TUM School of Computation, Information and Technology
Technische Universität München

Grid-based sensor data fusion for static and dynamic environment perception

Joseph Wessner

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz:
Prof. Dr.-Ing. Klaus Diepold

Prüfer*innen der Dissertation:
1. Prof. Dr.-Ing. Wolfgang Utschick
2. Priv.-Doz. Dr.-Ing. habil. Dirk Wollherr

Die Dissertation wurde am 08.06.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 23.01.2023 angenommen.

Danksagung

Diese Dissertation ist im Rahmen meiner Tätigkeit bei der *ZF Mobility Solutions GmbH* in Zusammenarbeit mit dem Lehrstuhl *Methoden der Signalverarbeitung* an der *Technischen Universität München* entstanden.

Ganz besonders bedanken möchte ich mich bei meinem wissenschaftlichen Betreuer und Doktorvater Prof. Dr. Wolfgang Utschick. Ohne seine Unterstützung wäre diese Arbeit sicherlich so nicht möglich gewesen. Ich möchte mich für die treffenden Analysen, hervorragenden Ideen und konstruktiven Diskussionen bedanken, die maßgeblich zu dieser Arbeit beigetragen haben.

Außerdem möchte ich meinem Betreuer und Vorgesetzten Dr. Frank Keck danken. Durch seine Unterstützung als Person und als Geschäftsführer der Firma ZF Mobility Solutions GmbH wurde diese Arbeit erst ermöglicht. Der regelmäßige Austausch mit vielen anregenden Diskussionen beeinflusste die Dissertation besonders positiv.

Des Weiteren möchte ich Dr. Arne Vater und meine Kollegen bei der ZF Mobility Solutions GmbH hervorheben, die stets für Fragen oder einen Gedankenaustausch zur Verfügung standen.

Meinen Eltern Hans und Maria danke ich für die Ermutigungen zu dieser Dissertation während des Studiums. Während der Anfertigung dieser Arbeit hielt mir meine Frau Andrea stets den Rücken frei und unterstützte mich mit ihrem Verständnis und ihrer Bestärkung, dafür danke ich ihr von Herzen. Ein besonderes Wort richte ich an meine beiden Töchter Nora und Ella, die mir mit ihrer Lebensfreude den Ausgleich zum Arbeitsalltag gebracht haben.

Zusammenfassung

Die Umfeldwahrnehmung spielt eine entscheidende Rolle bei modernen Fahrerassistenzsystemen oder Anwendungen für autonomes Fahren. Um die Limitierungen einzelner Sensoren zu überwinden, müssen die Daten mehrerer Sensoren kombiniert werden. Diese Sensordatenfusion spielt eine entscheidende Rolle als Teil eines sicherheitskritischen Systems, das auf die Informationen seiner direkten Umgebung angewiesen ist.

In dieser Arbeit stellen wir einen gridbasierten Ansatz zur Sensordatenfusion vor, der die Fusion in einem sehr frühen Verarbeitungszustand ermöglicht. Um die Sensordaten von Lidar- und Radarsensoren in dieser gridbasierten Fusion zu ermöglichen, beschreiben wir inverse Sensormodelle für beide Technologien. Mit Hilfe dieser inversen Sensormodelle werden die Sensorrohdaten in eine gridbasierte Darstellung, die für die Fusion benötigt wird, überführt.

Ein Nachteil von klassischen Belegungskarten ist, dass sie von einer statischen Umgebung ausgehen. Deshalb zeigen wir eine Erweiterung der klassischen Belegungskarte, die es nicht nur ermöglicht, die dynamische Belegung, sondern auch die Geschwindigkeit der einzelnen Zellen zu schätzen.

Des Weiteren schlagen wir vor, eine online generierte Orientierungskarte als a priori Information für die Initialisierung von Partikeln zu verwenden. Dieses zusätzliche Wissen führt zu einer schnelleren Konvergenz der Partikel und damit zu einer besseren Dynamikschätzung der Zellen. Dadurch verbessert sich auch die geschätzte Geschwindigkeit der extrahierten Objekte, was wiederum zu einer verbesserten Gesamtperformance des vorgestellten Ansatzes führt.

Mit der so erzeugten dynamischen Belegungskarte ist es möglich, den Freiraum im Sinne von befahrbarer Fläche zu extrahieren. Zusätzlich schlagen wir eine Methode vor, um (dynamische) Objekte aus der Belegungskarte zu extrahieren. Um die Qualität der extrahierten Objekte zu verbessern, verwenden wir einen Kalman-Filter basierten Ansatz zur Objektverfolgung. Durch das Bereitstellen dieser beiden Schnittstellen (Freiraumkontur und dynamische Objektliste) ist es möglich, das von uns vorgeschlagene System der Umfeldwahrnehmung für aktuelle Fahrerassistenzsysteme zu verwenden.

Um unseren Ansatz zur Umfeldwahrnehmung zu validieren, evaluieren wir die Ergebnisse anhand eines offenen Datensatzes. Diese Auswertung zeigt, dass die vorgeschlagene Methode funktioniert und vergleicht die Ergebnisse aus verschiedenen Verarbeitungsstufen. Zusätzlich zeigen wir die Eignung unseres Ansatzes für den Einsatz auf einem Steuergerät mit einer Laufzeitmessung.

Abstract

Environment perception plays a crucial role in advanced driver assistance systems or autonomous driving applications. To overcome the limitations of single sensors, the data of multiple sensors has to be combined. This sensor data fusion is a critical part of a safety-critical system, which depends on the information of its direct surrounding environment.

In this work we propose a grid-based sensor data fusion approach, which allows sensor data fusion on a very low processing level. To enable the use of lidar and radar sensor data in this grid-based fusion, we present inverse sensor models for both technologies. Those inverse sensor models transform the raw sensor data in the grid representation used for data fusion.

One drawback of classic occupancy grid maps is, that they rely on the environment to be static. Therefore we show an extension to the classic occupancy grid mapping, which allows not only to estimate dynamic occupancy, but also to estimate the velocity of the single cells.

In addition we propose to take usage of an online generated orientation prior, which leads to a faster convergence of the used particles. The better velocity estimation of the particles allows a better dynamic estimation of the extracted objects and therefore leads to a better overall performance of the proposed method.

With this generated dynamic grid map it is possible to extract freespace, in terms of drivable area. Additionally we propose a method to extract (dynamic) objects out of the grid map. To improve the quality of the extracted objects, we use a standard Kalman filter based object tracking approach. Providing those two representations of the environment (freespace contour and dynamic object list) allows us to use our proposed system for current driver assistance systems.

To validate our proposed environment perception system we evaluate the results using an open data set. This evaluation shows that the proposed method is working and compares the results at different processing stages. Furthermore, we show that the proposed perception system is able to run on an automotive ECU by measuring the runtime performance.

Contents

1 Introduction

1.1 Motivation

Advanced driver assistance systems (ADAS), providing more comfort and safety for the driver and passengers of modern cars, play an important part in modern cars. With further research and development in the area of driver assistance systems and autonomous driving, those system will be part of even more vehicles in the future. The first companies are already testing self-driving cars on public roads [1, 2] and car manufacturers are selling so called *Autopilot* systems [3, 4]. Additionally, various different autonomous driving mini-buses, commonly known as *shuttles*, have been announced in Europe (e.g. [5, 6]) recently. Even with the already available systems and efforts brought into bringing those autonomous vehicles to the street, there are still a lot of challenges in this area.

One limiting factor for the autonomous driving is the environment perception ([7, 8]). The used sensors around the vehicle in combination with their detection and tracking algorithms are the interface of the vehicle to its surrounding world in terms of its input signals. All decision making processes, e.g. deciding whether to brake, rely on those input information. Therefore the environment perception needs to supply a precise representation of its surrounding environment. Not only the position and kinematics of the obstacles around the ego vehicle are important, but also knowing the areas where *no* obstacles are located in (so called freespace) plays a crucial role.

In order to provide the information in the required quality of today's and future driving functions, an information fusion of multiple sensors is required. With a single sensor the requirements in terms of range and full surround view are not feasible. Using multiple sensors the quality and robustness of the perception system can be improved. Furthermore, the advantages of different sensor technologies, e.g. the precision of lidar sensors can be combined with the high range, radial velocity measurement principle and weather robustness of radar sensors.

Another crucial point are the overall costs for the perception system. As car manufacturers are producing those system for mass markets and are continuing to bring ADAS also into smaller and cheaper vehicles, the environment perception should be able to scale with a different number or configuration of input sensors. Therefore it is important to design an environment perception in a scalable manner.

Most of the current driving functions rely on either an object list of the dynamic objects or a contour list of the surrounding freespace or both inputs. These two data formats have the huge advantage of describing the vehicle's surroundings in a very compressed way in terms of data size. This allows to transfer the information even over low bandwidth buses, e.g. CAN. The drawback on the other side is, that, with the compression, a simplification of the environment has to be made, which is less generic. One goal of the environment perception should be to be as simplified/abstract as required, but on the other hand as generic as possible.

1.2 State of the Art

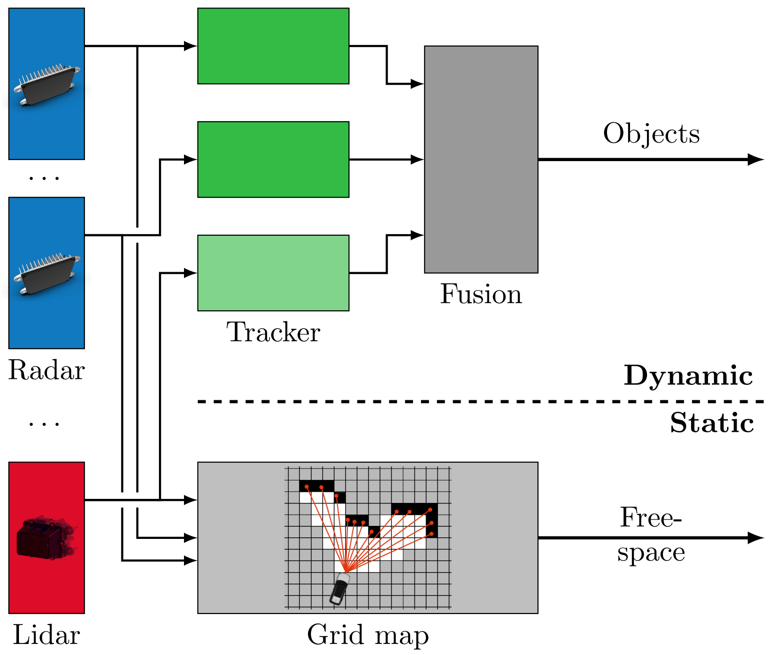


Figure 1.1: State of the art sensor data fusion

Most of today's environment perception systems work as shown in figure 1.1. The processing of the dynamic part of the surroundings is separated from the processing of the static part. The object tracking in the dynamic part is usually further divided into single sensor object tracking tasks, where those tracked object lists are fused into one object list containing all the dynamic objects (e.g. cars) of the surroundings later on. This approach has several advantages:

- Parallelization: Not only the static and dynamic processing can be run in parallel, but also each single object tracker is independent of the other ones. Often the object tracking algorithms are even executed on the sensors itself.
- Abstract interface: The interface for tracked objects, the output format of the object trackers, can be defined on quite an abstract level, allowing a sensor-type independent data format. Therefore replacing single sensors is possible, as long as the sensor specific object tracker is replaced with the sensor.
- Avoid unnecessary computation: In cases where only the dynamic or only the static environment is of interest (e.g. ACC or automated parking systems), the compu-

tation of the not required parts can be easily switched off to save computational power.

On the downside, this approach has two major disadvantages, the first one is the loss of information through processing. In each processing step information is lost, due to the processing itself, as well as to the simplification, which arises from the abstraction of the interface. For instance consider a vehicle, which is partly sensed by two sensors. Each of those two sensors is detecting a too small portion of the vehicle, so that none of the follow up trackers is generating a tracked object. Combining the raw measurements of both sensors on the other hand would lead to enough evidence for a (hypothetical) common tracker to establish a tracked object.

The second main drawback is the potential inconsistency of both representations. As the detection and tracking of dynamic objects is completely decoupled from the estimation of the static surroundings, both representations do not have to match each other perfectly. If one sensor detects an object, but another sensor is not detecting anything in the region of that object, it is possible that the static processing postulates freespace in that area, while the dynamic processing is estimating a moving car in the same area.

With the stated drawbacks it seems to be obvious, that the best choice for an environment perception system is the direct fusion of raw sensor data and then estimate dynamic objects and freespace only based on this fused data simultaneously. This approach indeed promises good results, but it comes with the cost of a very sensorset specific system with a high computational effort. Therefore we choose a grid-based fusion approach as middle course to fuse sensor data at a very low processing level, but still have a generic interface to support multiple sensorset configurations.

1.3 Related Work

The occupancy grid mapping approach was originally introduced by Moravec and Elfes in 1985 [9] for static environments. Various approaches have been developed to enable the basic principle of the occupancy grid mapping to also model dynamic environments. One approach is to use the occupancy grid map to model the static environment and additionally classify cells into *dynamic* and *static*. [10] and [11] use such classified dynamic cells as input for their object tracking algorithm, whereas their model of the static surroundings is based on the occupancy grid. In [12] the authors use two separate layers of grids. One grid is used for the estimation of occupancy probability, the other one for the dynamic probability estimation. With the information accumulated in both grid layers, object hypothesis are extracted and used in a particle-based object tracking algorithm afterwards.

Yuan et al. present a particle-based multiple model approach for a dynamic occupancy grid [13]. In their work they use particles to model the occupancy and dynamic state of grid cells in a two-stage process. The work described in [14] uses a binary Bayesian state to model the occupancy/freeness of each cell. The velocity of the cells is represented by an additional random variable, containing all possible velocities in a discrete manner.

Based on this work, Gindele et al. present an extension using prior map information [15] in order to improve the map prediction, hence improving the overall performance. [16] uses a Bayesian occupancy grid for occupancy estimation, but models the velocity distribution of the cells using particles. This combined approach allows comparable results with much lower requirements on memory consumption and increases runtime performance. Tanzmeister [17], Steyer [18] et al. propose using a grid map based on the Dempster-Shafer theory to model the states of cells in combination with particles to model the dynamics of the surroundings. This work is the basis of the dynamic grid described in chapter 3.2, e.g. we carry over the same set of hypothesis, the static prediction formula and the linear prediction of the particles itself as well as the color scheme for the mass visualization.

A very efficient way to model inverse sensor models for lidar and radar sensors is to calculate the occupancy estimates in a polar grid first and transform it into a Cartesian one afterwards. Homm et al. present such an approach for the inverse sensor model taking advantage of the parallel processing capabilities of a GPU [19]. In [20] Werber et al. present radar specific adaptations for inverse sensor models, compared to lidar models. One key difference is that radar sensors can also detect objects located behind other objects, which would be occluded for lidar sensors. This difference has to be considered in the design of the appropriate inverse sensor model. The work described in [21] argues, that regions without any detections, have to be considered as freespace. Especially, this is the case on roads or parking lots without traffic, where the road itself is not providing any detections so that no freespace assumption can be generated in traditional inverse radar sensor models.

In [22] Lindenmaier et al. describe different association methods used for a track to track fusion approach. They present a point to point association method as well as one using the 2-D extension of objects. Furthermore they describe a frame based birth model, which is very similar to the one we are using. Diehl et al. use a dynamic occupancy grid mapping approach for fusing only on radar data [23]. The objects are extracted purely based on the static and dynamic estimated cell masses. With their work they demonstrate, that the grid-based fusion approach works with using only radar sensors and is sufficient for object estimation. In [24] Steyer et al. use the dynamic occupancy grid of [18] to extract objects based on the dynamic occupancy estimation of cells. For clustering of the dynamic cells, a combination of a density-based and connectivity-based method is used. Additionally the particles get labeled with the track id for better clustering in future time frames. In [25] this concept is extended by using an unscented Kalman filter (UKF) to track the extracted objects and to address the objects shape estimation in terms of length and width of the corresponding bounding box.

The dissertation published 2021 by Steyer [26] describes a very similar approach as ours: the grid-based sensor fusion, the dynamic grid estimation using predict/update cycles, as well as an additional object tracking in the end. Although we use a very similar approach, the single steps differs in detail. Steyer is using camera sensors as additional input to radar and lidar sensors, but is not providing detailed description of the implemented sensor models. Furthermore, we extended the grid prediction by using an online generated orientation prior and prefer faster approaches in favor of ECU compatibility.

1.4 Contribution & Goals

Goals The main goal of this work is to develop an environment perception approach, which provides all relevant information of the vehicle's surroundings to the (autonomous) driving function. The proposed method has to be capable of fusing multiple sensors using a very generic abstraction layer. Furthermore the approach has to be scalable regarding the number of used sensors.

The proposed method should also be able to operate as a replacement of existing perception approaches, hence it has to provide the traditional interfaces of object and contour list for dynamic objects and freespace information.

Finally, the implementation of the proposed algorithm in this work has to be fast enough to run on an automotive ECU, fusing the data of multiple sensors.

Contribution In this thesis we present a straight forward derivation of a dynamic Bayesian occupancy grid, based on the static one. Furthermore, we justify the advantages of the particle-based Dempster-Shafer approach over the Bayesian one and extend it using an online generated orientation prior. Additionally, we introduce a derivation for inverse lidar and radar sensor models and how they can be fused together. The extraction of the freespace contour and the object extraction with additional tracking closes the gap between our proposed grid-based environment modeling and the traditional object / freespace model used in current systems. We propose to use common image processing methods, a simple thresholding and a connected component algorithm to extract object hypothesis from the dynamic grid.

Publications and Supervised Theses Parts of this thesis have been published as peer-reviewed conference papers:

- In [27], we present a straight forward derivation of the prediction and update rule of the dynamic occupancy grid, based on the well-known static Bayesian occupancy grid mapping. The content of this publication is summarized in section 3.1.

- In [28], we present the online generation of an orientation prior of newly initialized particles. This orientation prior is used during the particle re-sampling (3.2.2.2), whereas the generation of the orientation prior is described in 3.2.3.

Furthermore the author wants to thank Youssef Jazi for his Master Thesis [29], in which object tracking based on a dynamic grid as input is investigated. This work lays the foundation of the object tracking described in chapter 5.

