatterers (marked with red crosses), which can be seen in Figure 7.1b.



(a) A 2D overview of the OGM, where the Z-axis is projected onto the X and Y plane.



(b) A highlight of how the inverse sensor model is successful in creating a map around the true positions of the scatterers, which are marked by red crosses.

Figure 7.1: The occupancy grid map created from the simulation data. Lower log-odds values are darker while higher log-odds values are brighter. The effect of f_{emp} can clearly be seen.

An OGM was also created with the test scenario, described in Section 2.4. A 2D projected version of the OGM is shown in Figure 7.2. We utilize the overview from Google map, Figure 2.4, and the scenario picture, Figure 2.3, for evaluation. To visualize how the created OGM coincide with the real world, the map and the overview picture are aligned, which is shown in Figure 7.3. We can see from the result that we are able to distinguish the poles in the guard rail at the side of the road, and also the walls of the small houses behind the guard rail. At the beginning of the test case there are no objects that can provide reliable detections, and the map is indistinct. Different illustrations of the full 3D map can be found in Appendix B.

As described in Section 2.4, in some of the tests the car overtakes another vehicle, but they are not shown in the presented map. This is because the detections from the other car is screened by equation (5.1). This shows one of the strengths of radar sensors.

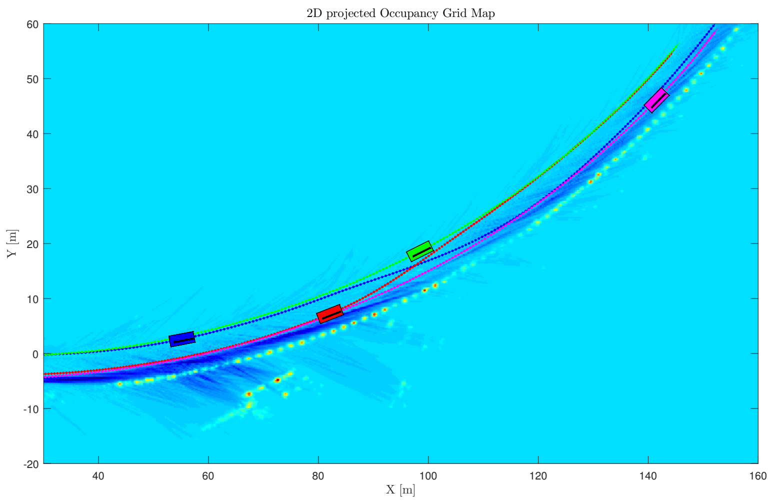


Figure 7.2: 2D projected OGM created from the test case by four drives. Cyan areas describe free space, areas with color towards red are occupied areas and areas with darker blue represent free space.

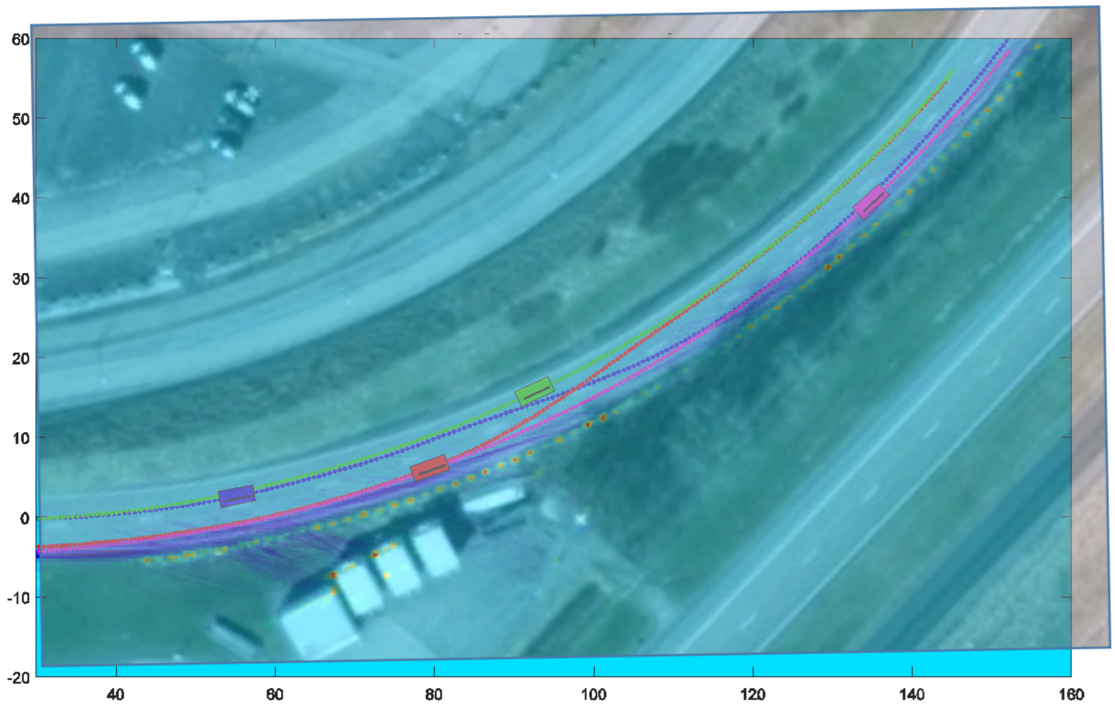


Figure 7.3: The 2D projected OGM from Figure 7.2 and the real test scenario from Figure 2.4 are aligned, and the OGM is set transparent. The alignment isn't perfect, but the dots of high probability of occupancy in the OGM correspond to the poles in the guardrail and the corners of the small houses.

7.2 Simulation environment performance

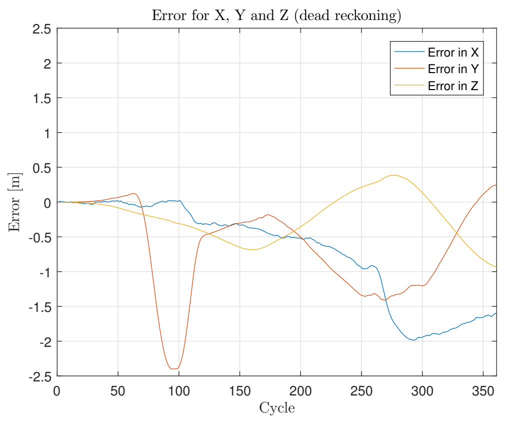
There are five different runs, and we form the map with four of the runs and use the last for localization. We use cross-validation by leaving one out technique, since we do not have a large data set for the real test scenario, and we don't want the simulation to differ too much from the real scenario. We first begin to evaluate the estimation errors of the filters, shown in Figure 7.4, where the errors are computed by

$$\mathbf{x}_{\epsilon} = \mathbf{x} - \hat{\mathbf{x}}. \tag{7.1}$$

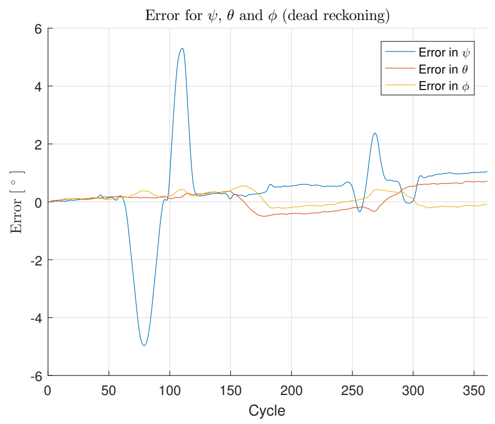
If the EKF is used with only the IMU measurements, and using the same noise parameters as for the two proposed solutions, the drift in X, Y and Z accumulate a large error. There is a smaller, albeit more clear, drift in yaw, pitch and roll. This is due to the measurement accuracy of their respective angular rate, which means that only a small error will arise in each cycle. From Figure 7.4a and 7.4b the accumulative errors are visualized, and both indicate that the internal sensors in fact cannot determine the global positioning due to drifting. The anomaly is Y and Z, which revert back toward zero error, but then diverge again. This is caused by the trajectory, the vehicle first turns left and then in the other direction which means that the error for Y will pass zero before starting to diverge again. For Z it is the hill and the lane change in the velodrome curve that cause the zigzagging.

When the gradient search is employed, the algorithm can estimate the ego-position with an accuracy within 0.2 meters of the ground truth data in most cycles. There are some two up to $0.4 - 0.5$ meters, which is at the start of the trajectory and after exiting an acute curve. The larger uncertainty in Z originate from the uncertainty of the radar in elevation (v). As the map will not be as distinct in height compared to the horizontal plane, it will not be possible to estimate the position as accurately. Figure 7.4c present the error in X, Y and Z, between the estimated states and the ground truth. Angular errors are presented in Figure 7.4d, but the improvement compared to Figure 7.4b is harder to see than for the positions. However, given a longer trajectory the drift would become so large that the improvement is obvious. The algorithm can estimate the angles pitch and roll within 1 ° for the entire trajectory, which is very good given that the uncertainty of the radar is 0.5 ° in azimuth and 2 ° in elevation. The yaw estimation is is very good for most of the trajectory, but there are 4 peaks that reach undesirably large values. They occur in connection with a swift lane change in a hill. Comparing the same region with the dead reckoning case, a clear improvement can be seen. The plots show that all drifts are successfully removed as all metrics fluctuate around zero mean error.

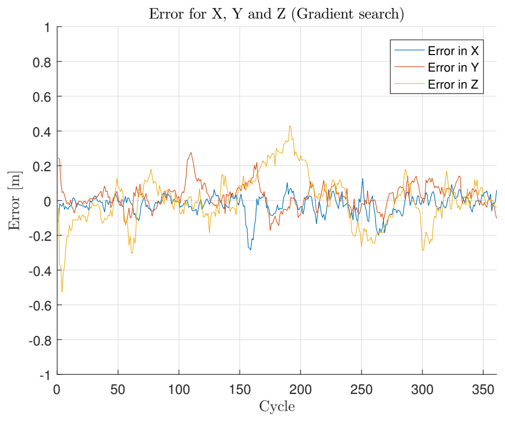
Lastly, the results for the modified Rao-Blackwellized particle filter are shown in Figure 7.4e and 7.4f. Similar to the gradient search, it is successful in removing the drift. It does estimate the position and the angle with similar accuracy to the gradient search, but further comparisons have to be made to clearly distinguish if one performs better than the other. Four peaks around $\pm 2°$ can be seen in the estimation, which are related to lane changes and swift curve changes. Comparing the



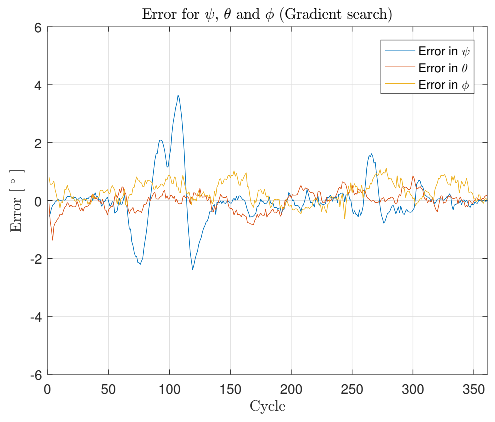
(a) Errors for X, Y and Z in meters, EKF with only IMU.



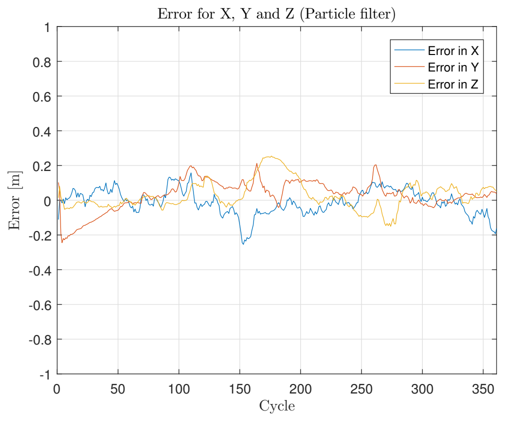
(b) Errors for ψ, θ and ϕ in °, EKF with only IMU.



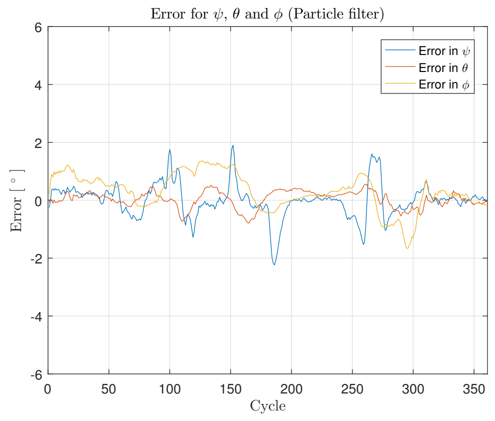
(c) Errors for X, Y and Z in meters, EKF with gradient search.



(d) Errors for ψ, θ and ϕ in °, EKF with gradient search.



(e) Errors for X, Y and Z in meters, Rao-Blackwellized particle filter.



(f) Errors for ψ, θ and ϕ in °, Rao-Blackwellized particle filter.

Figure 7.4: The errors, $\mathbf{x}_\epsilon$, for the three filters in the simulation environment, which are computed according to equation (7.1).

lane change with the other two filters, the particle filter is superior in estimating ψ. The positional estimates are very good, almost entirely within 0.2 meters compared to ground truth, for the entire trajectory.

The Euclidean distance error (Root Square Error, RSE),

$$d_\epsilon = \sqrt{(X - \hat{X})^2 + (Y - \hat{Y})^2 + (Z - \hat{Z})^2} \tag{7.2}$$

is computed for the three filters, and Figure 7.5 present the error values for the entire trajectories. EKF with only the IMU shows a clear drift, where the descent originate from the previously explained form of the trajectory. Both the gradient approach and the particle filter fix this drift, and have very similar RSE. Both move within 0.25 m error in most cycles, which is a very good result given that the grid size is 0.2 meters, and that the radar resolution is 0.3 meters in range.

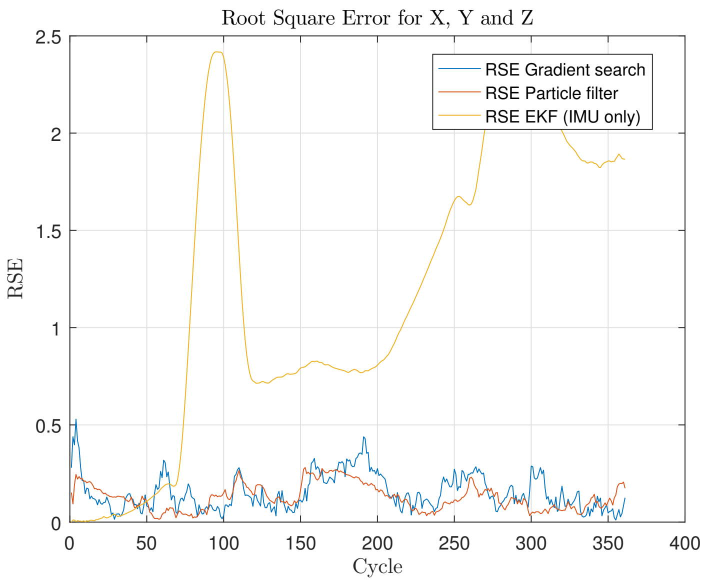


Figure 7.5: Euclidean distance error for the positions, X, Y and Z for the three filters. The EKF with only IMU produce bad positional estimates, while the two proposed solutions show promising results.

To evaluate which filter performs the best, we compute mean values and standard deviations for the three filters. Table 7.1 present the mean values, μ, and standard deviations, σ, of the error for the entire trajectories for the three filters. As the figures have illustrated, the two proposed solutions have mean values of the error around zero, i.e. the estimates vary around zero mean. This is not the case for the EKF with only IMU measurements, where the drift push the mean error away from zero. The mean error would be even larger if the trajectory would be less symmetric, i.e. that the vehicle does not do a S-turn but instead travelling mainly in one direction. This is also the reason why the angular estimates are very good for the EKF without the radar input, and the low standard deviations also depend on very good IMU measurements.