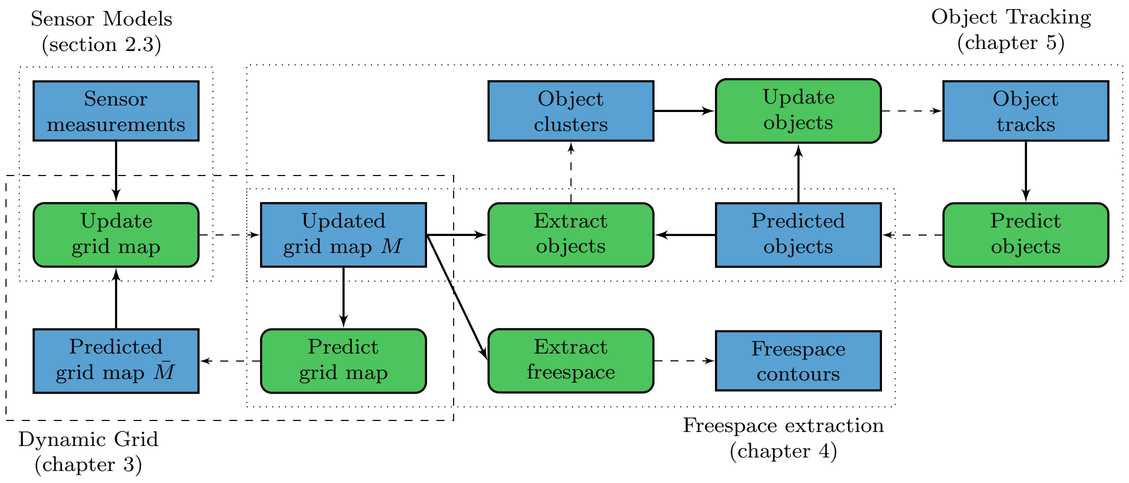


Figure 1.2: System overview

1.5 Structure of This Work

In chapter 2 we give a summary of the static occupancy grid map approach, using the classic Bayesian approach, as well as using a Dempster-Shafer approach. Additionally, the inverse sensor models (2.3) are presented in this chapter. Those sensor models work as input data for the grid mapping, as shown in the upper left corner of figure 1.2.

Chapter 3 describes the extension of the static occupancy grid to a dynamic one and introduces the online orientation prior. Furthermore, we discuss the advantages of using a particle-based Dempster-Shafer dynamic grid here. This part is illustrated in the lower left part of figure 1.2.

The freespace extraction, shown in the lower right part of figure 1.2 is described in chapter 4, whereas the object extraction and tracking part (upper right part of the figure) is described in chapter 5. Both chapters are using the output of the dynamic grid as their inputs to provide the required information for the traditional object and contour list interface to describe dynamic objects and freespace around the vehicle.

An evaluation and validation of our approach is presented in chapter 6. In chapter 7 we conclude this work and give an outlook for further research.

2 Static Occupancy Grid

The static occupancy grid is the standard approach for representing a static environment in today's driver assistance and autonomous driving systems. The basic idea of grid-based representations is dividing the environment in small cells and estimating the state of the environment for every single cell individually.
This chapter presents the basics of such a grid-based approach using Bayes theory and introduces an alternative using Dempster-Shafer theory.

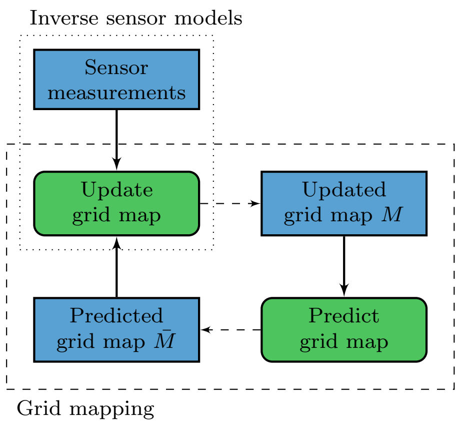


Figure 2.1: Overview of static occupancy grid mapping

Figure 2.1 gives a small overview of the static occupancy grid mapping and the role of the inverse sensor models. In case of the static occupancy grid the *Predict grid map* function is usually an identity function, which predicts the cells as they are and is therefore not handled in this section. In some implementations the predict function can be used to translate/rotate the grid according to the ego vehicle movement or implement features like decay of outdated information.

2.1 Bayes

The main idea of the Bayesian occupancy grid is to divide the hard problem of estimating the occupancy of surrounding environment into estimating the occupancy of single cells. Thrun et al. [30, pp. 85–90] give a great introduction into the Bayes filtering process in the grid map.

The traditional occupancy grid for static environment uses two main assumptions, which are discussed in the following section.

Binary Cell State Each cell is assumed to be either occupied or empty (free). This assumption directly motivates the usage of a binary random variable to represent a cell state: $O \in \{\text{occ}, \text{emp}\}$. With only two possible states, each state is the complementary of the other leading to: $P(O = \text{occ}) = 1 - P(O = \text{emp})$.

Independence of the Cells The overall goal of the static grid map is to estimate the most likely occupancy map over all cells $P(O)$. For computational reasons the cells are assumed to be independent of each other: $P(O) = \prod_c P(O_c)$. Additionally this independence assumption allows a very efficient way of estimating the occupancy probability of every cell in a parallel way.

Derivation of Update Rule With those two assumption the goal of the occupancy grid mapping can be formulated as to give a probability of how likely a cell is being occupied (or free) after a given set of measurements $Z^{1:t}$. We call this $P(O_c|Z^{1:t})$.

Using Bayes rule $P(O_c|Z^{1:t})$ can be calculated as:

$$\begin{aligned}
P(O_c|Z^{1:t}) &= P(O_c|Z^t, Z^{1:t-1}) \\
&= \frac{P(Z^t|O_c, Z^{1:t-1})\ P(O_c, Z^{1:t-1})}{P(Z^t, Z^{1:t-1})} \\
&= \frac{P(Z^t|O_c, Z^{1:t-1})\ P(O_c|Z^{1:t-1})\ P(Z^{1:t-1})}{P(Z^t|Z^{1:t-1})\ P(Z^{1:t-1})} \\
&= \frac{P(Z^t|O_c, Z^{1:t-1})\ P(O_c|Z^{1:t-1})}{P(Z^t|Z^{1:t-1})}.
\end{aligned}$$

Assuming that O_c contains all relevant information of the measurements, which have been used for updating so far, $P(Z^t|O_c, Z^{1:t-1})$ can be substituted with $P(Z^t|O_c)$ leading to following simplification:

$$P(O_c|Z^{1:t}) = \frac{P(Z^t|O_c)\ P(O_c|Z^{1:t-1})}{P(Z^t|Z^{1:t-1})}.$$

Applying Bayes rule again, leads to:

$$P(O_c|Z^{1:t}) = \frac{P(O_c|Z^t)\ P(Z^t)\ P(O_c|Z^{1:t-1})}{P(O_c)\ P(Z^t|Z^{1:t-1})}.$$

Exploiting the nature of a binary random variable the odds-ratio representation $R(X)$ is equivalent to the representation using probability values $P(X)$:

$$\begin{aligned}
\frac{P(X)}{1 - P(X)} &= R(X) \\
P(X) &= \frac{R(X)}{1 + R(X)}
\end{aligned}$$

The odds-ratio representation allows further simplification of the update formula:

$$\begin{aligned}
\frac{P(O_c|Z^{1:t})}{1-P(O_c|Z^{1:t})} &= \frac{P(O_c|Z^{1:t})}{P(\neg O_c|Z^{1:t})} \\
&= \frac{\frac{P(O_c|Z^t)\ \cancel{P(Z^t)}\ P(O_c|Z^{1:t-1})}{P(O_c)\ \cancel{P(Z^t|Z^{1:t-1})}}}{\frac{P(\neg O_c|Z^t)\ \cancel{P(Z^t)}\ P(\neg O_c|Z^{1:t-1})}{P(\neg O_c)\ \cancel{P(Z^t|Z^{1:t-1})}}} \\
&= \frac{P(O_c|Z^t)\ P(O_c|Z^{1:t-1})\ P(\neg O_c)}{P(\neg O_c|Z^t)\ P(\neg O_c|Z^{1:t-1})\ P(O_c)} \\
&= \underbrace{\frac{P(O_c|Z^t)}{1-P(O_c|Z^t)}}_{\text{Update term}} \cdot \underbrace{\frac{P(O_c|Z^{1:t-1})}{1-P(O_c|Z^{1:t-1})}}_{\text{Recursive term}} \cdot \underbrace{\frac{1-P(O_c)}{P(O_c)}}_{\text{Prior}} .
\end{aligned} \tag{2.1}$$

The derivation of the update formula (eq. 2.1) shows, that estimation of the occupancy probability of a cell c ($P(O_c|Z^{1:t})$) only depends on

- an update term $P(O_c|Z^t)$, which denotes the conditional probability of being occupied given only the current measurement Z^t (more details in section 2.3),
- the previous estimation $P(O_c|Z^{1:t-1})$, after all previous measurements apart from the latest one (Z^t), and
- a prior probability of being occupied given no measurement data $P(O_c)$.

Most commonly the prior occupancy probability $P(O_c)$ is set to 50%, meaning that the probability of being occupied equals the probability of being free. This additional assumption eliminates the third factor in equation 2.1 and makes the update formula even more handy to use:

$$\frac{P(O_c|Z^{1:t})}{1-P(O_c|Z^{1:t})} = \frac{P(O_c|Z^t)}{1-P(O_c|Z^t)} \cdot \frac{P(O_c|Z^{1:t-1})}{1-P(O_c|Z^{1:t-1})}. \tag{2.2}$$

For numerical and performance reasons the logarithmic version of eq. 2.2 is commonly used:

$$\begin{aligned}
l(x) &= \ln\left(\frac{P(x)}{1-P(x)}\right) \\
l(O_c|Z^{1:t}) &= l(O_c|Z^t) + l(O_c|Z^{1:t-1})
\end{aligned} \tag{2.3}$$

Furthermore, minimal and maximal thresholds for $l(O_c|Z^{1:t})$ are used to avoid numeric instabilities and final (not updatable) probability values:

$$\hat{l}(O_c|Z^{1:t}) = \begin{cases} l_{\max} & , l(O_c|Z^{1:t}) > l_{\max} \\ l_{\min} & , l(O_c|Z^{1:t}) < l_{\min} \\ l(O_c|Z^{1:t}) & , \text{otherwise} \end{cases}$$

Example We take a look at two examples. In the first case we estimate a cell c two times as occupied with a probability of 60% (table 2.1). In the second case we estimate a cell c with an occupancy probability of 30% (so more likely being free than occupied) and at the second time $t = 2$ with an occupancy probability of 84% (table 2.2).

t	$P(O_c \mid Z^t)$	$l(O_c \mid Z^t)$	$P(O_c \mid Z^{1:t})$	$l(O_c \mid Z^{1:t})$
0	-	-	0.5	0
1	0.60	0.405	0.60	0.405
2	0.60	0.405	0.69	0.811

Table 2.1: Example 1 for Bayesian update rule

t	$P(O_c \mid Z^t)$	$l(O_c \mid Z^t)$	$P(O_c \mid Z^{1:t})$	$l(O_c \mid Z^{1:t})$
0	-	-	0.5	0
1	0.30	-0.847	0.30	-0.847
2	0.84	1.658	0.69	0.811

Table 2.2: Example 2 for Bayesian update rule

As shown in the tables above, in both cases the occupancy probability estimation after the two timestamps is at 69%. Using the Bayesian approach, it is not possible to differentiate between the both cases in the end, although it makes a huge difference if a cell as been sensed two times as occupied or if it has been free and, due to a moving object, has been occupied later.

2.2 Dempster Shafer

The *Dempster-Shafer theory*, or also called *theory of belief functions* or *evidence theory*, can be seen as a generalization of Bayesian theory. As in the Bayesian case described above, the independence of the cells is assumed, but instead of a binary random variable, the hypotheses are modeled with values representing the *degree of belief* (referred as *mass*). Furthermore, the Bayes update rule is replaced with an adapted combination rule, which will be described in this section.

Hypotheses There exist two distinct possibilities for a cell's state for the static grid: either the cell is empty/free (F) or the cell is (static) occupied (S). This set of possibilities

$$\Theta = \{F, S\} \tag{2.4}$$

defines the following frame of discernment (power set):

$$2^\Theta = \{\emptyset, \{F\}, \{S\}, \{F, S\}\}\,. \tag{2.5}$$

For better readability obvious brackets and commas will be skipped from here on, leading to following simplified notation:

$$2^\Theta = \{\emptyset, F, S, FS\}\,. \tag{2.6}$$

Every element in the frame of discernment is called a hypothesis. The $\emptyset$ describes the hypothesis that none of the assumed possibilities is true, in this case the cell is neither empty nor occupied. The FS hypothesis describes that a cell is either free or occupied.

Degree of Belief For every hypothesis $\theta \in 2^\Theta$ a mass function $m(\theta)$ is defined. The mass function can be seen as subjective probability and is used to calculate the belief and plausibility value of a hypothesis. Like probability values, the mass functions also have to follow two constraints:

$$\begin{aligned} m(\theta) &\geq 0, \quad \forall \theta \in 2^\Theta \\ \sum_{\theta \in 2^\Theta} m(\theta) &= 1. \end{aligned}$$

With those constraints the mass function of the Θ hypothesis is not necessarily modeled explicitly, but can be calculated form the remaining mass functions.

Belief and Plausibility The belief of a hypothesis θ is defined as the sum over all masses of subsets of θ:

$$bel(\theta) = \sum_{\gamma | \gamma \subseteq \theta} m(\gamma), \tag{2.7}$$

and is a lower bound for the probability of θ. The plausibility of a hypothesis θ is the sum of all masses of sets that intersect θ:

$$pl(\theta) = \sum_{\gamma | \gamma \cap \theta \neq \emptyset} m(\gamma), \tag{2.8}$$

and is an upper bound for the probability of θ. In short: the probability of a hypothesis $P(\theta)$ is bound by the belief and plausibility as:

$$bel(\theta) \leq P(\theta) \leq pl(\theta).$$

For a complete and distinct set of possibilities the belief and plausibility of the empty hypothesis are 0, whereas for the universal hypothesis they have to be equal to 1. This concludes that the mass of the empty hypothesis has to be 0.

$$\begin{aligned} bel(\emptyset) &= pl(\emptyset) = 0 \\ \Rightarrow m(\emptyset) &= 0 \\ bel(\Theta) &= pl(\Theta) = 1 \end{aligned}$$

Combination Rule Combining two sets of masses m_1 and m_2 plays an important role in the update process of a Dempster-Shafer grid map. The update process is modeled as combination of the previous grid map with the current sensor measurement. The combination rule is defined as:

$$\begin{aligned} m_{1,2}(\emptyset) &= 0 && (2.9)\\ \widehat{m}_{1,2}(\theta \neq \emptyset) &= (m_1 \oplus m_2)(\theta) \\ &= \sum_{\theta_1 \cap \theta_2 = \theta} m_1(\theta_1) \cdot m_2(\theta_2) && (2.10)\\ \zeta_{1,2} &= \sum_{\theta_1 \cap \theta_2 = \emptyset} m_1(\theta_1) \cdot m_2(\theta_2), && (2.11) \end{aligned}$$

with the conflict term $\zeta_{1,2}$. The standard combination rule distributes the conflict mass over all combined masses:

$$m_{1,2}(\theta) = \frac{\widehat{m}_{1,2}(\theta)}{1 - \zeta_{1,2}}. \tag{2.12}$$

Other possible ways to deal with the conflict term are either the addition to Θ:

$$\begin{aligned} m_{1,2}(\theta \neq \Theta) &= \widehat{m}_{1,2}(\theta) \\ m_{1,2}(\Theta) &= \widehat{m}_{1,2}(\Theta) + \zeta_{1,2}, && (2.13) \end{aligned}$$

or the distribution based on *expert knowledge*, which will be utilized in chapter 3.2.

Example We use the same two examples as in 2.1. In the first example (table 2.3) the cell is both times estimated as occupied with a low “probability”. In the second example (table 2.4) the cell is first being estimated as free and then estimated as occupied. Both tables show the current estimation $z(\theta)$, the grid map result $m(\theta)$ and the lower and upper bound of the corresponding probabilities (belief & plausibility), with assigning the conflict mass ζ to FS.

t	$z(F)$	$z(S)$	$m(F)$	$m(S)$	$P(F)$	$P(S)$
0	-	-	0	0	$[0.00; 1.00]$	$[0.00; 1.00]$
1	0	0.20	0	0.20	$[0.00; 0.80]$	$[0.20; 1.00]$
2	0	0.20	0	0.36	$[0.00; 0.64]$	$[0.36; 1.00]$

Table 2.3: Example 1 for Dempster Shafer update rule

Using the Dempster Shafer theory, the different estimation of the two measurements is reflected in the final grid map. This effect, in combination with a custom conflict assignment, plays a crucial role in section 3.2.

t	$z(F)$	$z(S)$	$m(F)$	$m(S)$	$P(F)$	$P(S)$
0	-	-	0	0	$[0.00; 1.00]$	$[0.00; 1.00]$
1	0.40	0	0.40	0	$[0.40; 1.00]$	$[0.00; 0.60]$
2	0	0.68	0.13	0.41	$[0.13; 0.59]$	$[0.41; 0.87]$

Table 2.4: Example 2 for Dempster Shafer update rule

2.3 Sensor Models

Sensor models, or inverse sensor models, play a crucial role in the occupancy grid mapping approach, since those models transform the rather abstract *sensor data* into a grid representation. This grid representation can then be used to fuse multiple sensors (e.g. figure 2.2) or to update the actual grid map in a generalized sensor-independent manner.

Sections 2.3.2 and 2.3.3 describe the used sensor models for lidar and radar sensors used in this work, whereas section 2.3.4 explains the fusion of multiple sensor grids into one final sensor grid.

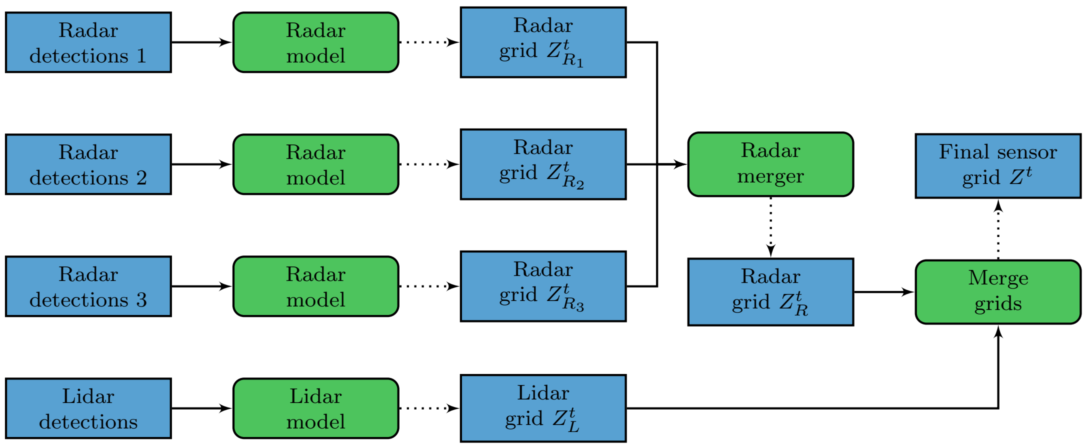


Figure 2.2: Overview of a system with three radar and one lidar sensor.