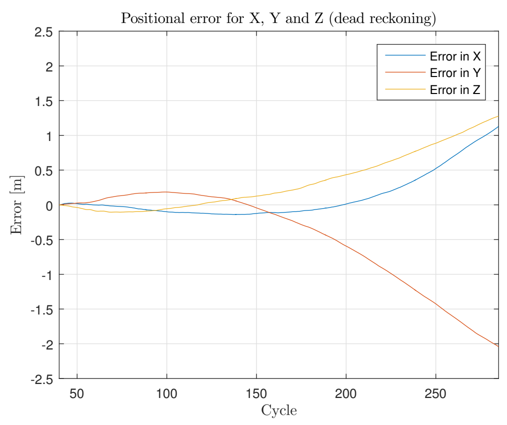
Mean values themselves do not represent the performance of the filter, huge offsets in opposing directions would cancel out and not be noticed. More interesting for evaluation purposes are the standard deviations of the different filters, which indicate how well the proposed solutions fluctuate from the error mean. The standard deviation of the error correspond to Root Mean Square Error, RMSE, for an unbiased estimator. It is a very good metric of how well something is estimated. The standard deviations in Table 7.1 show that the EKF is worse in all categories except for roll, ϕ. The data reveal that the particle filter solves the estimation slightly better than the gradient search. The gradient search does estimate X and Y and ϕ better, but the difference in Z and ψ is so large that the particle filter overall performs better. The conclusion of Table 7.1 is that, since both mean values and standard deviations are low for both proposed solutions, the localization algorithms provides good estimates throughout the entire sequence, but the particle filter is on average slightly better.

		μ		σ	
EKF (only IMU)	X	-0.7073	m	0.6981	m
	Y	-0.6585	m	0.6535	m
	Z	-0.2188	m	0.3323	m
	ψ	0.3658	°	1.4079	°
	θ	0.1047	°	0.3541	°
	ϕ	0.1017	°	0.2155	°
Gradient search	X	-0.0233	m	0.0256	m
	Y	0.0239	m	0.0711	m
	Z	0.0002	m	0.1477	m
	ψ	-0.0049	°	0.8523	°
	θ	-0.0205	°	0.3213	°
	ϕ	0.2943	°	0.3568	°
Particle filter	X	-0.0106	m	0.0718	m
	Y	0.0300	m	0.0849	m
	Z	0.0281	m	0.0816	m
	ψ	-0.0079	°	0.5949	°
	θ	0.0369	°	0.2852	°
	ϕ	0.3048	°	0.6258	°

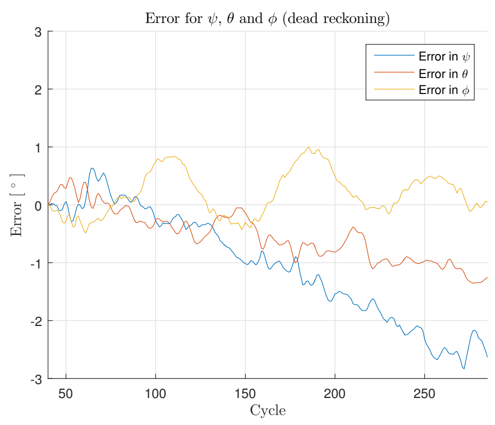
Table 7.1: The table consists of mean values and standard deviations for the error of the localization metrics. The values are computed for all cycles of the simulation.

7.3 Test case performance

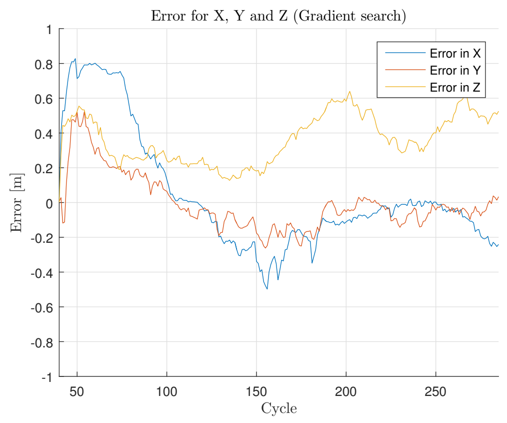
To evaluate the localization for the test case, the OGM was created with three of the four runs shown in Figure 7.2, and the fourth run was used for localization. The errors of the filter estimations are shown in Figure 7.6. The errors are computed according to equation (7.1).



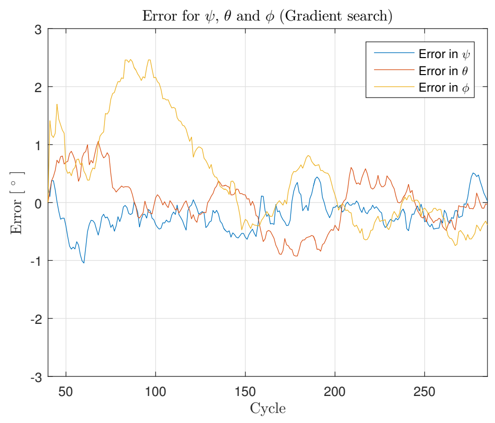
(a) Errors for X, Y and Z in meters, EKF with only IMU.



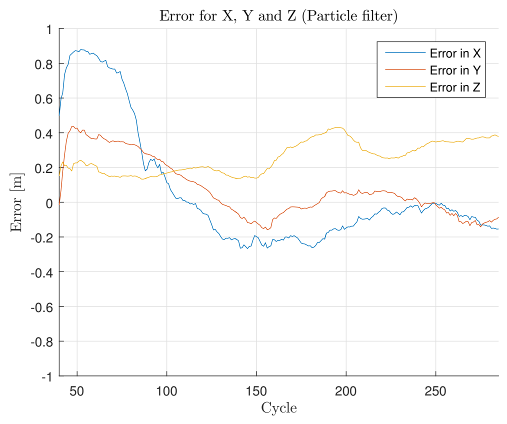
(b) Errors for ψ, θ and ϕ in °, EKF with only IMU.



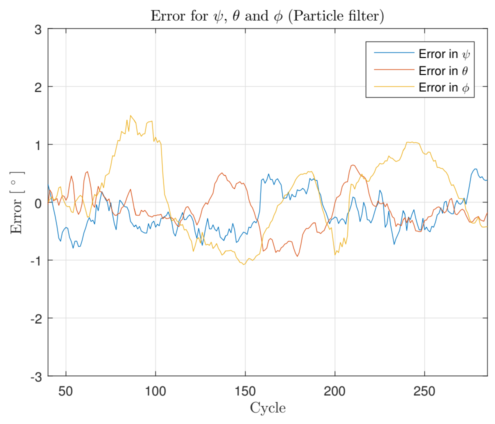
(c) Errors for X, Y and Z in meters, EKF with gradient search.



(d) Errors for ψ, θ and ϕ in °, EKF with gradient search.



(e) Errors for X, Y and Z in meters, Rao-Blackwellized particle filter.



(f) Errors for ψ, θ and ϕ in °, Rao-Blackwellized particle filter.

Figure 7.6: The errors, $\mathbf{x}_\epsilon$, for the three filters in the test case (guard rail), which are computed according to equation (7.1).

The dead reckoning case show a clear drift in all variables except the roll, ϕ. It can be seen that both localization methods manage to compensate for this drift. The estimation of roll, ϕ, pitch, θ, and altitude, Z, are a bit worse than the other three. This could be a consequence of the uncertainty being larger in elevation than in azimuth, and all the three aforementioned states are correlated to the altitude of the detections. Another factor is that the reference data has higher uncertain in those three states, which as a consequence makes the map even more blurry in the Z direction. The alignment in Z between different runs of the reference data are faulty. There should be a maximum of a couple of centimeters of errors in height between the runs, but it is instead around one decimeter. This, together with the higher uncertainty in v cause the more uncertain estimates for Z.

As was described earlier in Section 7.1, the map is bad before the guard rail appears. The test run for the localization starts to detect the guard rail around cycles 60-80, which is why the localization tests and evaluations do not start at cycle one. We can see in the results that the filter estimates drift in the beginning, but converge as soon as the environment is rich enough.

The results indicate that the particle filter method performs slightly better estimations than the gradient search, especially for the orientation. It has the same performance issue in the first cycles as the gradient search, also due to the bad map.

The RSE is computed for real data as well, see equation (7.2). As the car does not turn in both directions in the test case, the imminent drift can be seen clearly. The filters do remove the drift, but there is a big deviation at the first 80 cycles caused by the bad map. When looking at the estimates from cycle 80 and forward, the results are similar, albeit a bit worse, compared to the simulation tests.

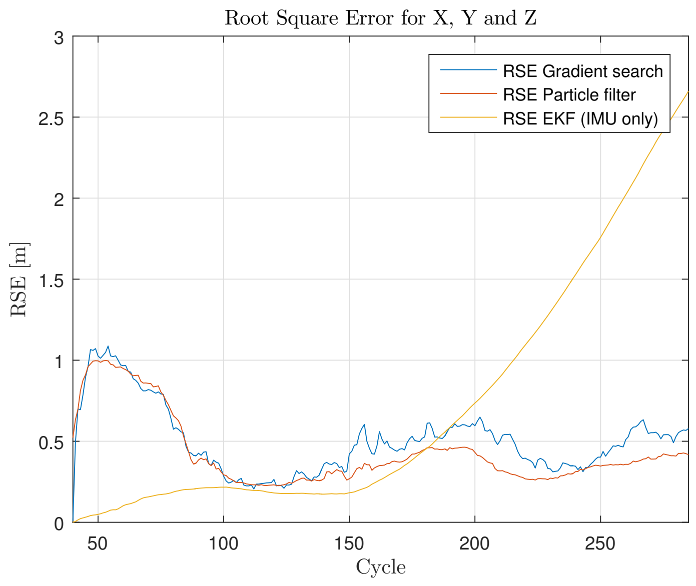


Figure 7.7: Euclidean distance error for the positions, X, Y and Z for the three filters.

The values in Table 7.2 are computed from cycle 80 to the end. As the first part of the map is bad, the localization performance in the beginning impair the mean values and standard deviations. We base the thesis on that we have a rich environment, which is why we remove that part when evaluating the values in the table. The table immediately reveal the performance increase by the two proposed solutions, where the particle filter (again) outperform the gradient search.

		μ		σ	
EKF (only IMU)	X	0.1633	m	0.3705	m
	Y	-0.6480	m	0.6978	m
	Z	-0.2834	m	0.3940	m
	ψ	-1.4185	°	0.7752	°
	θ	-0.7123	°	0.3372	°
	ϕ	-0.2679	°	0.3804	°
Gradient search	X	-0.1248	m	0.1251	m
	Y	-0.0766	m	0.0716	m
	Z	0.3700	m	0.1450	m
	ψ	-0.1811	°	0.2471	°
	θ	-0.1359	°	0.3742	°
	ϕ	0.1660	°	0.6835	°
Particle filter	X	-0.1176	m	0.0919	m
	Y	0.0049	m	0.0860	m
	Z	0.2834	m	0.0882	m
	ψ	-0.1628	°	0.3257	°
	θ	-0.1800	°	0.3583	°
	ϕ	-0.0114	°	0.6441	°

Table 7.2: The table consists of mean values and standard deviations for the error of the localization metrics for the test case. The values are computed from cycle 80 to remove the transient phase with bad environment.

7.4 Performance of Rao-Blackwellized modification

The simplification to the Rao-Blackwellized particle filter (RBPF), described in Section 6.2.1, makes the particle filter more computationally efficient. But to understand how this simplification affects the results, both methods were used on the real test case. Both filters, the full RBPF and the modified RBPF, were tested with the same number of particles and the RSE from the two runs can be seen in Figure 7.8. The results are very similar, and the simplified version seem to perform as well as the full RBPF. To also give an example of how much the computational performance improved, both methods were tested in MATLAB and time logged with MATLAB's profiler function for 10 cycles. The result showed that the simplified method solved it in approximately two seconds, while the full method took over two minutes. The full RBPF implementation has some bottlenecks that probably could be solved, but it should still take no less than approximately 20 seconds. The result from this test is just an approximation, but indicate that the simplification makes the method much more computationally efficient, despite no loss in estimation accuracy.

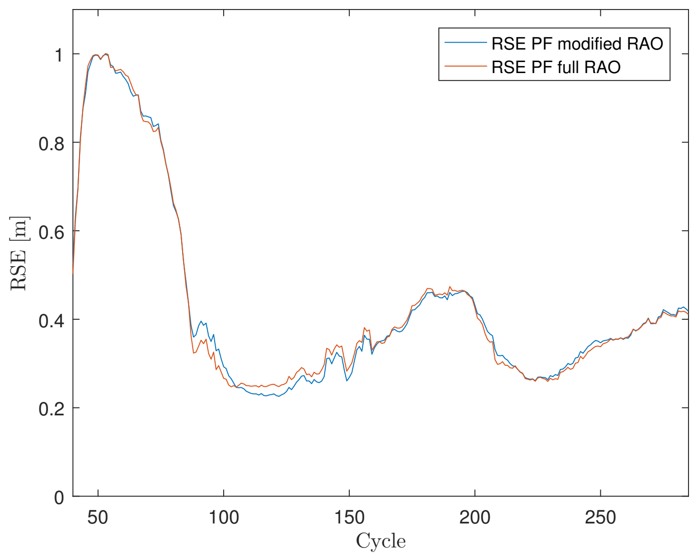
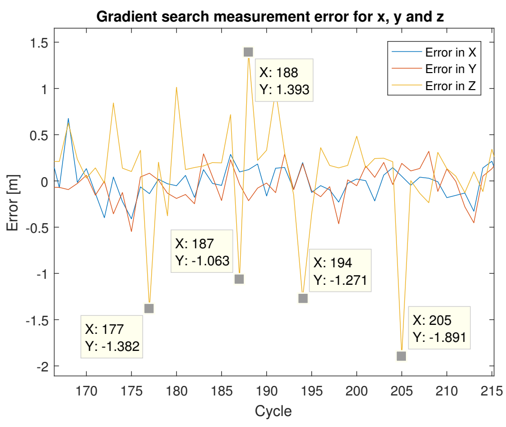


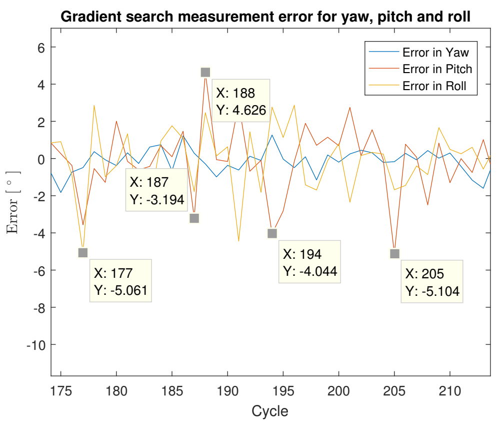
Figure 7.8: Euclidean distance error for the positions, X, Y and Z for a full Rao-Blackwellized particle filter and the modified Rao-Blackwellized particle filter.

7.5 Multiple model performance

The concept of the multiple models and why we use it can be found in sections 3.2.3 and 6.1.3 respectively. The implementation has to be validated, does it work like we intend it to do? From the simulated case, a part of the trajectory is chosen where the gradient search performs badly, and we highlight that the multiple models in fact work as intended. Figures 7.9a and 7.9b show five different cycles, where the gradient search has moved far and ended up in the incorrect local optima. The errors are calculated by the state value found by gradient search minus the ground truth data. The weights of the two models are shown in Figure 7.10, and the performance is apparent. Whenever the predicted position and the measurement from the gradient search differ too much, the high covariance model peaks and the input from the map is added with a high uncertainty. From the error plots we can see that when this occurs, the gradient search has indeed moved into an incorrect optima. Figure 7.10 imply that the gradient search also works well, as the low covariance model has a high weight most of the cycles.



(a) Positional error computed by gradient search value minus ground truth, for cycles with large errors.



(b) Angular error computed by gradient search value minus ground truth, for cycles with large errors.

Figure 7.9: The two MATLAB plots illustrate the difference between the gradient search found values, compared to ground truth. The data points further highlight which cycle that is evaluated.