2.3.1 Polar Sensor Model

Both sensor technologies used throughout this work, namely lidar (2.3.2) and radar (2.3.3) sensors, follow a polar measurement principle. The sensors are measuring a distance r for a given (horizontal) angle or angular segment φ. For sensors capable of measuring 3-D points an additional angular parameter, the evaluation angle ϑ, is measured.

Although the fusion and accumulation of the occupancy grid is done using Cartesian coordinates, performing the preprocessing in a polar grid has two main advantages, caused by the mentioned nature of measurement. The first advantage is, that often the uncertainties of measurement data are given in the measurement system, e.g. maximal error in measured range or azimuth angle. Processing of the data using the same coordinate frame allows considering such conditions in a much easier and straight forward manner.

Secondly, the occupancy probability of the inverse sensor models at a given position relies mainly on detections under the respective horizontal angle of this position. Therefore only a few or none dependencies between detections of different angles exists, allowing a parallelizable implementation of the inverse sensor models using a polar grid.

The two inverse sensor models discussed in the following section are using the same algorithmic structure:

1. Preprocessing of the polar grid
2. Detection counting
3. Occupancy probability calculation
4. Transformation into Cartesian cells

Preprocessing The first step is a preprocessing or cleanup of the sensor's polar grid. In this step all information, which does not rely on any sensor measurement is calculated and stored in the polar cells. The calculation of the expected freespace probability in absence of any detection data is performed here.

Detection Counting Afterwards, there is a detection counting step, where the polar grid is populated with values like, how many detections are located in each polar cell, or how many detections are located behind the current cell, albeit in the same angular bin. The calculations are done using the respective detection lists. This is the only step, in which the detection lists are accessed. All relevant information for the subsequent processing steps are included as information in the polar grid after this step.

Occupancy Probability Calculation After the processing of the detection list itself, the occupancy probability of each polar cell is calculated based on the information of the first preprocessing step. This results in an occupancy probability value for each cell in the polar grid.

Cartesian Transformation Finally, the occupancy probabilities of the polar cells are transformed into a Cartesian grid for updating the grid map itself, figure 2.4 shows an example. As there exist no unambiguous assignment of a Cartesian and polar cell, a bi-linear interpolation or maximum function is used for the transformation. Given a Cartesian cell c with its center at (x, y) the corresponding polar coordinates (r, φ) are

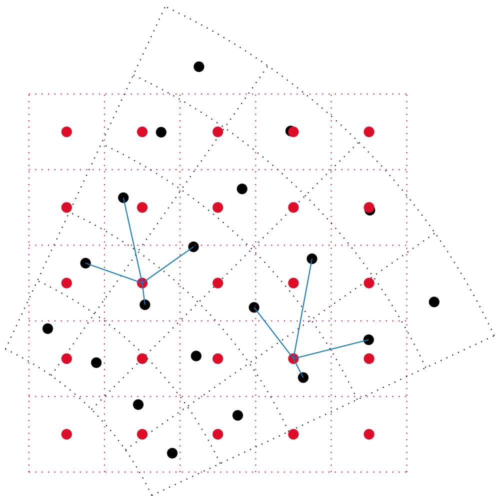


Figure 2.3: Cartesian grid (red) overlay on a polar grid (black). Two Cartesian center points are connected with their closest polar center points (p_0, p_1, p_2, p_3) using blue lines.

defined as:

$$r = \sqrt{x^2 + y^2} \tag{2.14}$$

$$\varphi = \begin{cases} +\arccos\frac{x}{r} & , y \geq 0 \\ -\arccos\frac{x}{r} & , y < 0 \end{cases}. \tag{2.15}$$

With the cell size $(\Delta_r, \Delta_\varphi)$ of the polar grid, the closest polar cells are (see fig. 2.3):

$$p_0 = \left(\left\lfloor \frac{r}{\Delta_r} \right\rfloor, \left\lfloor \frac{\varphi}{\Delta_\varphi} \right\rfloor \right) \tag{2.16}$$

$$p_1 = p_0 + (1, 0) \tag{2.17}$$

$$p_2 = p_0 + (0, 1) \tag{2.18}$$

$$p_3 = p_0 + (1, 1). \tag{2.19}$$

Using the bilinear interpolation method the occupancy probability of Cartesian cell c can be calculated as:

$$P(O_c|Z^t) = \eta \cdot \big((r_1 - r) \cdot (\varphi_1 - \varphi) \cdot P(O_{p_0}|Z^t) \tag{2.20}$$

$$+(r - r_0) \cdot (\varphi_1 - \varphi) \cdot P(O_{p_1}|Z^t) \tag{2.21}$$

$$+(r_1 - r) \cdot (\varphi - \varphi_0) \cdot P(O_{p_2}|Z^t) \tag{2.22}$$

$$+(r - r_0) \cdot (\varphi - \varphi_0) \cdot P(O_{p_3}|Z^t)\big). \tag{2.23}$$

In case of the max function the transformed occupancy probability is given by:

$$P(O_c|Z^t) = \max\left(P(O_{p_0}|Z^t), P(O_{p_1}|Z^t), P(O_{p_2}|Z^t), P(O_{p_3}|Z^t)\right). \tag{2.24}$$

The bilinear interpolation method provides good results in cases, where a certain uncertainty of the measurement position is included in the polar grid, e.g. if a single measurement affects more than one single cell. Whereas the max function method is superior if one detection can be located in exactly one polar cell and is not influencing the neighboring ones.

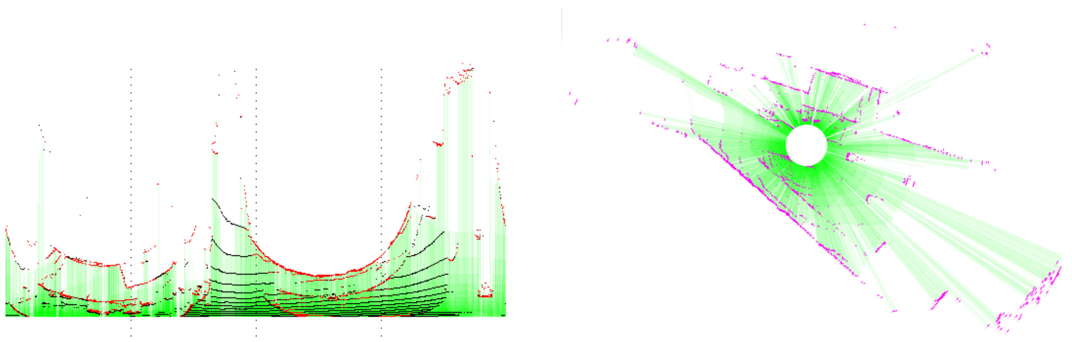

(a) Polar grid **(b)** Cartesian grid

Figure 2.4: Example of a transformation from polar to Cartesian grid. Green cells represent freespace, whereas purple cells represent occupancy. Black cells represent detections classified as ground and are shown only in the polar coordinate system for orientation.

2.3.2 Lidar Sensor Model

Lidar sensors normally provide a very regular scan pattern, e.g. one distance measurement every 0.2° horizontal and 2° vertical (e.g. [31]). Furthermore, lidar sensors are currently the most accurate sensors in terms of position. Therefore they are perfectly suited for occupancy grids. This subsection defines the data structure of the lidar detections and a novel approach for an inverse lidar sensor model based on the discussed polar sensor model (2.3.1). The angular resolution of the used polar grid is chosen to match the angular resolution of the lidar sensor or, to favor runtime performance over accuracy, to be an integral multiple of it.

2.3.2.1 Input Data

For the proposed inverse lidar sensor model, the lidar sensor is expected to work in a fixed scanning pattern. Especially every Δ_φ (horizontal) and Δ_ϑ (vertical) angle a distance measurement is expected. With this assumption the lidar scan can also be interpreted as image, where every row corresponds to a vertical measurement angle and every column corresponds to a horizontal measurement angle. The pixel intensity/color would be the measured distance.

With this regularity of scan pattern, every detection (or pixel of the hypothetical image) has to provide the following information per scan point:

- azimuth angle φ,

- elevation angle ϑ,
- measured distance r,
- flag indicating if a distance could be measured for this scan point,
- classification whether this detection is classified as ground point or not (see [32] for a lidar ground-point classifier).

Additional information such as the intensity of the measured reflection can be used as filter criteria, but are not considered in this section.

2.3.2.2 Occupancy Estimation

The novel inverse lidar sensor model presented in this work follows the steps mentioned in 2.3.1 and is described in this section.

Preprocessing For the inverse lidar sensor model no preprocessing of the polar grid, except the reset of the polar cells, is necessary. The information stored per cell c is listed in the table below.

Name	Description	Start value
n_{object}	Number of rays hitting an object in this cell	0
n_{ground}	Number of rays hitting the ground in this cell	0
$n_{\text{traversing}}$	Number of rays traversing this cell without hitting anything in this cell	0
$z_{\text{ray,min}}$	Minimum z coordinate of a traversing ray	z_{max}
$z_{\text{ray,max}}$	Maximum z coordinate of a traversing ray	z_{min}
$P(O_c\|Z^t)$	Estimated occupancy probability for this cell	0.5

Table 2.5: Information stored in each cell c

Detection Counting For every detection, a ray originating at the sensor with the length of the measured distance or the sensor's maximum range is traversed. For every traversed polar cell the value of $n_{\text{traversing}}$ is incremented and the values of $z_{\text{ray,min}}$ and $z_{\text{ray,max}}$ are updated with the respective height of the ray at the given location. If the cell of the measured distance is reached n_{object} or n_{ground} is incremented instead, depending on the classification of the detection. $z_{\text{ray,min}}$ and $z_{\text{ray,max}}$ are not updated in case of the last cell. The pseudo code for the detection counting is shown in algorithm 1.

Figure 2.5 shows an example of a single angular bin from the polar grid. The sensor is placed on the left and an example object is placed on the right (shown as gray box). Three rays are reaching the ground plane in front of the object (shown in green), two

Algorithm 1: Inserting a ray into the polar grid

```
for $r \leftarrow 0$ to $r_{max}$ do
    $z \leftarrow$ GetCurrentRayZ()
    if $z < z_{min}$ or $z > z_{max}$ then
        break;
    end

    if AtTargetCell() and ClassifiedAsGround() then
        $n_{\text{ground}} \leftarrow n_{\text{ground}} + 1$
    else if AtTargetCell() and ClassifiedAsObject() then
        $n_{\text{object}} \leftarrow n_{\text{object}} + 1$
    else
        $n_{\text{traversing}} \leftarrow n_{\text{traversing}} + 1$
        $z_{\text{ray,min}} \leftarrow \min(z_{\text{ray,min}}, z)$
        $z_{\text{ray,max}} \leftarrow \max(z_{\text{ray,max}}, z)$
    end
end
```

rays are hitting the object (shown in red) and another three rays are not hitting anything (shown in blue). In table 2.6 the resulting numbers of the detection counting for the three cells c_1, c_2 and c_3 (as marked in the figure) are presented.

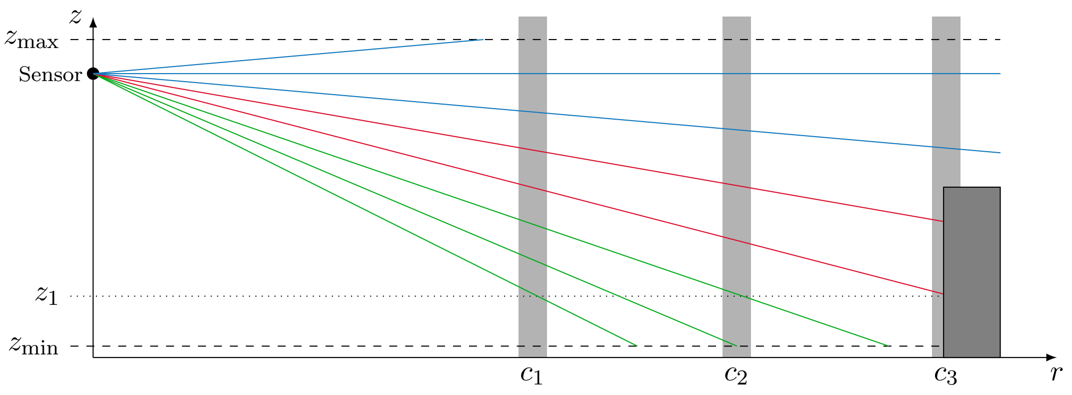


Figure 2.5: Rays hitting the ground, an obstacle, or nothing for one horizontal angle φ.

Occupancy Probability Calculation The occupancy probability is calculated purely on the information in the polar grid, the detection list itself is not used anymore. In the case of at least one object detection counted in cell c ($n_{\text{object}} > 0$) the cell is considered as occupied and the number of traversing or ground-classified counts $(n_{\text{ground}}, n_{\text{traversing}})$

Cell	n_{object}	n_{ground}	$n_{\text{traversing}}$	$z_{\text{ray, min}}$	$z_{\text{ray, max}}$
c_1	0	0	7	z_1	z_{Sensor}
c_2	0	1	5	z_{min}	z_{Sensor}
c_3	2	0	2	z_1	z_{Sensor}

Table 2.6: Detection counting result of three example cells, illustrated in figure 2.5

is not taken into account. The occupancy probability of the cell is estimated as:

$$\begin{aligned} P(O_c|Z^t) &= \frac{1}{2} + \frac{1 - (p_{\text{false-positive}})^{n_{\text{object}}}}{2} \\ &= 1 - \frac{(p_{\text{false-positive}})^{n_{\text{object}}}}{2}, \end{aligned} \tag{2.25}$$

or alternatively using the Dempster-Shafer framework:

$$\begin{aligned} Z_c(SD) &= 1 - (p_{\text{false-positive}})^{n_{\text{object}}} \\ Z_c(FSD) &= 1 - Z_c(SD), \end{aligned} \tag{2.26}$$

with $p_{\text{false-positive}}$ being the probability of a false-positive detection. In the reference implementation a false-positive probability of 5% is used.

If no object is detected in the current cell, the cell can either be considered as free or as unknown, figure 2.6 illustrates an example. Caused by the occlusion of a closer object a possible object (marked with the "?" in the figure) cannot be detected by the sensor. To avoid freespace estimation in such blind-spot areas, no freespace estimation is taking place behind objects, unless a ground detection has been counted. The occupancy probability estimation for the blind-spot areas is therefore:

$$\begin{aligned} P(O_c|Z^t) &= \frac{1}{2} \\ Z_c(FSD) &= 1 \end{aligned} \tag{2.27}$$

In the cases, where a ground detection, but no object is counted, freespace is more likely than occupancy and therefore freespace is estimated. Freespace is defined as the absence of some reference object with a fixed horizontal width w_{ref} and a fixed height h_{ref} (e.g. $w_{\text{ref}} = h_{\text{ref}} = 0.1\,\text{m}$). All potential smaller objects are not considered as false-negatives. To estimate the probability of being free, we have to answer the question of how likely the reference object should have been measured. To answer that question we calculate the covered (vertical) area A_{covered} of the cell, as well as maximal possible

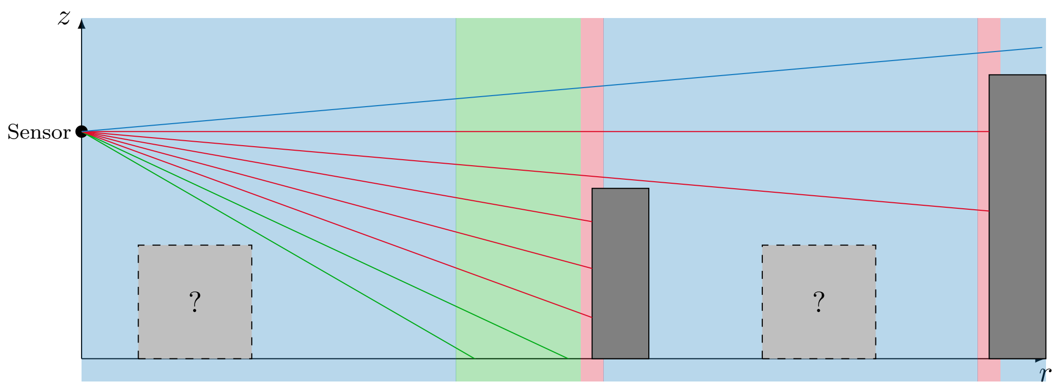


Figure 2.6: Illustrating possible objects in blind spot areas and occupied, free and unknown estimated areas.

covered area $A_{\text{covered,max}}$:

$$\begin{aligned} w_{\text{covered}} &= \Delta_{\varphi} \cdot r_c & (2.28)\\ h_{\text{covered}} &= z_{\text{ray,max}} - z_{\text{ray,min}} & (2.29)\\ h_{\text{covered,max}} &= z_{\max} - z_{\min} & (2.30)\\ A_{\text{covered}} &= w_{\text{covered}} \cdot h_{\text{covered}} & (2.31)\\ A_{\text{covered,max}} &= w_{\text{covered}} \cdot h_{\text{covered,max}}, & (2.32) \end{aligned}$$

with r_c being the range of the cell and $z_{\text{ray, min}}, z_{\text{ray, max}}$ being the cell's result of the *Detection Counting* step. With those results, the average area of every traversing ray is calculated:

$$\bar{A}_{\text{ray}} = \frac{A_{\text{covered}}}{n_{\text{ground}} + n_{\text{traversing}}}. \qquad (2.33)$$

Additionally the area of the reference object, covered by this cell, has to be limited by the covered width and height of that cell:

$$A_{\text{ref}} = \min\left(w_{\text{covered}}, w_{\text{ref}}\right) \cdot \min\left(h_{\text{covered}}, h_{\text{ref}}\right). \qquad (2.34)$$

Assuming a uniform distribution of the traversing rays through the covered area, we can now calculate the detection probability of the reference object in the given cell:

$$p_{\text{detect}} = \frac{A_{\text{covered}}}{A_{\text{covered,max}}} \cdot \max\left(\frac{A_{\text{ref}}}{\bar{A}_{\text{ray}}}, 1\right). \qquad (2.35)$$

Putting together the detection probability of the reference object and the fact, that no object has been detected in this cell leads to following occupancy probability estimation:

$$\begin{aligned} P(O_c|Z^t) &= \frac{1}{2} - \frac{p_{\text{detect}}}{2}\\ Z_c(F) &= p_{\text{detect}}\\ Z_c(FSD) &= 1 - Z_c(F). & (2.36) \end{aligned}$$

Cartesian Transformation The last processing step is the transformation of the polar grid into a Cartesian one for incorporation into the actual grid map. As our inverse lidar model does not include any angular or range uncertainties for the lidar detections, i.e. every detection influences exactly one angular bin, we prefer the max operator instead of the interpolation for the transformation into Cartesian (see 2.3.1). Using interpolation would cause some structures, like fences, to be blurred out, as some rays will traverse such structures. In the domain of autonomous driving, a solid occupied structure in the grid is preferred.

2.3.3 Radar Sensor Model

Apart from cameras, radar is the most commonly used sensor technology for environment perception today. The two most important advantages of radar sensors are their ability to measure the radial velocity, as well as their robustness in terms of weather conditions. This subsection lists the data structure of radar detections and shows the novel inverse radar sensor model implemented in this work.

2.3.3.1 Input Data

For the proposed inverse radar sensor model only a two-dimensional radar sensor is assumed, meaning that no elevation angle or height information is provided. If the sensor provides such an information, the measured distance has to be projected on the 2-D x-/y-plane. Therefore the proposed inverse sensor needs the following information per radar detection:

- azimuth angle φ
- measured distance r
- measured radial velocity v_{rad}

Additional information, such as the signal to noise ratio (SNR), or radar cross section (RCS), can be used for filtering or further enhancement of the proposed inverse radar sensor model, but are not part of this work.

2.3.3.2 Occupancy Estimation

Our proposed inverse radar sensor model is divided into four processing steps, following the convention mentioned in 2.3.1. First the default freespace probability in case of the absence of any detection is calculated, followed by some sort of ray counting. The final occupancy calculation based on the counting result and the transformation into Cartesian coordinates finalizes the proposed sensor model.

Preprocessing Similar to the inverse lidar sensor model (2.3.2) we define freespace as the absence of a reference object. Instead of defining the reference object with a fixed dimension, a reflecting surface σ_{ref} is used to define the reference object. The received power P_r at the radar sensor can be estimated using the radar equation [33]:

$$P_r = \underbrace{\frac{P_t A_r}{(4\pi)^2}}_{= \text{const}} \cdot \sigma \cdot G(\varphi) \cdot r^{-4}, \tag{2.37}$$

given the transmitting power P_t, the effective aperture area A_r, the radar cross section (RCS) of the target σ, the distance between the sensor and the target r, and the angle dependent antenna gain $G(\varphi)$. For the proposed inverse radar sensor model, the transmitting power P_t and the effective aperture area A_r are assumed to be constant for the sensor. Furthermore we assume a minimal power threshold $P_{\text{r,min}}$, which is used to distinguish between measurements and noise.

We use a simple exponential function to model the antenna gain of the radar sensor, which can be replaced by a more accurate one, if the specifics from the used sensor are known:

$$G(\varphi) = G_{\max} \cdot \exp\left(-\gamma \cdot \frac{\varphi^2}{\Phi^2}\right). \tag{2.38}$$

This model of the antenna gain uses a maximal gain $G_{\max}$ for the main axis of the sensor, some positive tuning parameter γ, as well as the maximal angle Φ defining the sensor's field of view.

With the detection probability basically being a function over the signal to noise ratio (SNR, [34]), we can express the detection probability of our reference object with respect to the angle and range using the following function:

$$p_{\text{detect}} = \frac{1}{1 + \frac{P_{\text{r,min}}}{P_r}} \tag{2.39}$$

$$p_{\text{detect}}(r, \varphi) = \frac{1}{1 + P_{\text{r,min}} \cdot C \cdot \sigma_{\text{ref}}^{-1} \cdot r^4 \cdot G(\varphi)^{-1}}. \tag{2.40}$$

This detection probability $p_{\text{detect}}(r, \varphi)$ of the reference object, scaled by a factor α_{free}, is used as freespace probability in the absence of detections. As this inverse radar model is considering the impact of detections in the later *occupancy calculation* step, we insert this freespace "probability" as default value in our polar grid cell's p_{free}. Note that as this value does not depend on the current detections, the default freeness probability can be precomputed and re-used in every update cycle (example shown in figure 2.7).

Detection Counting Due to the higher uncertainty of the radar sensor, all polar bins in the range $\varphi \pm 2\sigma_\varphi$ are updated by the detections with azimuth angle φ and range r. σ_φ is a sensor specific parameter for the expected standard deviation of detections in their azimuth angle. Figure 2.8 illustrates the affected cells for each detection. Cells with ranges less than $r_{\min}$ or more than $r + 2\sigma_r$ are not updated and skipped completely. Cells with ranges lower than $r - 2\sigma_r$ are considered as more likely to be free, whereas cells with higher ranges are considered more likely to be occupied.

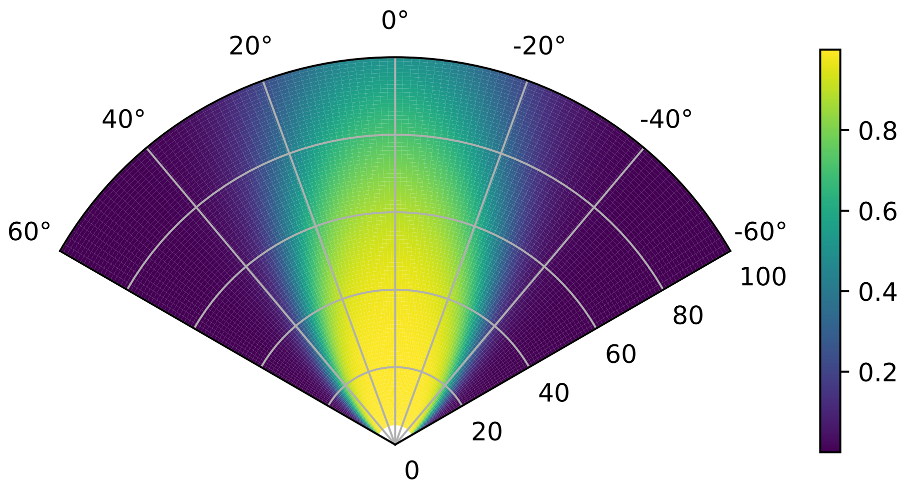


Figure 2.7: Detection probability p_{detect} of our reference object on a radar's field of view with max range 100 m and aperture angle 120°.

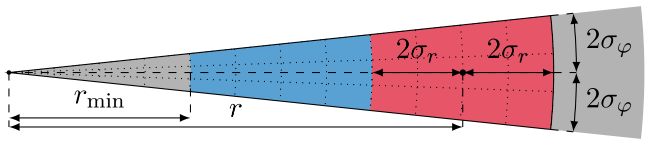


Figure 2.8: Showing the areas, which are ignored, estimated as freespace or as occupied, with respect to the given uncertainty $(\sigma_r, \sigma_\varphi)$. Dotted lines represent borders of the underlying polar cells.

All cells of the considered polar bins with a radius r_c above r_{min} and below the current detection ($r_c \in [r_{\text{min}}, r - 2\sigma_r]$) are considered to be more likely free than occupied. We model the freeness "probability" with a quadratic decay in range as well as in the angle difference to the given detection point:

$$f_\varphi(\varphi_c) = \gamma_{\text{angle, free}} - \gamma_{\text{angle, free}} \cdot \left(\frac{\varphi_c - \varphi}{2\sigma_\varphi}\right)^2 \tag{2.41}$$

$$f_r(r_c) = \gamma_{\text{range, free}} - \gamma_{\text{range, free}} \cdot \left(\frac{r_c - r_{\text{min}}}{r - r_{\text{min}}}\right)^2 \tag{2.42}$$

$$f_{r,\varphi}(r_c, \varphi_c) = f_\varphi(\varphi_c) \cdot f_r(r_c), \tag{2.43}$$

with $\gamma_{\text{angle, free}}$ and $\gamma_{\text{range, free}}$ being constant parameters for the inverse sensor model.

In the area, which is considered to be measured as occupied ($r_c \in [r \pm 2\sigma_r]$), we assume a Gaussian distribution of the measured detection. Therefore the probability density function is:

$$p(R, \Phi) \sim \mathcal{N}(r - R, \sigma_r^2) \cdot \mathcal{N}(\varphi - \Phi, \sigma_\varphi^2). \tag{2.44}$$

The occupancy for a given cell c can then be estimated using the probability density function (eq. 2.44) and the area of the cell A_c (given by the polar grid resolution):

$$o(r_c, \varphi_c) = A_c \cdot p(r_c, \varphi_c). \tag{2.45}$$

For every polar cell c we can define two subsets of all detections $(r, \varphi) \in \mathcal{D}$. In the first subset $\mathcal{F}_c$ all detections indicating freespace are included, whereas in the second subset $\mathcal{O}_c$ all detections indicating occupancy are included:

$$\mathcal{F}_c = \{(r, \varphi) \in \mathcal{D} | r \geq r_c + 2\sigma_r \wedge \varphi \in [\varphi_c \pm 2\sigma_\varphi]\} \tag{2.46}$$
$$\mathcal{O}_c = \{(r, \varphi) \in \mathcal{D} | r \in [r_c \pm 2\sigma_r] \wedge \varphi \in [\varphi_c \pm 2\sigma_\varphi]\}. \tag{2.47}$$

With these two subsets, we can calculate a freeness v_{free} and an occupancy v_{occ} value for every cell:

$$v_{\text{free}} = \min\left(1, \alpha_{\text{free}} \cdot p_{\text{detect}}(r, \varphi) + \sum_{(r,\varphi)\in\mathcal{F}_c} f_{r,\varphi}(r_c, \varphi_c)\right) \tag{2.48}$$
$$v_{\text{occ}} = \min\left(1, \sum_{(r,\varphi)\in\mathcal{O}_c} o_{r,\varphi}(A_c)\right). \tag{2.49}$$

Figure 2.9 shows an example for the two values, given two detections.

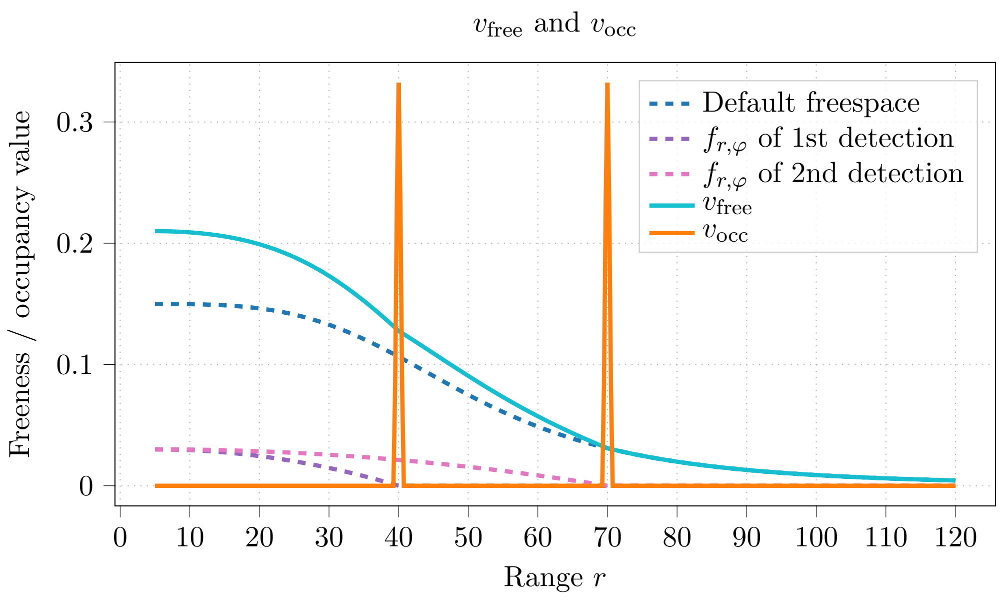


Figure 2.9: The freeness v_{free} and occupancy v_{occ} values for different ranges. This example shows the impact of two detections at ranges 40 m and 70 m.

Occupancy Probability Calculation With the results of the *ray counting* and the *default freespace*, we are able to estimate the occupancy probability for every cell.

In order to model occlusion caused by closer detections, we are considering every angular bin individually. For better readability we skip the angular coordinate in this section for now and use $v_{\text{free}}(r)$ and $v_{\text{occ}}(r)$ as functions returning the freeness and occupancy value for the cell with range index r.

We define the sum of occupancy probabilities over all closer cells as:

$$O(r) = \min\left(1, \sum_{i=1}^{r} o(i)\right). \tag{2.50}$$

The occupancy probability of a single cell with range index r is:

$$o(r) = \max\left(0, v_{\text{occ}}(r) - \lambda O(r-1)\right), \tag{2.51}$$

with λ being a fixed sensor parameter, defining the occlusion impact of previous detections. The freeness probability is defined as:

$$f(r) = \max\left(0, v_{\text{free}}(r) - O(r)\right). \tag{2.52}$$

Figure 2.10 shows the occupancy and freeness probabilities using the example of the same two detections as previously.

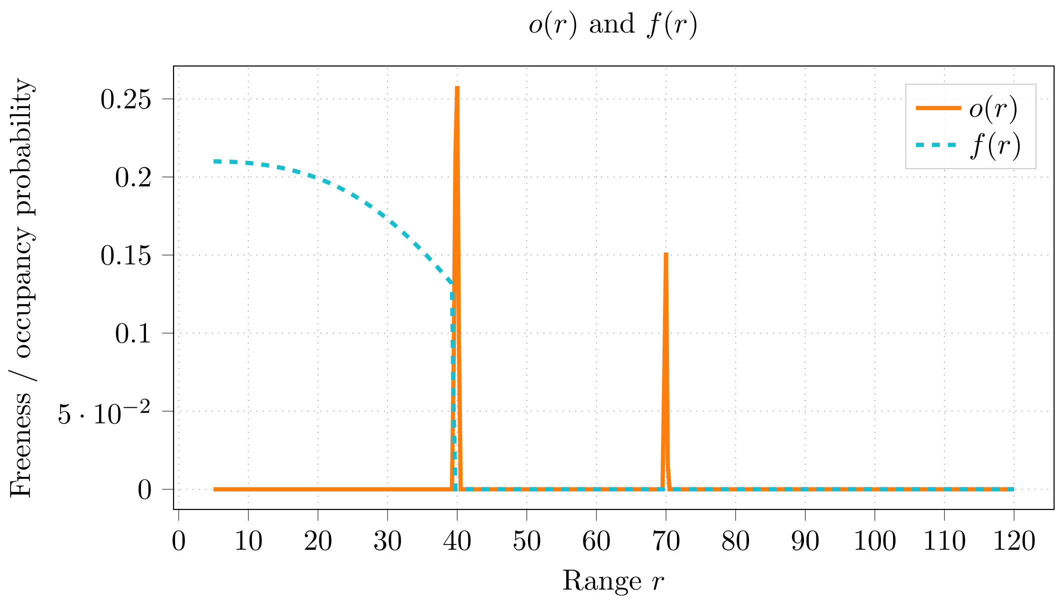


Figure 2.10: Occupancy $o(r)$ and freeness $f(r)$ probabilities for different ranges. This example shows the effect of the same two detections at ranges 40 m and 70 m as in figure 2.9. Occlusion impact chosen as $\lambda = \frac{1}{4}$.

Combining both values leads to the final occupancy probability / masses of the cell c:

$$P(O_c|Z^t) = \begin{cases} \frac{1+o(r)}{2} & , o(r) > 0 \\ \frac{1-f(r)}{2} & , \text{otherwise} \end{cases} \tag{2.53}$$

$$Z_c(SD) = \begin{cases} o(r) & , o(r) > 0 \\ 0 & , \text{otherwise} \end{cases} \tag{2.54}$$

$$Z_c(F) = \begin{cases} 0 & , o(r) > 0 \\ f(r) & , \text{otherwise} \end{cases} \tag{2.55}$$

$$Z_c(FSD) = 1 - Z_c(SD) - Z_c(F). \tag{2.56}$$

Cartesian Transformation The last step of the inverse radar sensor is the transformation into a Cartesian grid. Since we modeled the sensor uncertainties with an impact on not only the measured cell itself, but also its neighboring cells, we recommend the bi-linear interpolation over the max-operator in the case of radar sensors.

2.3.3.3 Dynamic Estimation

One of the advantages using radar sensors is the fact, that they measure a radial velocity v_{rad} for every detection. This additional information is often used to distinguish between "static" and "dynamic" detections. Since the radial velocity is only a projection of the real velocity vector, static detection cannot be distinguished from velocity vectors orthogonal to the sensor. Therefore only dynamic detections can be identified as such, and all non-dynamic classified detections are either static or dynamic (moving only tangentially to the sensor).

Ego Motion Compensation The very first step before taking advantage of the measured radial velocities is to compensate the measured values by the own ego motion. The kinematics of the own vehicle have a huge impact on the measured velocities of the target point, e.g. if the ego vehicle is driving with 20 m/s, a detection caused by a static object directly in front of the vehicle would be measured with a radial velocity of -20 m/s.