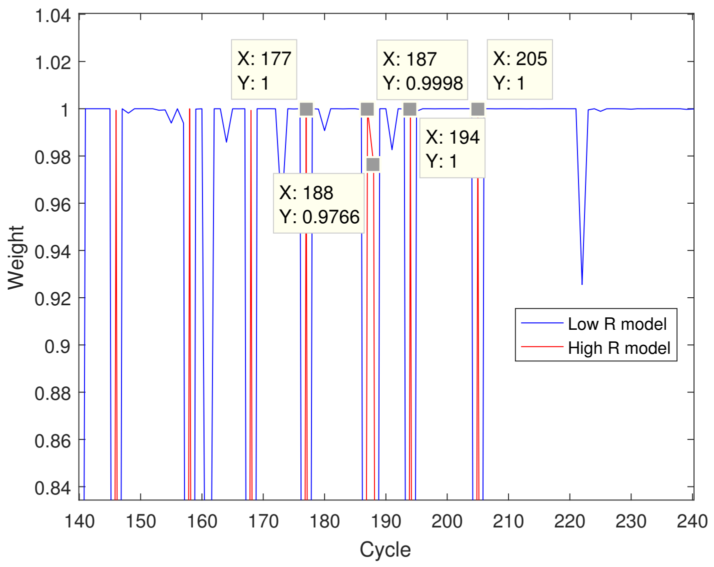


Figure 7.10: The figure illustrates the weight distribution between two models, where the blue represent the model with low covariance, and the red represent the model with high covariance.

One could argue that we should not use the registration input at all when we have moved to an incorrect optima. However, the algorithm still provides the EKF with some information. The gradient search cannot move several meters, due to low step lengths, and the incorrect local optima is still close to the global optima. This means that the information is not destructive for convergence. This is of course also conditioned on a good map and few erroneous radar detections. Our utilization of the multiple models imply a negative aspect, once the estimation has moved too far away from the true state, it will be extremely slow to converge back, if at all. By having a good prior and a good motion model, the prior for the multiple models will be good, and this case will not occur as the multiple models will keep the estimates from diverging from the true trajectory.

8
Discussion and future work

The localization results in simulation are better than the ones on real data, which is not too surprising. When working with real data, there tend to always arise both suspected and unsuspected problems. One known issue we have is that the transmitter antenna is defective, which cause (aliased) erroneous detections. This leads to a worse map, and also worse localization. Another issue for the localization is that the test case environment is bad in the beginning, and so also the map. There are no objects, e.g. a guard rail, which generates consistent detections. This can be seen in the first approximately 80 cycles of the localization estimation. There are a lot of false detections, both aliased detections and due to that the car stands still with the radar running and gathering detections from clutter. When skipping the first part of the map, prior the guard rail, the estimation errors go down to very similar values of the simulation. This imply that the localization requires maps where there are good scatterers and not just clutter. Continuing on the subject of environment, it would be interesting to see how well our proposed solutions would work in an urban environment. We have shown that it works with mainline driving with a guard rail, but how will it perform in an inner city environment, with more houses, other cars and traffic signs? Since the radar will not retrieve endless amount of detections, when there are too many dynamical objects in the radar FOV, which are also strong scatterers, the localization is prone to failure.

Another thing that would enhance the solution is to have side-looking radars instead of a front-looking radar. This is because of ambiguity between Doppler and high DOA angles when the radar is looking in the direction of the velocity of the vehicle. This sensor placement would also allow the radars to see more of the environment around the roads, rather than what is on it. This is more desirable for localization purposes. More radars would also be better, since each radar can produce as many detections as just one, having several could allow more detections being mapped to the environment and, thus, a better localization can be performed. Furthermore, future work based on radar setups is to use radars for both dynamic target tracking and localization simultaneously. This can be done as radar can distinguish dynamic and static targets by Doppler. Another interesting aspect of using radar for localization is to try to utilize other types of maps (e.g. created by lidar) and evaluate the possibility of accurate ego-positioning.

When comparing the localization results, we notice that the particle filter works better than the gradient search approach for real data in our specific test case. Despite the scenario being the same, there exist many parameters to tune and the gradient search might work better for another scenario. Further tests have to be made, on varying environments, to determine if one approach definitely is better than the other. Both localization algorithms could also be tested with other types of sensors, e.g. test the Rao-Blackwellized particle filter (RBPF) when using lidar, and a map created by lidar. The RBPF should be easy to implement for other sensor installations. The implementation for localization is very general and does, currently, only depend on the format of the map (which can be changed). More sensors, and their measurements, can easily be incorporated in the current framework. Our tricubic interpolation implementation could be utilized on any discrete map, where a continuous map is desired.

When performing registration between two data sets, it is often considered as an optimization problem. Most optimization methods assume continuous variables, and our solution with tricubic interpolation is quite ingenious. Not only does it circumvent the issue of a discrete map, but it also produces a smooth function, C^1, between grid points. More demanding optimization methods could, thus, have been used, e.g. Newton-Raphsons [28, p.285], but given that the approach already is quite computationally heavy and we had to focus on other areas, we decided that a more demanding algorithm was not worth the time compared to working on the filter designs. Noteworthy is that the 64×64 matrix used in tricubic interpolation can be optimized, it contains many zeroes and the amount of operations can be decreased by a smarter implementation than matrix multiplication. There exist more computationally efficient methods to create a smooth likelihood from a discrete map than tricubic interpolation. Likelihood fields and NDT solve a similar problem problem by estimating the map as Gaussian. The methods introduce more uncertainties in the likelihood, but the calculations are easier. Also, trilinear interpolation may work to some extent, but the discontinuous derivatives between grid points could create problems during a gradient search.

The vehicle model works well for describing the movement of a vehicle with six degrees of freedom. However, it is of course only a model and far from perfect. The Ackermann or bicycle model could be incorporated to improve the vehicle model further. By including front wheel angle, FWA, and front wheel speed, FWS, measurements, we can get another measurement for body fixed rotations around z, w_z. This would improve the estimation of heading, which is one of the most important states for a vehicle. We considered doing this, but as we couldn't find any wheel speed measurements on the front wheel, and a general model between the back wheel speed and front wheel speed is very difficult, we omitted the improvement.

Creating offline maps with the inverse sensor model has proven to work well, but the map generation would most likely be better if a forward model is used. The possibility of creating a good forward radar model to use with OGM can be investigated, and new algorithms could possibly be created which can achieve better maps. Further, a general improvement for OGM is to utilize adaptive grid cell sizes, which would severely decrease the amount of required data stored in the map. This would be vital in a real-time implementation. For an OGM algorithm with ISM to be robust, it must be able to handle accumulation of evidence of occupancy/empty space in a smart way. If the log-odds ratio becomes too large, and the static environment changes, it takes a long time for the algorithm to remove the evidence. This can partially be solved by thresholds on the log-odds ratio, but this will eventually flatten out the peaks around scatterers which is not desirable.

In this thesis the state estimation of the vehicle has been for six degrees of freedom, which may seem a bit unnecessary. Without doubt, the most important states for a vehicle is X, Y and ψ, i.e. planar movement and heading, but since 3D data were available from the radar a full state estimation could be done. We considered a general solution to be better, where future simplifications could be done to reduce the state estimation to the aforementioned dimensions. To keep the state estimation in 3D, but without having to estimate Z, θ, or ϕ from the radar detections, another map could be created which contain information about altitude and inclination of the road. This input could be utilized in the same as the radar, where the input to the map is only the estimated position of the vehicle, which could improve the estimation of the aforementioned variables.

9
Conclusion

This thesis conclude that it is possible to create good occupancy grid maps, using an inverse sensor model for radar, and to use these maps for localization of an ego-vehicle travelling within the map. The map is created offline, and by evaluation in the simulation environment it is apparent that the map creates occupied areas around the position of objects. Both proposed localization solutions, the Rao-Blackwellized particle filter and the gradient search with tricubic interpolation smoothing of the map, are able to successfully remove drifts originating from localization by dead reckoning and the filter accuracy keep the estimates within 0.25 meters ($\mathbb{R}^3$ Euclidean distance) for simulation and within 0.3 meters on real data in the X-Y plane ($\mathbb{R}^2$ Euclidean distance). Due to problems with the reference data, the estimation results in Z is non-representative. Of the two proposed solutions, the particle filter is able to perform the best state estimation. Based on the localization results, radar sensors known robustness against rough conditions, and the ability to measure relative radial velocity, we conclude that radar can play a vital part as a sensor used for localization.