To correct the measured radial velocity, we subtract the expected radial velocity of a static detection at the exact same position. Therefore we transform a virtual detection at the location of each detection with an attached velocity vector v of $(0,0)^T$ from the static world coordinate system into the moving sensor coordinate system (see A.2) and name the transformed velocity vector v_s. The expected radial velocity of this virtual static detection can be calculated using vector projection given the position of the detection

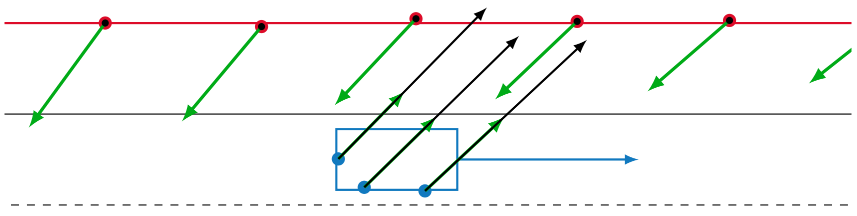


Figure 2.11: Radar-detections of a static guardrail and moving target vehicle. The vehicle with the mounted radar sensor is coming from the bottom and making a right turn to enter the lane of the target vehicle. The measured radial velocities by the sensor are shown in green, whereas the compensated radial velocities are shown in black.

within the sensor coordinate system:

$$p_s = \begin{pmatrix} \cos(\varphi) \\ \sin(\varphi) \end{pmatrix} \cdot r \tag{2.57}$$

$$\hat{v}_{\text{rad}} = \frac{p_s^T \cdot v_s}{\sqrt{p_s^T \cdot p_s}}. \tag{2.58}$$

The ego motion compensated radial velocity $\widetilde{v}_{\text{rad}}$ is then:

$$\widetilde{v}_{\text{rad}} = v_{\text{rad}} - \hat{v}_{\text{rad}}. \tag{2.59}$$

To qualify a detection as "dynamic", we use an exponential function to provide a dynamic belief value between 0 and 1 (see 2.12):

$$p_{\text{dynamic}} = 1 - \exp\left(-\frac{\widetilde{v}_{\text{rad}}^2}{2 \cdot \sigma_{v_{\text{rad}}}^2}\right). \tag{2.60}$$

In the case of using Dempster-Shafer masses as output of the sensor model, we assign the ratio of p_{dynamic} from the static or dynamic mass SD (eq. 2.56) to the dynamic occupied mass D as follows:

$$Z_c(D) \leftarrow p_{\text{dynamic}} \cdot Z_c(SD) \tag{2.61}$$

$$Z_c(SD) \leftarrow Z_c(SD) - Z_c(D). \tag{2.62}$$

Gaussian Velocity Estimation Based on the ego motion compensated radial velocity $\widetilde{v}_{\text{rad}}$ we want to generate a statement for the 2-D velocity vector of the detection target. As getting the velocity vector out of a single measured radial velocity is not possible, we want to model our assumed velocity using a Gaussian distribution. Under the assumption, that the absolute speed is limited by some constant v_{max}, it follows, that with a higher absolute radial velocity, the possible absolute tangential velocity component is getting smaller (see fig. 2.13).

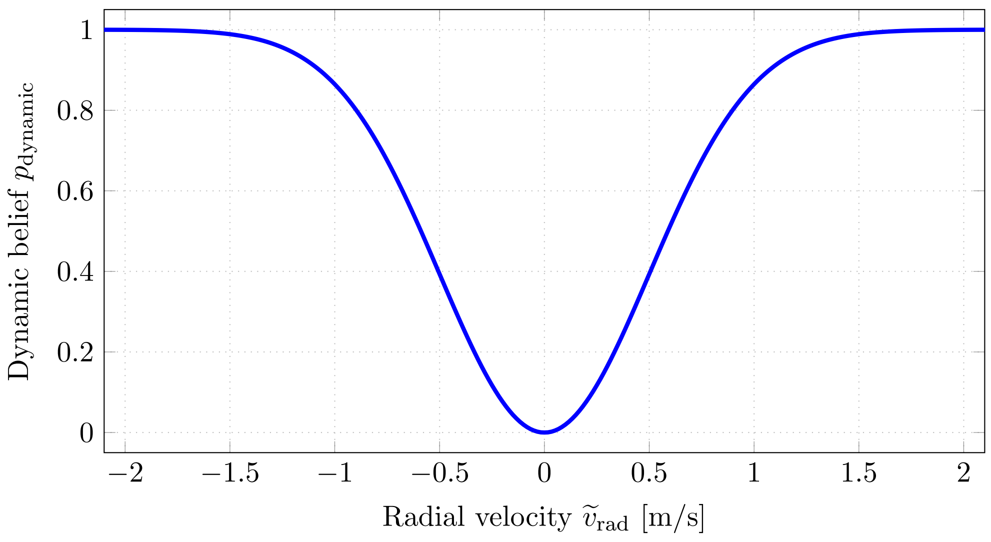


Figure 2.12: Belief for dynamic qualification of a radar detection given the ego motion compensated radial velocity.

The maximal tangential velocity value is defined by the assumed overall maximal speed value v_{max} and the radial velocity $\widetilde{v}_{\text{rad}}$:

$$v^2_{\text{tan, max}} = v^2_{\text{max}} - \widetilde{v}^2_{\text{rad}}. \tag{2.63}$$

We assume the standard deviation of the radial velocity component to be a sensor specific constant $\sigma_{v_{\text{rad}}}$. The standard deviation of the tangential component on the other hand is approximated by half of the maximal possible tangential velocity:

$$\begin{aligned} \sigma^2_{v_{\text{tan}}} &= \left(\frac{v_{\text{tan, max}}}{2}\right)^2 && (2.64) \\ &= \frac{1}{4}\left(v^2_{\text{max}} - v^2_{\text{rad}}\right). && (2.65) \end{aligned}$$

Given the standard deviations for the radial and tangential velocity components, as well as the measured angle (in world coordinates) $\widetilde{\varphi}$, we can express the Gaussian approximation of the velocity with mean v and covariance matrix Σ_v (fig. 2.14):

$$\begin{aligned} R &= \begin{pmatrix} \cos\widetilde{\varphi} & -\sin\widetilde{\varphi} \\ \sin\widetilde{\varphi} & \cos\widetilde{\varphi} \end{pmatrix} && (2.66) \\ v &= \begin{pmatrix} v_x \\ v_y \end{pmatrix} = R \begin{pmatrix} \widetilde{v}_{\text{rad}} \\ 0 \end{pmatrix} && (2.67) \\ &= \begin{pmatrix} \cos\widetilde{\varphi} \cdot \widetilde{v}_{\text{rad}} \\ \sin\widetilde{\varphi} \cdot \widetilde{v}_{\text{rad}} \end{pmatrix} && (2.68) \\ \Sigma_v &= R \begin{pmatrix} \sigma^2_{v_{\text{rad}}} & 0 \\ 0 & \sigma^2_{v_{\text{tan}}} \end{pmatrix} R^T. && (2.69) \end{aligned}$$

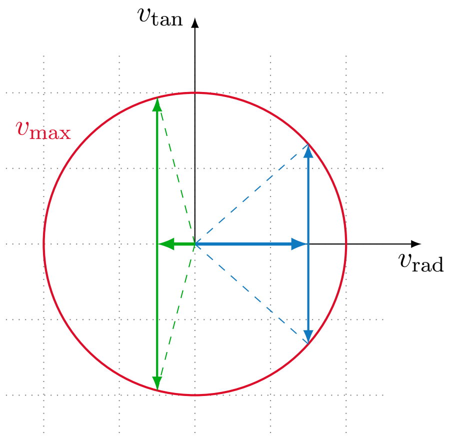


Figure 2.13: Two (blue and cyan) possible radial velocities (drawn as arrows) and their corresponding possible tangential velocity components (drawn as double-arrows).

2.3.4 Sensor Fusion

Instead of fusing the sensor grids prior to actually updating the grid map, all single sensor grids can be applied also directly to the grid map itself. Fusing the sensor grids beforehand has two main advantages, first of all the fusion can be easily parallelized (compare the parallel streams in fig. 2.2), but the second and even more important reason is that fusion between the different sensors can be controlled separately from the fusion with the accumulated data.

Fusion Using Bayes To fuse two sensor grids ($P_1(O_c|Z_1^t)$ and $P_2(O_c|Z_2^t)$), we can simply use an addition in the logarithmic odds representation (compare with eq. 2.3):

$$l(O_c|Z_1^t, Z_2^t) = l(O_c|Z_1^t) + l(O_c|Z_2^t). \tag{2.70}$$

The fused occupancy estimation $l(O_c|Z_{1,2}^t)$ can then be used to update the grid map itself. As the Bayesian framework has no explicit way of dealing with conflicts (e.g. sensor 1 estimates the cell as being free, whereas sensor 2 estimates the same cell as being occupied), the only options to influence the sensor fusion here is to limit the allowed contribution of each sensor, by applying a min/max operation before executing the addition or using a weighted sum:

$$\widetilde{l}(O_c|Z_1^t) = \min\left(\tau_{\text{max, sensor1}}, \max\left(\tau_{\text{min, sensor1}}, l(O_c|Z_1^t)\right)\right) \tag{2.71}$$

$$\widetilde{l}(O_c|Z_2^t) = \min\left(\tau_{\text{max, sensor2}}, \max\left(\tau_{\text{min, sensor2}}, l(O_c|Z_2^t)\right)\right) \tag{2.72}$$

$$l(O_c|Z_1^t, Z_2^t) = \alpha_1 \cdot \widetilde{l}(O_c|Z_1^t) + \alpha_2 \cdot \widetilde{l}(O_c|Z_2^t). \tag{2.73}$$

In our implementation we prefer using the Dempster-Shafer framework, which allows us to handle the conflicts in a more direct fashion, as described in the following paragraphs.

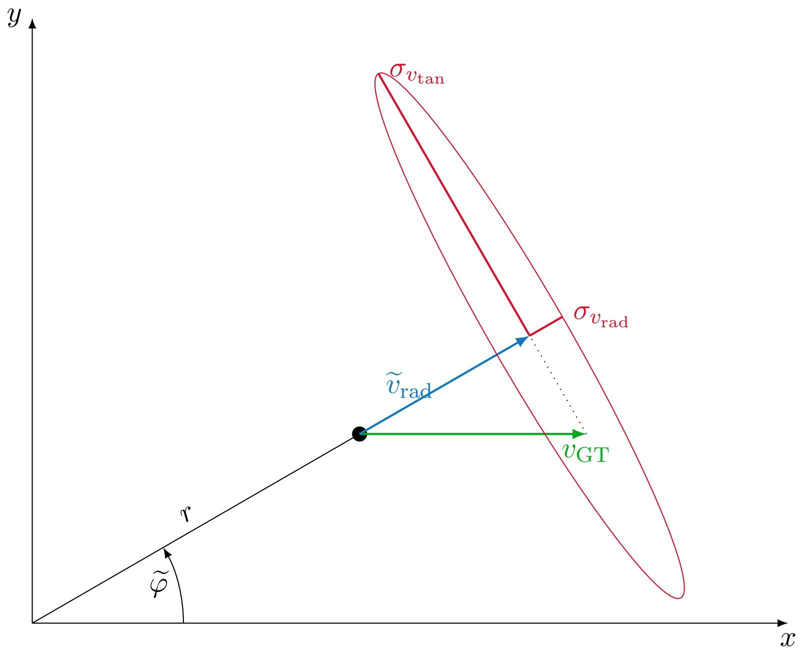


Figure 2.14: Estimation $(\sigma_{\mathrm{rad}}, \sigma_{\tan})$ of the ground-truth velocity v_{GT} from the radial velocity vector $\widetilde{v}_{\mathrm{rad}}$.

Dempster-Shafer Fusion of Two Lidar Grids In this paragraph we want to discuss the grid fusion of two grids generated by our inverse lidar sensor model (section 2.3.2): Z_1 and Z_2. We use the default combination rule, as described in section 2.2:

$$Z_{1,2} = Z_1 \oplus Z_2. \tag{2.74}$$

In the case of lidar grids, we only have the masses for F, SD and FSD, leading to following conflicts:

$$\zeta_1 = Z_1(F) \cdot Z_2(SD) \tag{2.75}$$
$$\zeta_2 = Z_1(SD) \cdot Z_2(F), \tag{2.76}$$

namely one sensor stating that a cell is free, whereas the other sensor is stating that the cell is occupied. For the lidar/lidar fusion we simply assign the conflict masses to the *unknown* mass FSD as there is no reasonable way, why we should trust one sensor over the other:

$$Z_{1,2}(FSD) \leftarrow \zeta_1 + \zeta_2. \tag{2.77}$$

Dempster-Shafer Fusion of Two Radar Grids The fusion of two sensor grids, generated by our inverse radar sensor model (section 2.3.3) differs only slightly compared to the fusion of two lidar grids. As we do have the additional mass D as output in the radar grids, we get two additional conflicts:

$$\zeta_3 = Z_1(F) \cdot Z_2(D) \tag{2.78}$$
$$\zeta_4 = Z_1(D) \cdot Z_2(F). \tag{2.79}$$

With the same reasoning as before, we assign those two conflict masses additionally to the FSD mass, as in 2.77:

$$Z_{1,2}(FSD) \leftarrow \zeta_3 + \zeta_4. \tag{2.80}$$

Dempster-Shafer Fusion of Lidar and Radar Grids In the case of fusing the sensor grid of a lidar Z_L and a radar grid Z_R, the same four conflicts mentioned previously $(\zeta_1, \zeta_2, \zeta_3, \zeta_4)$ may occur. Due to the high precision of the lidar compared to the radar, we trust the estimation of the lidar sensor more and therefore assign all conflict masses in favor of the lidar sensor:

$$\zeta_1 = Z_L(F) \cdot Z_R(SD) \tag{2.81}$$
$$\zeta_2 = Z_L(SD) \cdot Z_R(F) \tag{2.82}$$
$$\zeta_3 = Z_L(F) \cdot Z_R(D) \tag{2.83}$$
$$\zeta_4 = Z_L(D) \cdot Z_R(F) \tag{2.84}$$
$$Z_{1,2}(F) \leftarrow \zeta_1 + \zeta_3 \tag{2.85}$$
$$Z_{1,2}(D) \leftarrow \zeta_4 \tag{2.86}$$
$$Z_{1,2}(SD) \leftarrow \zeta_2. \tag{2.87}$$

Note, that the exact same argumentation holds true, if instead of the lidar grid a already fused lidar and radar grid is fused again.

Fusion of Gaussian Velocity Estimations In some cases we do get two or more velocity estimations for the same cell from different sensors (e.g. car in front of the ego vehicle is seen by two radar sensors). Therefore we need to fuse the information of sensor 1 (v_1, Σ_{v_1}) with the information of sensor 2 (v_2, Σ_{v_2}) to get the joint distribution v, Σ_v:

$$v = \Sigma_{v_2} \left(\Sigma_{v_1} + \Sigma_{v_2}\right)^{-1} v_1 + \Sigma_{v_1} \left(\Sigma_{v_1} + \Sigma_{v_2}\right)^{-1} v_2 \tag{2.88}$$
$$\Sigma_v = \Sigma_1 \left(\Sigma_{v_1} + \Sigma_{v_2}\right)^{-1} \Sigma_2. \tag{2.89}$$

Figure 2.15 shows an example of two such velocity estimations and the resulting joint distribution.

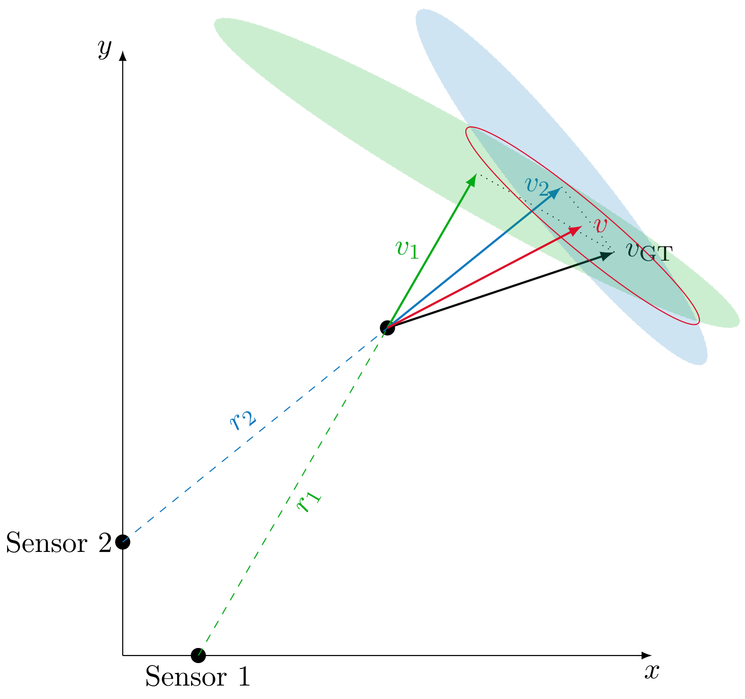


Figure 2.15: Fusion of two Gaussian velocity distributions

3 Dynamic Occupancy Grid

The dynamic occupancy grid can be interpreted as an extension of the static occupancy grid (chapter 2) for dynamic environments. In addition to the estimation of occupancy probability, the dynamic occupancy grid also provides an estimation of the velocity of the cells. In that sense the static occupancy grid is a special case of the dynamic one, with a fixed velocity of 0. In this chapter we will present the dynamic extension for both the occupancy grids generated using traditional Bayes (2.1) as well as Dempster-Shafer (2.2).

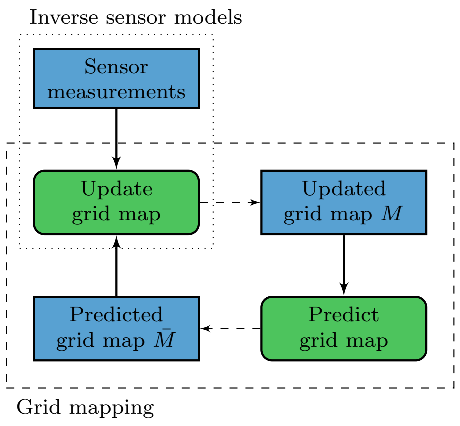


Figure 3.1: Overview of dynamic occupancy grid mapping

The dynamic occupancy grid uses classic update/predict cycles to estimate the static and dynamic environment (fig. 3.1).

3.1 Bayes 4-D

In section 2.1 the update formula for a static occupancy grid using Bayes' theorem was derived as (2.1):

$$\frac{P(O_c|Z^{1:t})}{1-P(O_c|Z^{1:t})} = \frac{P(O_c|Z^{t})}{1-P(O_c|Z^{t})} \cdot \frac{P(O_c|Z^{1:t-1})}{1-P(O_c|Z^{1:t-1})} \cdot \frac{1-P(O_c)}{P(O_c)}.$$

In a static environment the occupancy of a single cell should not change over time. In a dynamic environment a changing occupancy of cells is caused by moving objects, therefore the update formula has to be modified. Rather than updating conditional probabilities of time-invariant occupancy O_c, different time stamps have to be considered:

$$\frac{P(O_c^{\mathbf{t}}|Z^{1:t})}{1-P(O_c^{\mathbf{t}}|Z^{1:t})} = \frac{P(O_c^{\mathbf{t}}|Z^{t})}{1-P(O_c^{\mathbf{t}}|Z^{t})} \cdot \frac{P(O_c^{\mathbf{t}}|Z^{1:t-1})}{1-P(O_c^{\mathbf{t}}|Z^{1:t-1})} \cdot \frac{1-P(O_c^{\mathbf{t}})}{P(O_c^{\mathbf{t}})}. \tag{3.1}$$

As in the static case, we assume a prior occupancy probability of 50%, which eliminates the last factor:

$$\frac{P(O_c^{t}|Z^{1:t})}{1-P(O_c^{t}|Z^{1:t})} = \frac{P(O_c^{t}|Z^{t})}{1-P(O_c^{t}|Z^{t})} \cdot \frac{P(O_c^{t}|Z^{1:t-1})}{1-P(O_c^{t}|Z^{1:t-1})}. \tag{3.2}$$

Equation 3.2 defines the occupancy probability of a cell c being occupied at time t, given all sensor measurements up to time t. The included inverse sensor model $P(O_c^t|Z^t)$ stays the same as in the static occupancy grid since the occupancy estimation task for time t given the sensor measurement of time t did not change. The recursive term $P(O_c^t|Z^{1:t-1})$ on the other hand predicts the cell's occupancy probability for time t, given the past sensor measurements up to time $t-1$. Unlike in the static case, where this prediction can be modeled using the identity function, the dynamic case requires this prediction function to be specified.

3.1.1 Prediction Formula

The presented prediction formula is based on the following assumptions:

- In addition to the occupancy probability, each cell also has a velocity distribution.
- Only occupancy has a velocity, freespace is defined as not occupied and is not associated with a velocity.
- The velocity is described in terms of cells per discrete update cycle.
- The movement between cells is constant and the motion follows a linear model between two time points.
- A cell c is only reachable from exactly one other cell given a specific velocity v: $c_{\text{source}} + v = c_{\text{destination}}$.
- Cells outside the mapped area are treated as unknown regarding initial occupancy and velocity distribution.
- The initial velocity distribution of a cell is uniform (same as for the occupancy).

Following the stated assumptions, the joint distribution of being occupied with a given velocity can be computed. As no change in the velocity is assumed, a cell c being occupied with velocity v at time t has the same probability as cell $c-v$ at time $t-1$:

$$\begin{aligned} P(O_c^t = \text{occ}, V_c^t = v|Z^{1:t-1}) &= P(O_{c-v}^{t-1} = \text{occ}, V_{c-v}^{t-1} = v|Z^{1:t-1}) \\ &= P(O_{c-v}^{t-1} = \text{occ}|Z^{1:t-1}) \cdot P(V_{c-v}^{t-1} = v|O_{c-v}^{t-1} = \text{occ}, Z^{1:t-1}). \end{aligned} \tag{3.3}$$

This calculation requires the information of the velocity distribution and the occupancy probability to be stored for every cell, which is covered by our assumptions.

Next the predicted occupancy probability can be calculated by marginalization:

$$P(O_c^t = \text{occ}|Z^{1:t-1}) = \sum_{v \in V} P(O_c^t = \text{occ}, V_c^t = v|Z^{1:t-1}). \tag{3.4}$$

This predicted occupancy probability is used in equation 3.2 in combination with the result of the inverse sensor model (section 2.3) to estimate the occupancy probability for time t. In order to have the required information for the next prediction cycle (eq. 3.3), the velocity distribution, given that the cell is occupied, has to be re-calculated using the joint distribution:

$$P(V_c^t = v|O_c^t = \text{occ}, Z^{1:t-1}) = \frac{P(O_c^t = \text{occ}, V_c^t = v|Z^{1:t-1})}{P(O_c^t = \text{occ}|Z^{1:t-1})}. \tag{3.5}$$

As mentioned we assume a uniform distribution as initial value, meaning that each velocity is equally likely:

$$P(V_c^0 = v|O_c^0 = \text{occ}) = \frac{1}{|\mathcal{V}|}. \tag{3.6}$$

3.1.2 Forgetting Old Measurements

To enable faster correction of erroneous sensor data or prediction results and to increase the uncertainty of cells that have not been updated for a longer period of time, we introduce a forgetting factor ε. The forgetting factor ε represents the weight of the new state being equal to the initial state and extends the prediction equation (3.3):

$$\begin{aligned} &P(O_c^t = \text{occ}, V_c^t = v|Z^{1:t-1}) = \\ &\quad \boldsymbol{(1-\varepsilon)} \cdot P(O_{c-v}^{t-1} = \text{occ}|Z^{1:t-1}) \cdot P(V_{c-v}^{t-1} = v|O_{c-v}^{t-1} = \text{occ}, Z^{1:t-1}) \\ &\quad + \boldsymbol{\varepsilon} \cdot P(O_c^0 = \text{occ}) \cdot P(V_c^0 = v|O_c^0 = \text{occ}). \end{aligned} \tag{3.7}$$

With the initial values of the occupancy $P(O_c^0 = \text{occ})$ and the velocity distribution $P(V_c^0 = v|O_c^0 = \text{occ})$ being cell and velocity independent constants, we can substitute the last term with:

$$\varepsilon \cdot P(O_c^0 = \text{occ}) \cdot P(V_c^0 = v|O_c^0 = \text{occ}) = \frac{\varepsilon}{2 \cdot |\mathcal{V}|}. \tag{3.8}$$

The forgetting factor ε ensures that the dynamic occupancy map will convert to its initial state if no updates during measurements can be applied. Furthermore it also prevents the occupancy probabilities from getting too close to 0 or 1, which would lead to numeric instabilities.

3.1.3 Static Map as Special Dynamic Case

It is important to mention, that the static occupancy grid map is only a special case of the dynamic occupancy grid map, which in turn makes the proposed dynamic occupancy grid map a generalization of the static one. In the static case the probability of the cells having a velocity of 0 is 100%, leading to the probability of having $v \neq 0$ of 0%:

$$P(V_{c-v}^{t-1} = 0 | O_{c-v}^{t-1}) = 1 \tag{3.9}$$
$$P(V_{c-v}^{t-1} \neq 0 | O_{c-v}^{t-1}) = 0, \tag{3.10}$$

inserting this values into the joint distribution equation 3.3 leads to following occupancy prediction:

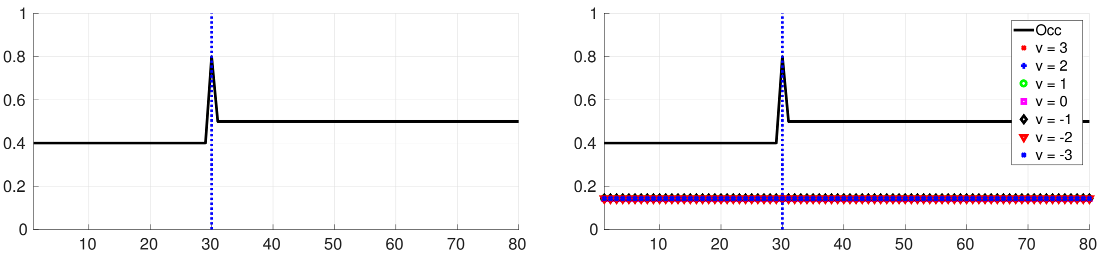
$$P(O_c^t = \text{occ} | Z^{1:t-1}) = P(O_c^t = \text{occ}, V_c^t = 0 | Z^{1:t-1}) \tag{3.11}$$
$$= P(O_c^{t-1} = \text{occ} | Z^{1:t-1}), \tag{3.12}$$

which is equal to the static prediction used in eq. 2.2.

3.1.4 Example

We illustrate the described algorithm using a simple one-dimensional example. For this we use the following setup:

- a grid map with 81 cells ($\mathcal{C} = \{c \in \mathbb{Z} : 0 \leq c \leq 80\}$),
- a distance measuring sensor located at $c = 0$,
- a fixed set of allowed velocities ($\mathcal{V} = \{v \in \mathbb{Z} : -3 \leq v \leq +3\}$), and
- a single target starting at position $c = 30$ with a constant velocity of $v = +2$.



(a) Measurement model for object located at $c = 30$.

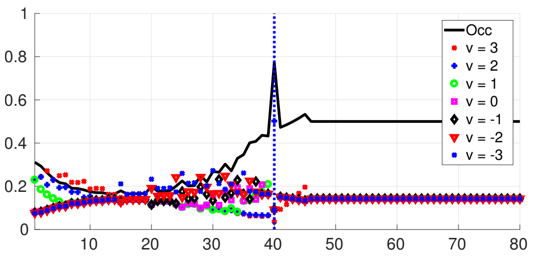
(b) Updated occupancy map for time stamp $t = 1$ given measurement Z^1.

Figure 3.2: Measurement model $P(O_c^t | Z^t)$, and updated occupancy map $P(O_c | Z^{1:t})$ for $t = 1$. The blue vertical line denotes the real position (ground-truth).

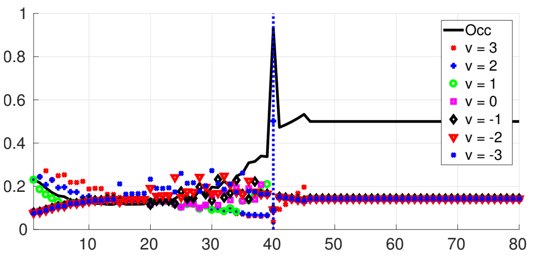
Our simple inverse sensor model $P(O_c^t | Z^t)$ is assuming an occupancy probability of 40% in front of the detection (ground-truth object), an occupancy probability of 80%

at the location of the object, and a value of 50% behind the object, as shown in figure 3.2a.

Figure 3.2b shows the map after incorporating the first measurement. The updated occupancy probabilities match with the inverse sensor model, as the initial values of 0.5 represents basically no prior information. As the update (eq. 3.2) is not affecting the velocity distribution, the probabilities for the single values are equivalent to their initial value (eq. 3.6) of $\frac{1}{|\mathcal{V}|} = \frac{1}{7}$.



(a) Predicted occupancy map for time stamp $t = 6$ given measurements $Z^{1:5}$.



(b) Updated occupancy map for time stamp $t = 6$ given the latest measurement Z^6.

Figure 3.3: Predicted occupancy map $P(O_c^t | Z^{1:t-1})$ and updated occupancy map $P(O_c^t | Z^{1:t})$ for $t = 6$. The blue vertical line denotes the real position (ground-truth).

After a few (here: five) time steps (figure 3.3), the predicted and updated map shows low occupancy probabilities in front of the object, as well as a high occupancy probability at the location of the object: $P(O_{40}^6 = \text{occ} | Z^{1:5}) = 0.77$. Furthermore the correct velocity of +2 is estimated as the most likely one with a probability of: $P(V_{40}^6 = +2 | O_{40}^6 = \text{occ}, Z^{1:5}) = 0.50$.

This small example shows, that the proposed system is able to estimate a cell velocity without explicit velocity measurements.

3.1.5 Limitations in Two Dimensions

Although the presented example only illustrated a one-dimensional world, the actual framework can be easily extended to two dimensions:

- The coordinate c has now two components, one for each dimension: $c = (c_x, c_y)^\top \in \mathcal{C} \subseteq \mathbb{Z}^2$, and
- the velocity V is also expressed using two components: $v = (v_x, v_y)^\top \in \mathcal{V} \subseteq \mathbb{Z}^2$.

The set of possible values for the velocities $\mathcal{V}$ has to be defined using the absolute velocity value:

$$\mathcal{V}_{\text{2-D}} = \{(v_x, v_y)^\top \in \mathbb{Z}^2 \mid \sqrt{v_x^2 + v_y^2} \leq v_{\max}\}.$$

The total number of possible velocities $|\mathcal{V}|$ can be approximated using the formula for calculating the area of a disk:

$$|\mathcal{V}| \approx v_{\max}^2 \pi.$$

As the complete velocity distribution has to be stored for every single cell, the amount of required memory increases not only quadratically with the dimension of the grid map, but it also increases quadratically with the maximum allowed velocity value.

Let's consider the following inner city automotive setup with:

- The grid map covering an area of $120\,\mathrm{m} \times 120\,\mathrm{m}$ around the vehicle,
- a cell-size of $0.125\,\mathrm{m} \times 0.125\,\mathrm{m}$,
- an update/predict frequency of $16\,\mathrm{Hz}$, and
- a maximum allowed velocity value $v_{\max}$ of $72\,\mathrm{km} = 20\,\mathrm{m/s}$.

This setup leads to a total number of cells:

$$N = \frac{120}{0.125} \cdot \frac{120}{0.125} = 960 \cdot 960 = 921\ 600.$$

Converting the maximum velocity to "cells per time step" we get the total number of possible velocities:

$$\begin{aligned} v_{\max} &= \frac{20\,\mathrm{m/s}}{16\,\mathrm{Hz} \cdot 0.125\,\mathrm{m}} = 10 \\ |\mathcal{V}| &\approx 10^2\pi \approx 315. \end{aligned}$$

We chose the single precision *float* data type to store our probability values. The size of a *float* is 4B (byte). Therefore, we get a total memory requirement of:

$$\begin{aligned} \text{MemoryRequirement} &= N \cdot (1 + |\mathcal{V}|) \cdot 4\,\mathrm{B} \\ &= 921\ 600 \cdot (1 + 315) \cdot 4\,\mathrm{B} \\ &= 1\,164\,902\,400\,\mathrm{B} \\ &\approx 1.1\,\mathrm{GiB}. \end{aligned}$$

Storing and processing of such an amount of data may be feasible with the processing power of today's computers, but is not feasible to work on an automotive ECU. Furthermore, this requirement increases quadratically with the maximum supported velocity value or the configured grid map size.

The majority of cells surrounding the vehicle is either unknown, freespace or static occupied. Only a very small amount of the cells actually represent a dynamic occupied area. Evaluating our ground-truth data set (chapter 6.1) shows that on average only 0.235% of the cells are dynamic occupied, with a maximum of 0.674%. Therefore storing and processing of the velocity distribution for every single cell is a waste of memory and computation time. This conclusion leads us to the following section, where an alternative approach is presented, which focuses on the processing of the dynamic occupied areas.

3.2 Particle Dempster Shafer Hybrid

The dynamic particle-based Dempster Shafer hybrid is an extension of the static one presented in 2.2. The main idea of the particle filter Dempster Shafer hybrid approach is to keep the computational efforts in unknown, empty or static areas of the grid map at a minimum, while enabling the required computation in dynamic areas. This balancing is essentially made possible by two key actions.

The first measure is to extend the set of possible hypothesis to Free-space, Static-occupied and Dynamic-occupied. This allows the classification of areas with dynamic occupancy directly on the cell's mass distribution. Secondly the prediction of the dynamic occupancy masses is carried out by a fixed number of particles, allowing a "constant" runtime and allows to focus the prediction computation on the areas of the map, where movement is expected.

In this section we will discuss the update and prediction steps of the *Particle Dempster Shafer Hybrid.*

Hypotheses In addition to the two possible hypotheses of the static Dempster Shafer grid, a third one D for dynamic occupied is added, ending up in:

$$\Theta = \{F, S, D\}. \tag{3.13}$$

This results in following frame of discernment (power set):

$$2^{\Theta} = \{\emptyset, F, S, D, FS, FD, SD, \Theta\}. \tag{3.14}$$

We follow the argumentation of [18] and define the mass of static occupied or freespace to be 0, as both events are mutually exclusive. As the mass of none hypothesis is also defined as 0, we end up with following possible hypothesis:

- F: Freespace: the cell is currently estimated as being free.
- S: Static occupied: the cell is currently estimated as static occupied.
- D: Dynamic occupied: the cell is currently estimated as dynamic occupied.
- FD: Freespace or Dynamic occupied: the cell is currently considered as free or dynamic occupied, meaning that the cell represents drivable area, e.g. it has been sensed as free before but is currently not sensed and hence could be occupied by a dynamic object again.
- SD: Static or Dynamic occupied: the cell is considered to be occupied, but it cannot be determined if it is occupied by a static or dynamic object.
- FSD: Either of the 3 hypothesis is possible, e.g. nothing is known about this cell (yet).

Particles Next to the cells, where the information about the mass distribution is stored, we also use particles for the dynamic prediction and velocity estimation of the cells. The total number of the particles over the complete grid map is chosen to be constant. Nevertheless the distribution of the particles over the cells changes from frame to frame. Each particle p_i carries the following information:

- 2-D position x_i: This position is absolute and interpreted in meters. The corresponding cell index for this position is denoted as $c(x_i)$.
- 2-D velocity vector v_i: The velocity vector is also absolute and interpreted in meters per second.
- Age a_i: The age describes how old a given particle is and is incremented in every prediction cycle. New particles start with an age of zero.
- Weight ω_i: The weight of the particle (between 0 and 1).

The weights of all particles belonging to the same cell are defined to sum up to 1:

$$\sum_{i|c(x_i)=c} \omega_i = 1. \tag{3.15}$$

This definition explicitly means that the sum of the particle weights in the complete grid map is neither defined to sum up to one, nor is expected to stay constant during multiple prediction cycles.

As the dynamics of the cells are represented by its associated particles, the estimated velocity vector of each cell c is calculated as a weighted sum over all particles belonging to that cell, which have a given minimum age τ_a:

$$v_c = \eta \sum_{i|c(x_i)=c \wedge a \geq \tau_a} (\omega_i \cdot v_i) \tag{3.16}$$

$$\eta^{-1} = \sum_{i|c(x_i)=c \wedge a \geq \tau_a} \omega_i. \tag{3.17}$$

3.2.1 Update

The update of the dynamic grid can be separated into two steps. During the first step (3.2.1.1) the mass values of the latest map prediction $\bar{M}^t$ (see 3.2.2) are updated with the mass values from generated sensor grid Z^t (see 2.3). Afterwards the particle weights will be adjusted (3.2.1.2) using the velocity estimation of the generated sensor grid (see 2.3.4).