Bibliography

[1] Fahed Abdallah, Amadou Gning, and Philippe Bonnifait. “Box particle filtering for nonlinear state estimation using interval analysis”. In: *Automatica* 44.3 (2008), pp. 807–815.

[2] Bradley M Bell and Frederick W Cathey. “The iterated Kalman filter update as a Gauss-Newton method”. In: *IEEE Transactions on Automatic Control* 38.2 (1993), pp. 294–297.

[3] Ben Bellekens, Vincent Spruyt, and Rafael Berkvens Maarten Weyn. “A survey of rigid 3D pointcloud registration algorithms”. eng. In: *Fourth International Conference on Ambient Computing, Applications, Services and Technologies, Proceedings*. Rome, Italy: IARA, 2014, pp. 8–13. ISBN: 9781612083568. URL: `http://www.thinkmind.org/download.php?articleid=ambient\_2014\_1\_20\_40015`.

[4] Peter Biber and Wolfgang Straßer. “The normal distributions transform: A new approach to laser scan matching”. In: *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on.* Vol. 3. IEEE. 2003, pp. 2743–2748.

[5] Antoni Burguera, Yolanda González, and Gabriel Oliver. “The likelihood field approach to sonar scan matching”. In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on.* IEEE. 2008, pp. 2977–2982.

[6] T.-T. V. Cao. “Constant false-alarm rate algorithm based on test cell information”. In: *IET Radar, Sonar & Navigation (Volume:2, Issue:3)* (2008), p. 200. DOI: `10.1049/iet-rsn:20070133`.

[7] S Cavassila et al. “Cramér–Rao bounds: an evaluation tool for quantitation”. In: *NMR in Biomedicine* 14.4 (2001), pp. 278–283.

[8] Frederic Chausse, Jean Laneurit, and Roland Chapuis. “Vehicle localization on a digital map using particles filtering”. In: *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE.* IEEE. 2005, pp. 243–248.

[9] Steve Clark and Gamini Dissanayake. “Simultaneous localisation and map building using millimetre wave radar to extract natural features”. In: *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on.* Vol. 2. IEEE. 1999, pp. 1316–1321.

[10] Alberto Elfes. “Occupancy grids: A stochastic spatial representation for active robot perception”. In: *arXiv preprint arXiv:1304.1098* (2013).

[11] A. Foessel-Bunting. “Radar Sensor Model for Three-Dimensional Map Building”. In: *Proc. SPIE 4195, Mobile Robots XV and Telemanipulator and Telepresence Technologies VII* (2001), p. 127. DOI: `10.1117/12.417295`.

[12] F Redrik Gustafsson and Alf J Isaksson. “Best choice of coordinate system for tracking coordinated turns”. In: *Decision and Control, 1996., Proceedings of the 35th IEEE Conference on.* Vol. 3. IEEE. 1996, pp. 3145–3150.

[13] Xiaoqing Hu et al. “Generalized Iterated Kalman Filter and its Performance Evaluation”. In: *Signal Processing, IEEE Transactions on* 63.12 (2015), pp. 3204–3217.

[14] Daniek Joubert. “Adaptive occupancy grid mapping with measurement and pose uncertainty”. PhD thesis. Stellenbosch University, 2012.

[15] R.E. Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *J. Basic Engineering, 82 (Series D): 35-45* (1960), pp. 35–45. DOI: `10.1115/1.3662552`.

[16] Viktor Kärnstrand. “Forward filtering and backward smoothing for radar applications”. MA thesis. Chalmers University of Technology, 2012.

[17] Kurt Konolige. “Improved occupancy grids for map building”. In: *Autonomous Robots* 4.4 (1997), pp. 351–367.

[18] Francois Lekien and J Marsden. “Tricubic interpolation in three dimensions”. In: *International Journal for Numerical Methods in Engineering* 63.3 (2005), pp. 455–471.

[19] Li Liang-Qun, Ji Hong-Bing, and Luo Jun-Hui. “The iterated extended Kalman particle filter”. In: *Communications and Information Technology, 2005. ISCIT 2005. IEEE International Symposium on.* Vol. 2. IEEE. 2005, pp. 1213–1216.

[20] Malin Lundgren. “Bayesian Filtering for Automotive Application”. PhD thesis. Chalmers University of Technology, 2015.

[21] Mathworks. *Constant False Alarm Rate (CFAR) Detection.* URL: `http://se.mathworks.com/help/phased/examples/constant-false-alarm-rate-cfar-detection.html?requestedDomain=www.mathworks.com`.

[22] Michael Montemerlo et al. “FastSLAM: A factored solution to the simultaneous localization and mapping problem”. In: *Aaai/iaai.* 2002, pp. 593–598.

[23] Malte Moritz and Anton Pettersson. “Estimation of Local Map from Radar Data”. MA thesis. Linköpings universitet, tekniska högskolan, 2014.

[24] John Mullane et al. “Including probabilistic target detection attributes into map representations”. In: *Robotics and Autonomous Systems* 55.1 (2007), pp. 72–85.

[25] NHTSA. *Crash Factors in Intersection-Related Crashes: An On-Scene Perspective.* `http://www-nrd.nhtsa.dot.gov/Pubs/811366.pdf`. Accessed: 2015-10-14.

[26] NHTSA. *Distracted Driving 2014.* `http://www-nrd.nhtsa.dot.gov/Pubs/812260.pdf`. Accessed: 2016-06-07.

[27] Michael Parker. *Radar Basics Part 1.* 2011. URL: `http://www.eetimes.com/document.asp?doc_id=1278779`.

[28] Michael Patriksson et al. *Introduction to Continuous Optimization.* Studentlitteratur, 2013.

[29] Ali Ufuk Peker, Oğuz Tosun, and Tankut Acarman. “Particle filter vehicle localization and map-matching using map topology”. In: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE. 2011, pp. 248–253.

[30] Carlos Gálvez del Postigo Fernández. “Grid-Based Multi-Sensor Fusion for On-Road Obstacle Detection: Application to Autonomous Driving”. MA thesis. KTH Royal Institute of Technology, 2015.

[31] R.S. Bucy R.E. Kalman. “New Results in Linear Filtering and Prediction Theory”. In: *J. Basic Eng 83(1), 95-108 (Mar 01, 1961)* (1961), pp. 95–108. DOI: `10.1115/1.3658902`.

[32] R Rouveure, MO Monod, and P Faure. “High resolution mapping of the environment with a ground-based radar imager”. In: *Radar Conference-Surveillance for a Safer World, 2009. RADAR. International*. IEEE. 2009, pp. 1–6.

[33] Robert Popoli Samuel Blackman. *Design and Analysis of Modern Tracking Systems*. Artech House Radar Library. Artech House, 1999.

[34] Ralph O Schmidt. “Multiple emitter location and signal parameter estimation”. In: *Antennas and Propagation, IEEE Transactions on* 34.3 (1986), pp. 276–280.

[35] Thomas Schön, Fredrik Gustafsson, and Per-Johan Nordlund. “Marginalized particle filters for mixed linear/nonlinear state-space models”. In: *Signal Processing, IEEE Transactions on* 53.7 (2005), pp. 2279–2289.

[36] Robin Schubert, Eric Richter, and Gerd Wanielik. “Comparison and evaluation of advanced motion models for vehicle tracking”. In: *Information Fusion, 2008 11th International Conference on*. IEEE. 2008, pp. 1–6.

[37] Adrian Smith et al. *Sequential Monte Carlo methods in practice*. Springer Science & Business Media, 2013.

[38] Pan Song, Chang-fu Zong, and Masayoshi Tomizuka. “A terminal sliding mode based torque distribution control for an individual-wheel-drive vehicle”. In: *Journal of Zhejiang University SCIENCE A* 15.9 (2014), pp. 681–693.

[39] Robert Stengel. *Aircraft Equations of Motion - 2*. 2014. URL: `https://www.princeton.edu/~stengel/MAE331Lecture9.pdf`.

[40] Lennart Svensson. *SSY320, Sensor fusion and nonlinear filtering*. 2015.

[41] Fox Thrun Burgard. *probabilistic robotics*. Intelligent Robotics and Autonomous Agents series. The MIT Press; Intelligent Robotics and Autonomous Agents series edition, 2005.