3.2.1.1 Cell Updates

This first update step is based on the standard combination rule with a few adaptions. As the sensor grid Z^t contains mass values for $\theta_Z \in \{F, D, SD, FSD\}$ and the predicted map $\bar{M}^t$ values for $\theta_{\bar{M}} \in \{S, D, FD, SD, FSD\}$, the following conflicts can occur:

$$\zeta_1 = \bar{M}(S) \cdot Z(F) \tag{3.18}$$
$$\zeta_2 = \bar{M}(S) \cdot Z(D) \tag{3.19}$$
$$\zeta_3 = \bar{M}(D) \cdot Z(F) \tag{3.20}$$
$$\zeta_4 = \bar{M}(SD) \cdot Z(F) \tag{3.21}$$

The conflict mass ζ_1 is equally assigned to S and F as there is no reason to prefer one over the other:

$$M(S) \leftarrow \frac{1}{2}\zeta_1 \tag{3.22}$$
$$M(F) \leftarrow \frac{1}{2}\zeta_1. \tag{3.23}$$

In the second case, in which the predicted grid estimates static occupancy, whereas the current sensor measurement is indicating dynamic occupancy, we assign the conflict mass to the static or dynamic occupancy mass value:

$$M(SD) \leftarrow \zeta_2. \tag{3.24}$$

The third and fourth conflict masses ζ_3, ζ_4 occur, because the sensor measurement indicates freespace, whereas the prediction is estimating occupancy. In this case we trust the current measurement more than the prediction assigning both conflict masses to F:

$$M(F) \leftarrow \zeta_3 + \zeta_4. \tag{3.25}$$

Additionally to the assignment of conflict masses, the update rule of the static or dynamic occupancy mass is modified similar to [18]. As both, the lidar and the radar, sensor models are not able to separate static occupancy, there would be no evidence for static occupancy at all. On the other hand, if a single cell is repeatedly sensed as being occupied it is highly likely that this cell is static occupied. Following this argumentation leads to the modification of the update rule in the case of SD:

$$M(SD) \mathrel{\hat{=}} \underbrace{\bar{M}(SD) \cdot Z(FSD)}_{\lambda_1} + \underbrace{\bar{M}(SD) \cdot Z(SD)}_{\lambda_2} + \underbrace{\bar{M}(FSD) \cdot Z(SD)}_{\lambda_3} \tag{3.26}$$
$$M(S) \leftarrow \beta \cdot \lambda_2 \tag{3.27}$$
$$M(SD) \leftarrow -\beta \cdot \lambda_2 \tag{3.28}$$

Final Update Rule Summarizing the conflict assignments and the modification of the SD combination leads to following update formula:

$$M(F) = \left(\bar{M} \oplus Z\right)(F) + \frac{1}{2}\zeta_1 + \zeta_3 + \zeta_4 \tag{3.29}$$

$$M(S) = \left(\bar{M} \oplus Z\right)(S) + \frac{1}{2}\zeta_1 + \beta\lambda_2 \tag{3.30}$$

$$M(D) = \left(\bar{M} \oplus Z\right)(D) \tag{3.31}$$

$$M(FD) = \left(\bar{M} \oplus Z\right)(FD) \tag{3.32}$$

$$M(SD) = \left(\bar{M} \oplus Z\right)(SD) + \zeta_2 - \beta\lambda_2 \tag{3.33}$$

$$M(FSD) = \left(\bar{M} \oplus Z\right)(FSD) \tag{3.34}$$

3.2.1.2 Particle Updates

In this update step the particle weights will be updated, if a velocity estimation is available for the particular cell. The velocity estimation of the sensor grid Z^t for a cell c is given as a two dimensional Gaussian distribution with mean value $v_{z,c}$ and covariance matrix $\Sigma_{z,c}$. In order to consider this velocity information we multiply the weight ω_i of each particle i, which is located in the particular cell, with the density value of the Gaussian velocity distribution, regarding the particle's velocity v_i:

$$\hat{\omega}_i = \omega_i \cdot \frac{\exp\left(-\frac{1}{2}\left(v_i - v_{z,c}\right)^T \Sigma_{z,c}^{-1}\left(v_i - v_{z,c}\right)\right)}{\sqrt{(2\pi)^2 \left|\Sigma_{z,c}\right|}} \tag{3.35}$$

After this update, the weights have to be normalized to sum up to one for each cell:

$$\omega_i = \frac{\hat{\omega}_i}{\sum_i \hat{\omega}_i} \tag{3.36}$$

Modifying the particle weights in this way, enables us to incorporate measured velocity components of the radar sensor. The additional velocity information leads to a faster convergence using the particles and therefore to a better velocity information in our dynamic grid approach.

3.2.2 Prediction

The grid map prediction can be separated into the static and the dynamic prediction part as illustrated in figure 3.4 and is explained in this section.

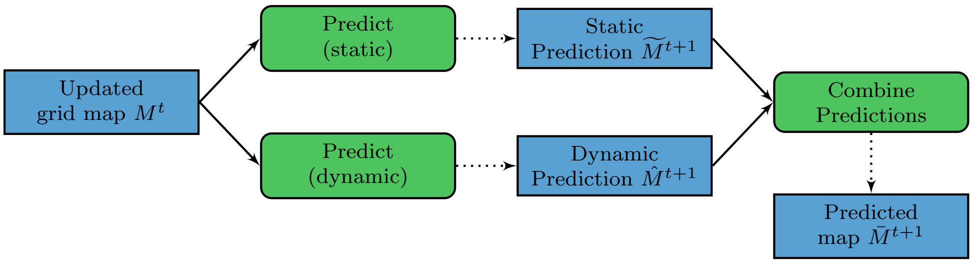

Updated grid map M^t
Predict (static)
Predict (dynamic)
Static Prediction $\widetilde{M}^{t+1}$
Dynamic Prediction $\hat{M}^{t+1}$
Combine Predictions
Predicted map $\bar{M}^{t+1}$

Figure 3.4: Dynamic grid prediction overview

3.2.2.1 Static Prediction

The static prediction "predicts" the mass values, which do not move and can be expressed with following prediction equations of [18]:

$$\widetilde{M}^{t+1}(F) = 0 \tag{3.37}$$

$$\widetilde{M}^{t+1}(S) = M^t(S) \tag{3.38}$$

$$\widetilde{M}^{t+1}(D) = 0 \tag{3.39}$$

$$\widetilde{M}^{t+1}(FD) = \frac{M^t(FD) + M^t(F)}{1 - M^t(D)} \tag{3.40}$$

$$\widetilde{M}^{t+1}(SD) = M^t(SD) \tag{3.41}$$

$$\widetilde{M}^{t+1}(FSD) = 1 - \left(\widetilde{M}^{t+1}(S) + \widetilde{M}^{t+1}(FD) + \widetilde{M}^{t+1}(SD)\right). \tag{3.42}$$

We do not predict any freespace, because cells, which are estimated as being free in the current timestamp can be occupied by a moving object in the next timestamp. Static occupied cells are predicted to be static occupied in the next timestamp, as they are not expected to move. The static prediction is not applied to any dynamic occupancy, but uses the mass value of dynamic occupancy to adapt the predicted mass value of being free or dynamic occupied FD. The mass value of being dynamic occupied or free is determined by the mass values of being F or FD. Unclassified occupancy (static or dynamic) is predicted to stay at its place as the static prediction only accounts for the static nature of SD. Lastly, all the masses that are not assigned during the prediction are assigned to FSD.

3.2.2.2 Dynamic Prediction

The dynamic prediction can be divided into several processing steps. First of all, a particle re-sampling is followed by a normalization. Finally, the particles will be predicted in order to transport the dynamic occupancy evidence.

Particle Re-Sampling At the beginning the particle re-sampling takes place. The particles are sampled according to the "weight" of the individual cells. To sample N particles we draw N cells, where those particles will be located in. The likelihood of a cell c being selected is determined by its sampling weight ω_c. The sampling weight is calculated based on the cell's SD and D masses and the age of the last update a_c:

$$\omega_c = \frac{\max\left(8 - a_c\ , 0\right)}{8} \cdot \left(M_c(SD) + M_c(D)\right).$$

The first factor, defined by the age of the last update, ensures that dynamic prediction in areas, which have not been sensed (for a while), will disappear as there are no particles populated there. The second factor, the sum of the static or dynamic and dynamic occupancy mass, ensures that the particles are located in cells where we do expect movement. Areas with only static occupancy or freespace are not populated and therefore no processing power is consumed to predict movement in areas without any dynamic movement. Furthermore, areas which are completely unknown ($M(FSD)$) are not considered for the dynamic prediction. With this particle distribution we focus the dynamic prediction on areas, where (potential) movement occurs.

For every selected cell c we generate a new particle with a probability of $\frac{M_c(SD)}{M_c(SD)+M_c(D)}$, otherwise an existing particle is copied.

A new particle is always located at the center of the selected cell. The velocity vector v_i of that new particle p_i is either sampled using the orientation map (see 3.2.3) or follows a uniform distribution over the possible velocities (see fig. 3.5):

$$\varphi_i = \begin{cases} \operatorname{atan2}(\widetilde{\Phi}_c) & , \widetilde{\Phi}_c \text{ available} \\ \mathcal{U}(0, 2\pi) & , \text{otherwise} \end{cases} \tag{3.43}$$

$$v_i = \begin{pmatrix} \cos(\varphi_i) \\ \sin(\varphi_i) \end{pmatrix} \cdot v_{\max} \cdot \sqrt{\mathcal{U}(0,1)}. \tag{3.44}$$

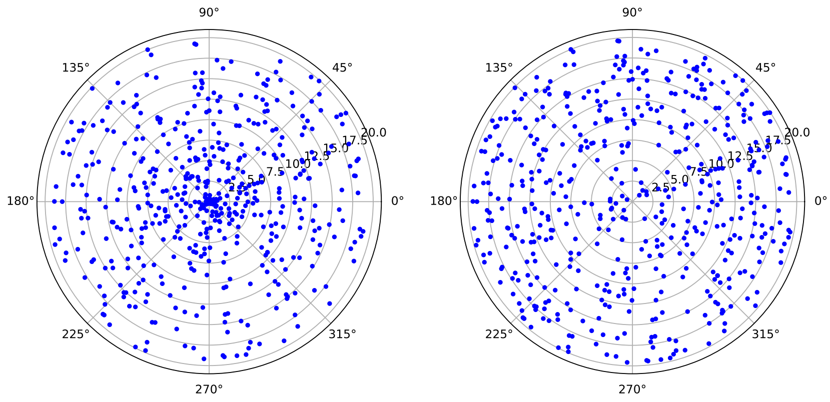


Figure 3.5: Uniform sampling of velocities. The left image is using the uniform sampling $\mathcal{U}(0,1)$ for the absolute velocity value, whereas the right images is using $\sqrt{\mathcal{U}(0,1)}$.

For copying an existing particle, one of the existing particles of that cell is drawn with respect to its weight ω_p.

Particle Normalization After the re-sampling of the particles, the new sampled particles' weights have to be normalized to sum up to one for each cell. To normalize the particles for a particular cell c, the weights of those particles are divided by the sum of all particle weights belonging to that cell:

$$\eta = \sum_{i|c(x_i)=c} \omega_i \tag{3.45}$$

$$\omega_i \leftarrow \eta^{-1} \cdot \omega_i, \forall i | c(x_i) = c \tag{3.46}$$

Particle Prediction For the prediction of the dynamic parts of the map, all particles are predicted using a constant linear motion model, as in [16]:

$$\begin{aligned}
v_i^{t+1} &= v_i^t + \mathcal{N}(0, \sigma_p^2) \\
x_i^{t+1} &= x_i^t + \delta t \cdot v_i^{t+1} \\
a_i^{t+1} &= a_i^t + 1 \\
\omega_i^{t+1} &= \omega_i^t \cdot \left(M_{c(x_i)}^t(D) + M_{c(x_i)}^t(SD) \right),
\end{aligned}$$

where v_i is the velocity vector, x_i the position and a_i the age of particle i. $\mathcal{N}(0, \sigma_p^2)$ denotes a 0-centered Gaussian noise term with variance σ_p^2 and δt the timespan of the prediction. The weight of the particle ω_i is multiplied with the dynamic occupancy mass value of the corresponding cell. This allows us to give particles originating from a cell with high evidence of dynamic occupancy higher weights than particles from cells with low evidence of being (dynamic) occupied.

We use the particle prediction to transport the evidence of being dynamic occupied from their source location to the destination cell. Every particle i transports a portion of the dynamic mass value of its source location proportional to its weight ω_i. In order to allow the transition from dynamic occupancy masses to static occupancy masses we transfer a part of the the dynamic mass D to the static/dynamic mass SD according to the particle's speed value $s = \|v\|_2$ (shown in fig. 3.6):

$$f_S(s) = e^{-\left(\frac{s}{\alpha}\right)^2}. \tag{3.47}$$

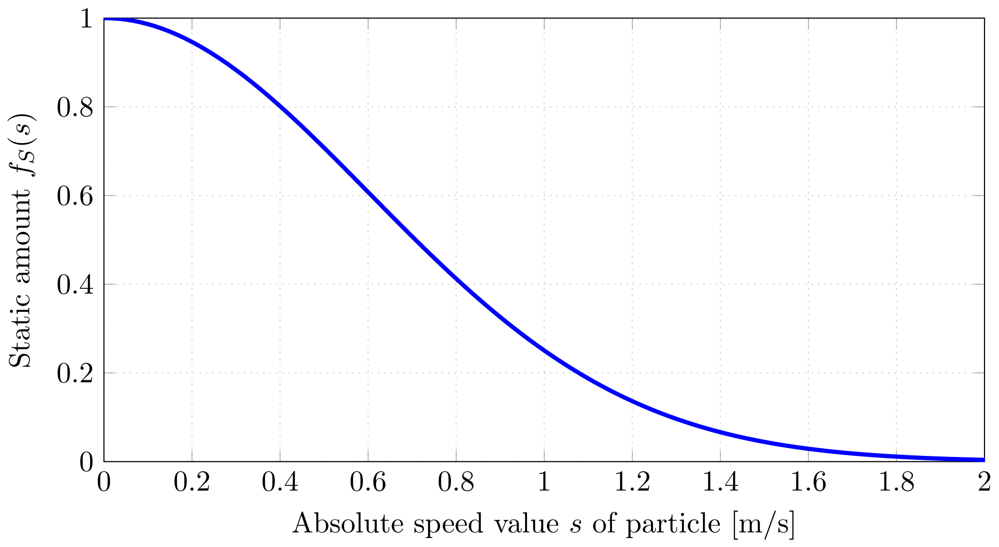


Figure 3.6: Amount of static mass during prediction for given particle speed ($\alpha = 0.85\,\mathrm{m/s}$).

Using the predicted particles and the information about their source x_i^t and destination location x_i^{t+1} the dynamic prediction of the grid cells becomes:

$$\widehat{M}_c^{t+1}(F) = 0 \tag{3.48}$$

$$\widehat{M}_c^{t+1}(S) = 0 \tag{3.49}$$

$$\widehat{M}_c^{t+1}(D) = \sum_{i|c(x_i^{t+1})=c} \left[\omega_i^t \cdot (1 - f_S(v_i^{t+1})) \cdot \left(M_{c(x_i^t)}^t(D) + M_{c(x_i^t)}^t(SD)\right)\right] \tag{3.50}$$

$$\widehat{M}_c^{t+1}(FD) = 0 \tag{3.51}$$

$$\widehat{M}_c^{t+1}(SD) = \sum_{i|c(x_i^{t+1})=c} \left[\omega_i^t \cdot f_S(v_i^{t+1}) \cdot \left(M_{c(x_i^t)}^t(D) + M_{c(x_i^t)}^t(SD)\right)\right] \tag{3.52}$$

$$\widehat{M}_c^{t+1}(FSD) = 1 - \left(\widehat{M}_c^{t+1}(D) + \widehat{M}_c^{t+1}(SD)\right). \tag{3.53}$$

Note that due to the assumed independence of single cells, the predicted dynamic mass values can exceed 1.0, e.g. if particles of two or more cells are predicted into the same destination cell. Therefore the predicted dynamic mass values have to be limited to ensure:

$$\widehat{M}_c^{t+1}(D) \leq 1.0 \tag{3.54}$$

$$\widehat{M}_c^{t+1}(SD) \leq 1.0 \tag{3.55}$$

$$\widehat{M}_c^{t+1}(FSD) \geq 0.0 \tag{3.56}$$

Additionally all particle weights have to be normalized again in order to fulfill 3.15 after the prediction.

3.2.2.3 Combination of Static and Dynamic Prediction

To generate the final predicted grid map $\bar{M}^{t+1}$, given the static $\widetilde{M}^{t+1}$ and dynamic $\widehat{M}^{t+1}$ prediction, we use the standard combination rule again. This combination causes exactly one conflict mass:

$$\zeta_1 = \widetilde{M}^{t+1}(S) \cdot \widehat{M}^{t+1}(D). \tag{3.57}$$

We assign this conflict to the static occupancy mass S, because the evidence for being static occupied is accumulated using several measurements, whereas the dynamic occupancy mass D is estimated using a prediction with some uncertainty. Therefore the final combination of the two predictions can be formulated as:

$$\bar{M}_c^{t+1}(F) = 0 \tag{3.58}$$

$$\bar{M}_c^{t+1}(S) = \left(\widetilde{M}^{t+1} \oplus \widehat{M}^{t+1}\right)(S) + \zeta_1 \tag{3.59}$$

$$\bar{M}_c^{t+1}(D) = \left(\widetilde{M}^{t+1} \oplus \widehat{M}^{t+1}\right)(D) \tag{3.60}$$

$$\bar{M}_c^{t+1}(FD) = \left(\widetilde{M}^{t+1} \oplus \widehat{M}^{t+1}\right)(FD) \tag{3.61}$$

$$\bar{M}_c^{t+1}(SD) = \left(\widetilde{M}^{t+1} \oplus \widehat{M}^{t+1}\right)(SD) \tag{3.62}$$

$$\bar{M}_c^{t+1}(FSD) = \left(\widetilde{M}^{t+1} \oplus \widehat{M}^{t+1}\right)(FSD). \tag{3.63}$$

3.2.3 Online Generated Orientation Prior

In order to optimize the initialization of new particles we propose to add an online generated orientation prior. This prior information about the orientation is used during the generation of new particles (eq. 3.43). Figure 3.7 shows the impact of having such a prior information on the distribution of newly created particles. With the majority of new particles pointing in the corrected direction of movement (e.g. if a new object is entering the field of view) a much faster convergence in velocity estimation can be achieved. This leads to a better static/dynamic classification and velocity estimation for the grid cells.

We store this orientation information as unit vector $\widetilde{\Phi} = \left(\widetilde{\Phi}_x, \widetilde{\Phi}_y\right)^T$ for every cell. This orientation vector indicates the direction of movement of (possible) objects at the cell location. If there is no prior orientation information available for one cell, the orientation vector is set to zero. As we do no want to use any additional information, such as map data, we need to calculate this prior information continuously within the existing framework.