[42] Sebastian Thrun. “Learning occupancy grid maps with forward sensor models”. In: *Autonomous robots* 15.2 (2003), pp. 111–127.

[43] L Vermare, P Hennequin, ÖD Gürcan, et al. “Detection of geodesic acoustic mode oscillations, using multiple signal classification analysis of Doppler backscattering signal on Tore Supra”. In: *Nuclear Fusion* 52.6 (2012), p. 063008.

[44] James A. Scheer William L. Melvin. *Principles of Modern Radar: Volume 3*. SciTech Publishing, 2013. ISBN: 978-1-89112-154-8.

A

Appendix 1

We show the derivations behind how the Rao-Blackwellized particle filter is simplified when assuming no noise on the particle filter states, i.e. $Q^n = 0^{6\times6}$. We rearrange the terms and multiply with transposes in order to create square matrices. Since the matrices are not square in general, these calculations would not be viable without the approach (due to the inability to invert a non-square matrix). We base the calculations according to the equations presented in [35, p.4]. For simplicity, we drop the time dependency and note that $P = P_{t|t}$ and $P_2 = P_{t+1|t}$.

$$\begin{aligned}
N_t &= A_t^n P_{t|t}(A_t^n)^T + Q^n = A^n P(A^n)^T \\
L &= A^l P(A^n)^T (A^n P(A^n)^T)^{-1} \iff LA^n P(A^n)^T = A^l P(A^n)^T \\
LA^n P(A^n)^T A^n &= A^l P(A^n)^T A^n \\
LA^n P &= A^l P \\
LA^n &= A^l \\
LA^n(A^n)^T &= A^l(A^n)^T \\
L &= A^l(A^n)^T(A^n(A^n)^T)^{-1}
\end{aligned} \tag{A.1}$$

$$\begin{aligned}
P_2 &= A^l P(A^l)^T + Q^l - A^l(A^n)^T(A^n(A^n)^T)^{-1}A^n P(A^n)^T(A^l(A^n)^T(A^n(A^n)^T)^{-1})^T = \\
&= Q^l + A^l P(A^l)^T - A^l \underbrace{(A^n)^T(A^n(A^n)^T)^{-1}A_n}_{=\mathrm{I}} P \underbrace{(A^n)^T((A^n(A^n)^T)^{-1})^T A^n}_{=\mathrm{I}}(A^l)^T = \\
&= Q^l + A^l P(A^l)^T - A^l P(A^l)^T = Q^l,
\end{aligned} \tag{A.2}$$

Further, the update of the expected value can be significantly simplified as well.

When the nonlinear noise is zero, the derivations for the expected value becomes

$$\begin{aligned}
\mathbf{z}_k &= \mathbf{x}^n_{k+1} - f^n_k = A^n_k \mathbf{x}^l_k + \mathbf{w}^n_k = A^n_k \mathbf{x}^l_k \\
\hat{\mathbf{x}}^l_{k+1|k} &= A^l_k \hat{\mathbf{x}}^l_{k|k} + \mathbf{K}_k(\mathbf{z}_k - A^n_k \hat{\mathbf{x}}^l_k) = A^l_k \hat{\mathbf{x}}^l_{k|k} + \mathbf{K}_k(A^n_k \mathbf{x}^l_k - A^n_k \hat{\mathbf{x}}^l_k) \\
&= \{\mathbf{K}_k = A^l_k (A^n_k)^T (A^n (A^n)^T)^{-1}\} = \\
&= A^l_k \hat{\mathbf{x}}^l_{k|k} + A^l_k (A^n_k)^T (A^n (A^n)^T)^{-1} (A^n_k \mathbf{x}^l_k - A^n_k \mathbf{x}^l_k) \\
&= A^l_k \hat{\mathbf{x}}^l_{k|k} + A^l_k \underbrace{(A^n_k)^T (A^n (A^n)^T)^{-1} A^n_k}_{=I} (\mathbf{x}^l_k - \hat{\mathbf{x}}^l_k) \\
&= A^l_k \hat{\mathbf{x}}^l_{k|k} + A^l_k (\mathbf{x}^l_k - \hat{\mathbf{x}}^l_k) = A^l_k \mathbf{x}^l_k.
\end{aligned} \tag{A.3}$$

The variable $\mathbf{x}^l_k$ is the true state and not the estimated, which, as we argue, means that we can utilize it.

B
Appendix 2

The three figures, B.1, B.2 and B.3 show the map for the test case with different thresholds on log-odds, where the colors display the log-odds value for each grid. As the OGM is formed with noise, the green, less probable, parts encapsulate the more probable regions which can be hard to see here. The color bars, to the right of the plots, display what color a certain log-odds value is represented by in the map. The four boxes represent the car at different time instances in different runs, differentiated with colours. The full trajectory of each car is also shown with the same colors as the vehicles.

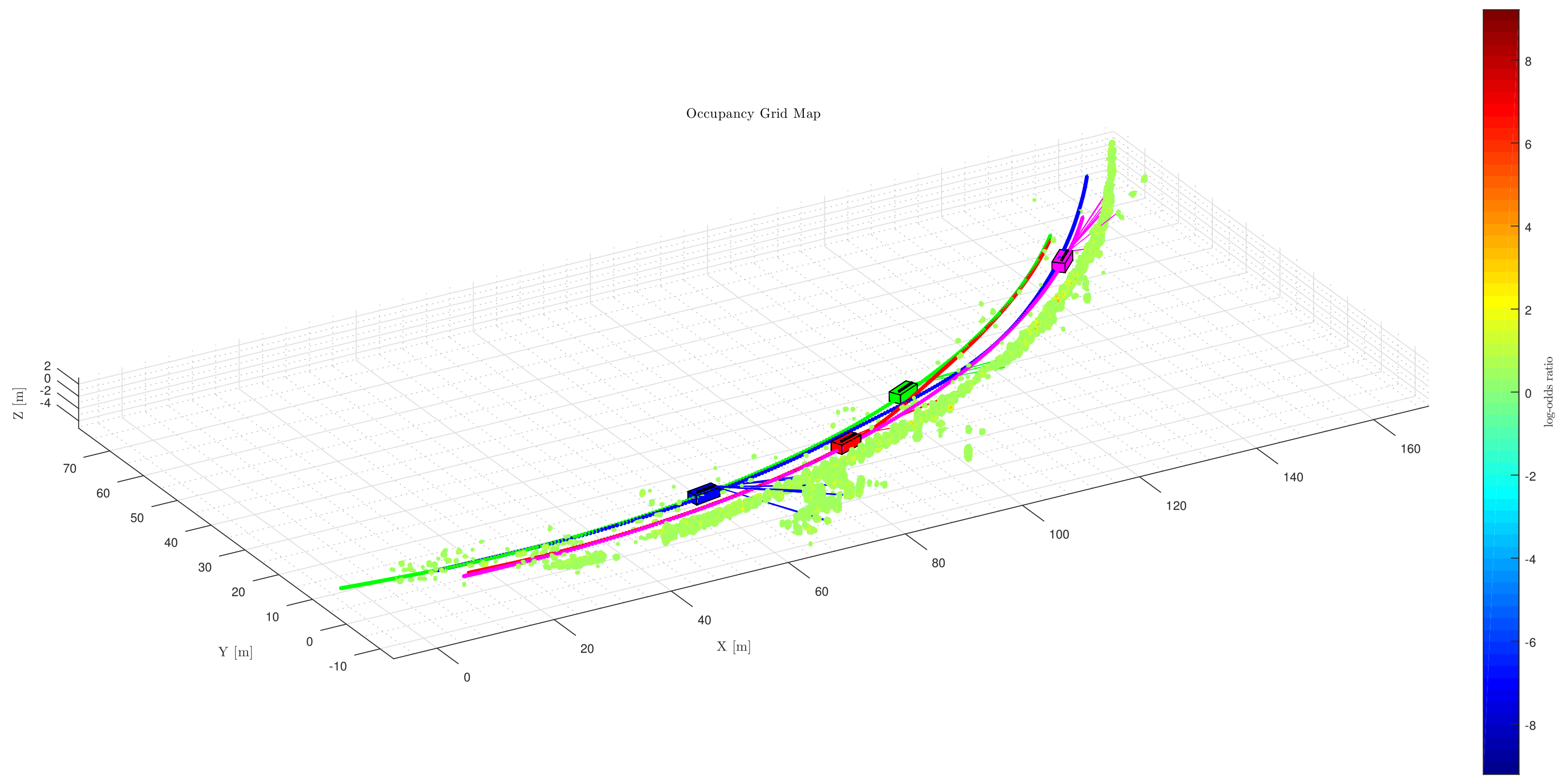


Figure B.1: Test scenario map. Grid cells with probability of occupancy more than 0.6 are shown.

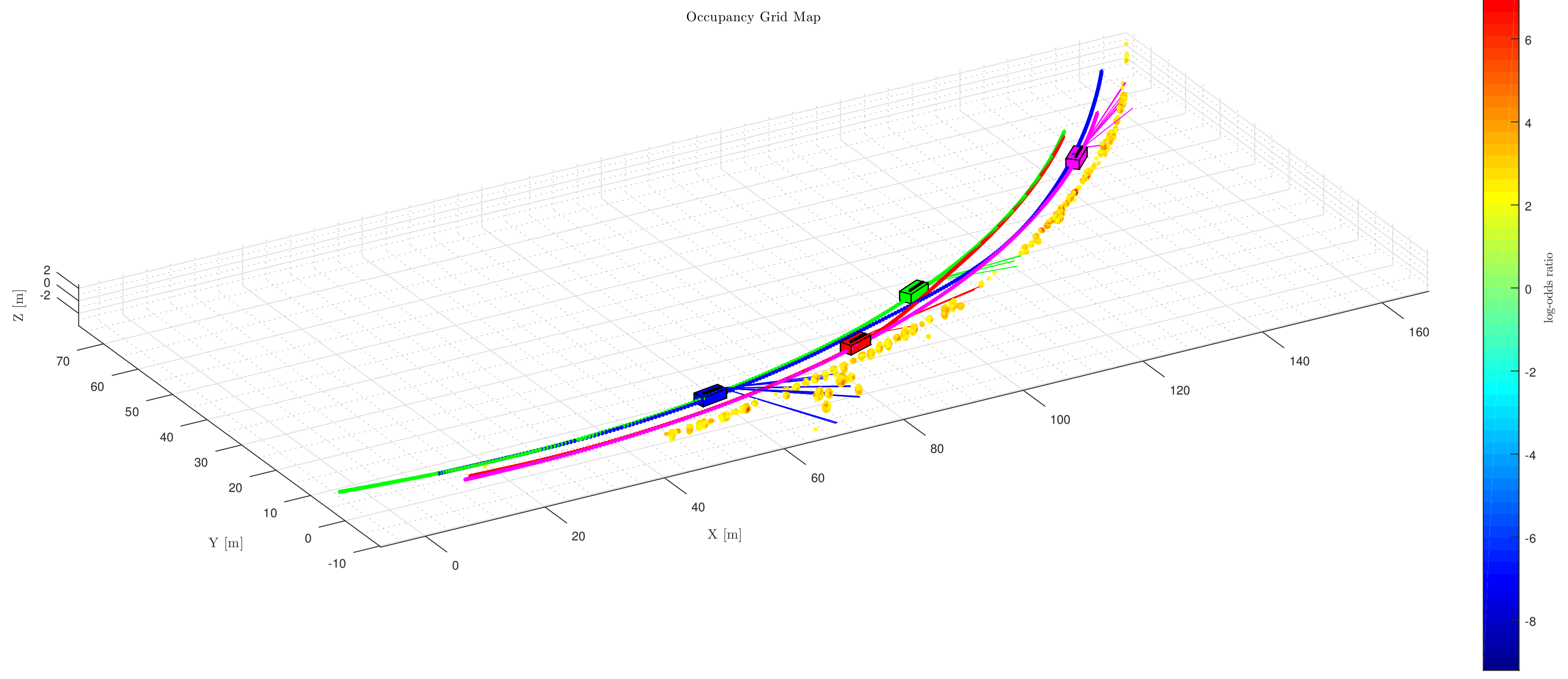


Figure B.2: Test scenario map. Grid cells with probability of occupancy more than 0.9 are shown.

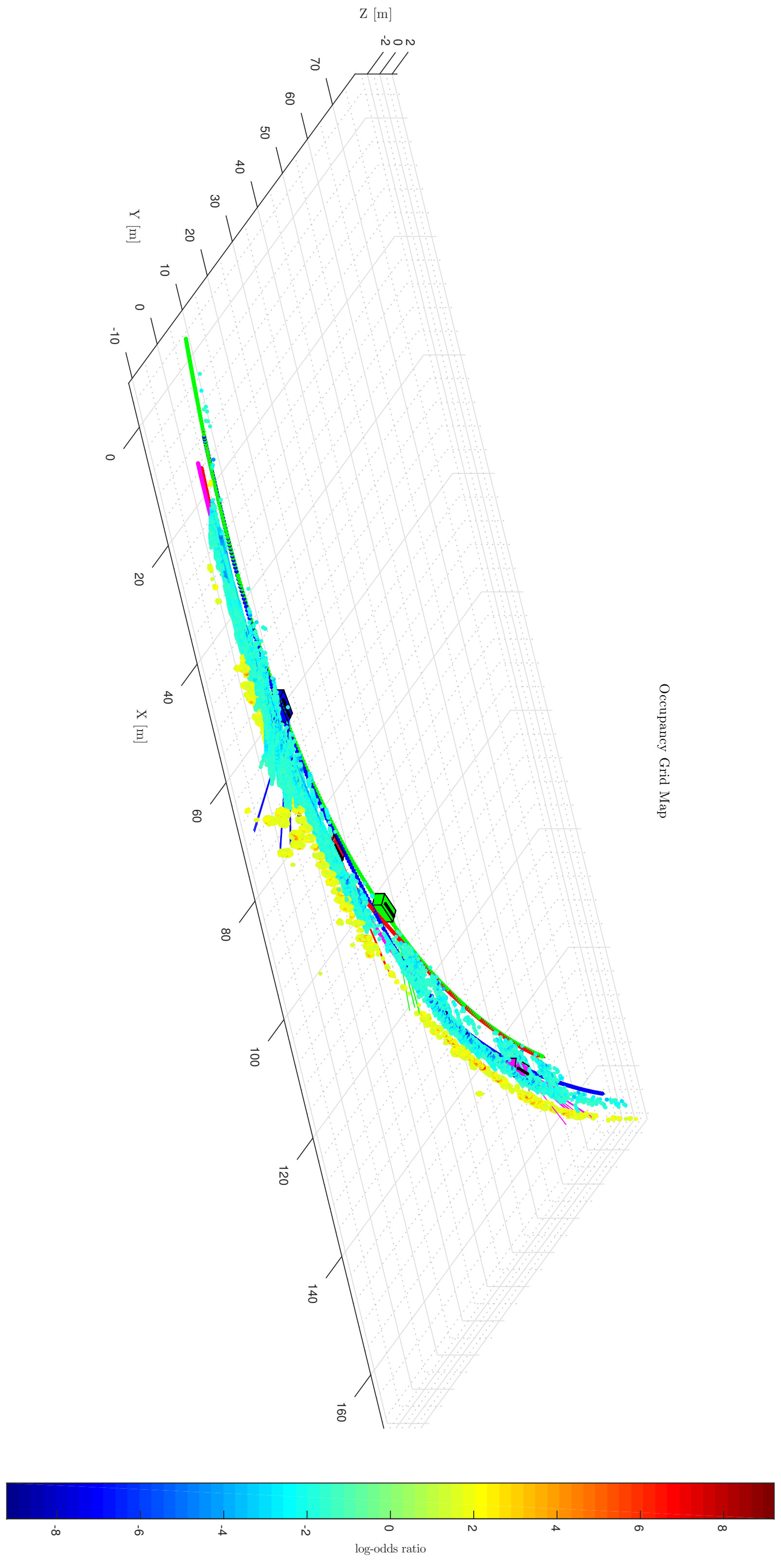


Figure B.3: Test scenario map. Grid cells with probability of occupancy more than 0.8 and less then 0.2 are shown.