After each map prediction step (3.2.2), we will update the online estimated orientation vector. To calculate the new orientation estimate, we use the orientation information of the previous timestamp Φ^t and the current estimation of the cell velocity (eq. 3.16),

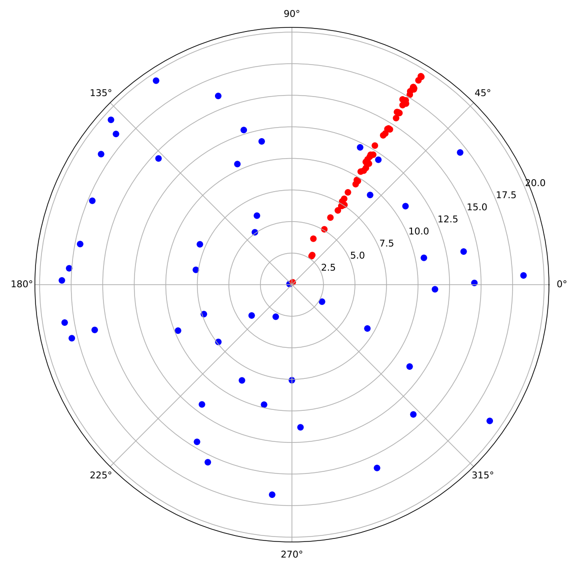
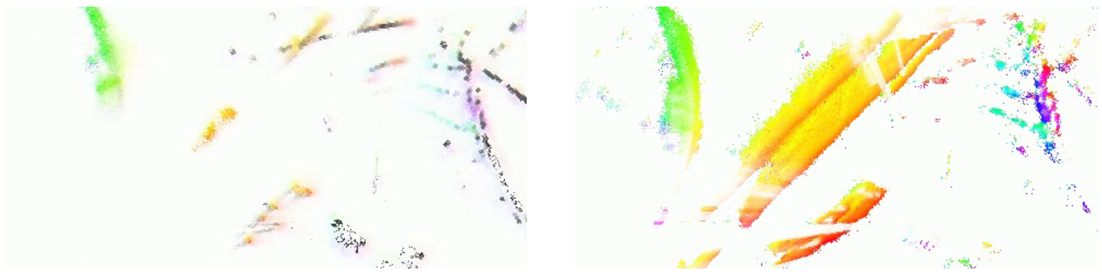


Figure 3.7: 50 uniform sampled particles (blue) and 50 particles sampled using the prior orientation (red).

weighted by a fixed decay factor γ and the dynamic mass of the cell $M(D)$:

$$\hat{\Phi}_c^{t+1} = \Phi_c^t + \gamma \cdot M_c^{t+1}(D) \cdot \frac{v_c}{\|v_c\|} \tag{3.64}$$

$$\Phi_c^{t+1} = \frac{\hat{\Phi}_c^{t+1}}{\|\hat{\Phi}_c^{t+1}\|}. \tag{3.65}$$



(a) Current orientation of v of cells. **(b)** Current orientation map Φ of cells.

Figure 3.8: Color encoded angles of current velocity estimation and orientation map.

Figure 3.8 shows an example of the current velocity estimation of the dynamic grid and the generated orientation map. For cells, which are considered static ($M(S) > \tau_S$), the orientation estimation is not performed, as single false estimated particles would lead to pointless orientation estimates for future particles in those areas. Furthermore, the orientation of the remaining cells is only updated if the estimated velocity is above a certain threshold ($v_c > \tau_v$). At low speeds the orientation based on the velocity vector is very error prune, as small errors in the velocity estimation can lead to rather huge errors in the orientation.

To further improve the generated orientation map, we apply a Gaussian kernel on the estimated orientation map Φ, resulting in a smoothed version and our final online generated orientation prior map $\widetilde{\Phi}$ (see fig. 3.9). Applying this smoothing operation removes single outliers and closes potential gaps (cells, where no orientation could be estimated). Note that cells without any orientation estimation (or $\Phi_c = 0$) have to be excluded in the calculation of the "blurred" orientation, to prevent negative influences on the result.

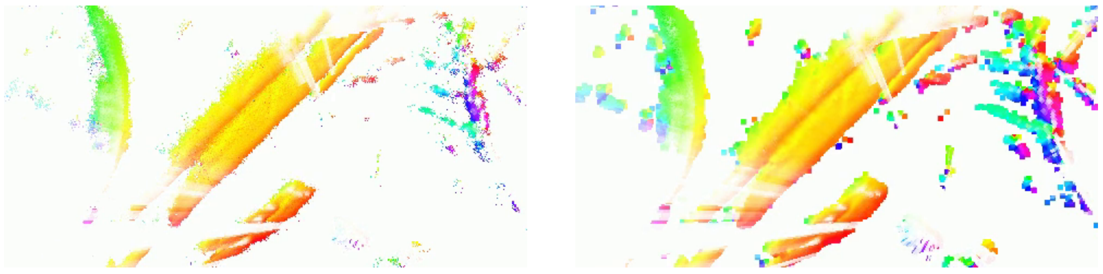

(a) Current orientation map Φ of cells. **(b)** Current smoothed orientation map $\widetilde{\Phi}$ of cells.

Figure 3.9: Color encoded angles of orientation map before and after smoothing.

The online generated orientation map $\widetilde{\Phi}$ is used as prior information during the initialization of new particles, as described in eq. 3.43. The advantages of using such a prior information is shown during evaluation in 6.3.2.2 and 6.3.3.

3.2.4 Memory Consumption in Two Dimensions

Let's consider the same setup as in 3.1.5:

- the grid map covering an area of $120\,\mathrm{m} \times 120\,\mathrm{m}$ around the vehicle and
- a cell-size of $0.125\,\mathrm{m} \times 0.125\,\mathrm{m}$.

For every grid cell we need to store the mass values for F, S, D, FD and SD (FSD can be calculated using the others). Additionally we store the two coordinates of the orientation prior $\widetilde{\Phi}$ leading to 7 values per cell.

We use the same number of particles as cells. For each particle we store the age, the x/y position, the x/y velocity vector and the weight of the particle, leading to 6 values per cell.

Using this configuration we end up with 13 values per cell, which leads to an overall memory consumption of:

$$\begin{aligned}\text{MemoryRequirement} &= 13 \cdot N \cdot 4\,\mathrm{B} &(3.66)\\ &= 13 \cdot 921\,600 \cdot 4\,\mathrm{B} &(3.67)\\ &= 47\,923\,200\,\mathrm{B} &(3.68)\\ &\approx 46\,\mathrm{MiB}. &(3.69)\end{aligned}$$

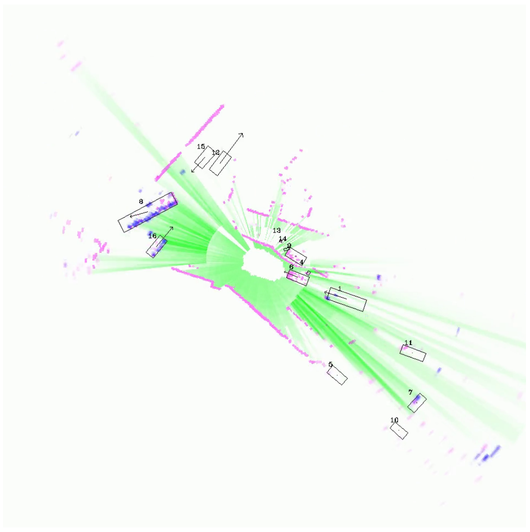
Using 46 MiB of memory is a magnitude less than the 1.1 GiB required by the Bayesian approach. Processing of 46 MiB in one cycle is absolutely feasible using modern computers and upcoming automotive ECUs.

3.2.5 Example

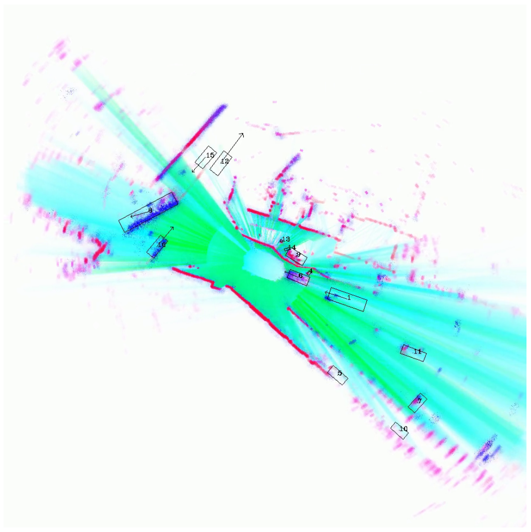
The example shown in figure 3.10 is based on *scene-0757* of the used dataset during the evaluation (6.1). The ego vehicle, located in the center of the images is driving straight towards the upper left corner of the image. The ego vehicle is approaching an intersection with crossing traffic (all ground-truth objects are drawn with black bounding boxes).

As shown in figure 3.10b, the visible dynamic objects behind the ego vehicle (1 and 6) and the two crossing objects (8 and 16) are classified as dynamic occupied (drawn in blue). Most of the structures next to the road are classified as static occupied (red), whereas the road itself is classified as free/drivable (green and cyan).

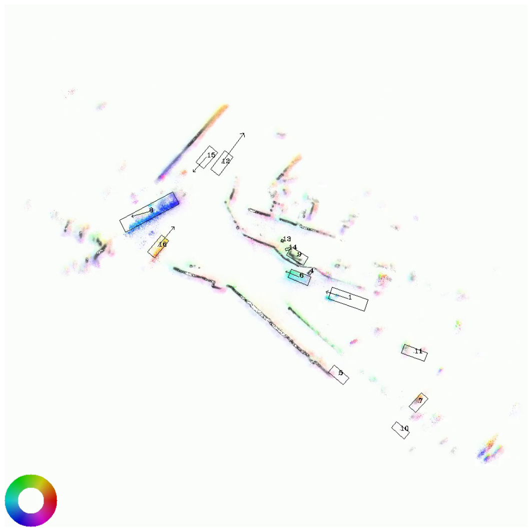
Figure 3.10c shows the orientation of the current estimated velocity vector of the cells. The orientation of this estimation is matching the velocity vectors of the ground-truth objects. The generated orientation prior is shown in figure 3.10d.



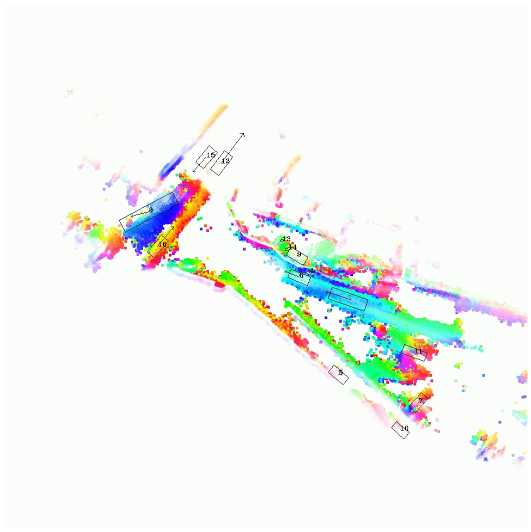
(a) Input sensor grid Z^t, using three Radar and one Lidar sensors. Freespace F is shown in green, dynamic occupancy D in blue, (static or dynamic) occupancy SD in purple and unknown area FSD in white.



(b) Updated grid map M^t. Same coloring used as in 3.10a, with additionally FD (freespace or dynamic occupancy) shown in cyan and static occupancy S shown in red.



(c) Orientation of the estimated velocities. Coloring of the angles as shown by the color wheel in the bottom left corner.



(d) Online generated orientation prior. Using the same coloring as in figure 3.10c.

Figure 3.10: Screenshots of the dynamic grid map using data of scene *scene-0757*. Ground-truth objects are shown in black for reader's understanding.

4 Freespace Extraction

Next to the list of tracked objects (chapter 5), freespace contours are the most important input for driver assistance systems and autonomous driving applications. With the freespace contours the area, which is safe to drive, can be described in a very compressed manner and is often used to describe the static environment around a vehicle in contrast to the dynamic environment of the tracked object list.

The freespace extraction in the dynamic grid does not differ from the standard freespace extraction used for static grid maps. The only slight difference is the summation of masses to get the freespace "probability" to be used for extraction, as dynamic occupancy can be easily integrated into the extracted freespace.

Figure 4.1 shows the output of the dynamic grid (chapter 3) which acts as input for the freespace extraction algorithm. The vehicle (located in the center of the image) is driving towards the upper left and approaching a traffic junction with crossing objects. The current road is flanked by a building on the left and a fence on the right (both shown as solid red-ish lines). The dynamic crossing vehicles, as well as one vehicle following the ego are shown as blue pixels.

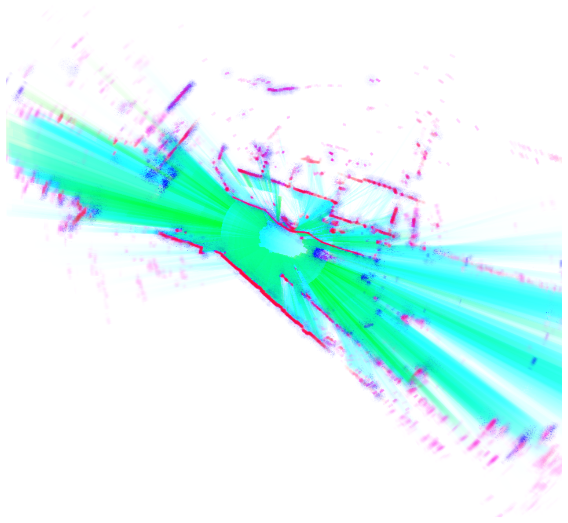


Figure 4.1: Input: Dynamic grid map. Green illustrates freespace $M(F)$, red static occupancy $M(S)$, blue dynamic occupancy $M(D)$, cyan free or dynamic occupancy $M(FD)$, purple static or dynamic occupancy $M(SD)$ and white unknown cells $M(FSD)$.

In the first step belief values for being free or occupied are extracted from the dynamic grid output. In the case of freespace detection we interpret dynamic occupied cells as being free, since we are mainly interested in freespace defined as general drivable area instead of actual empty areas. Therefore we define the belief of being free as:

$$bel(\text{free}) = M(F) + M(D) + M(FD). \tag{4.1}$$

Static occupied cells on the contrary are not allowed to be part of the extracted freespace. In the case of not knowing whether occupied cells are static or dynamic occupied, we choose the conservative interpretation and define the occupancy belief accordingly:

$$bel(\text{occ}) = M(S) + M(SD). \tag{4.2}$$

The extracted free and occupied beliefs are shown in the following figure 4.2.

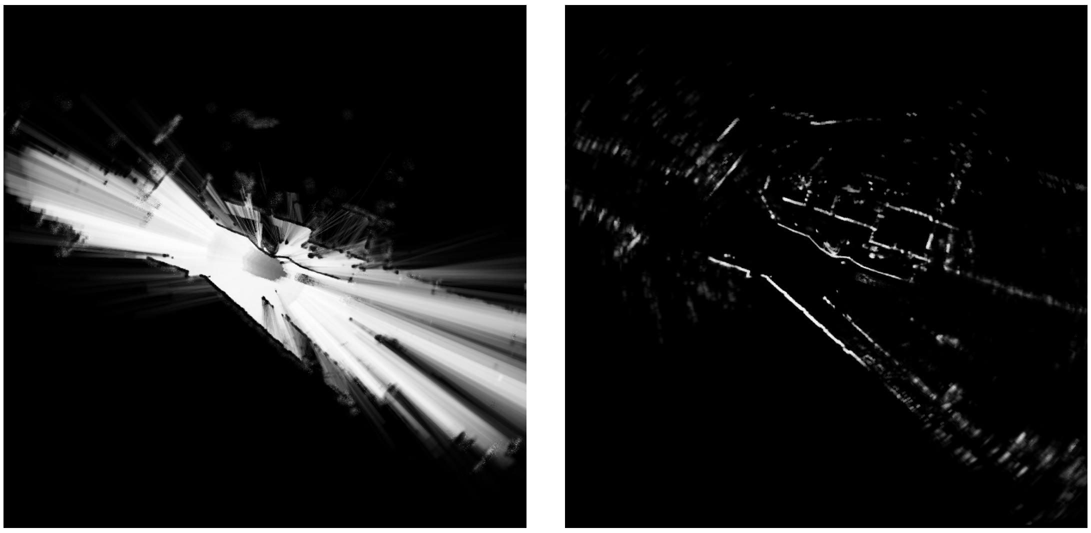

(a) Extracted freespace belief *bel*(free). **(b)** Extracted occupancy belief *bel*(occ).

Figure 4.2: Extracted values for being freespace or occupied area. Black pixels corresponds to values of 0, white pixels to values of 1.

In order to find the contours of the freespace we generate binary images from the freespace and occupancy belief images by applying a thresholding operation. Note that we are interested in extracting the freespace in the end, so the occupancy *image* gets inverted to represent a *not occupied* mask. The result of the operation is shown in figure 4.3:

$$B(\text{free}) = (bel(\text{free}) > \tau_{\text{free}}) \tag{4.3}$$
$$B(\neg\text{occ}) = (bel(\text{occ}) < \tau_{\text{occ}})\,. \tag{4.4}$$

To filter false-negatives from the free mask and to close small gaps (e.g. single cells not sensed or below the threshold τ_{free}) a morphological *close* transformation [35] is applied to the binary mask $B(\text{free})$ (fig. 4.4a). To prevent occupied cells from being

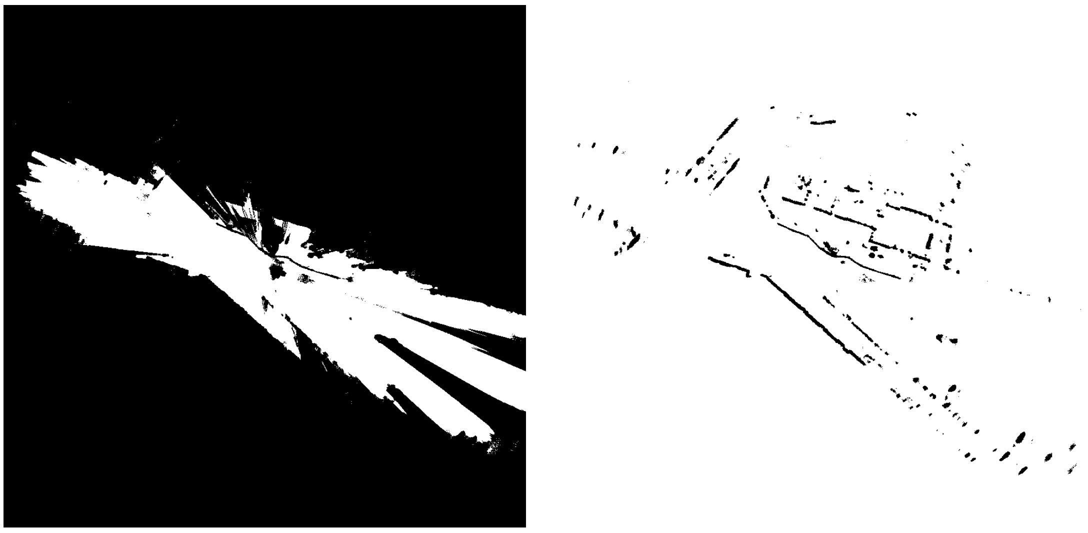

(a) Binary mask of being free $B(\text{free})$. **(b)** Binary mask of being not occupied $B(\neg\text{occ})$.

Figure 4.3: Binary mask for being free $B(\text{free})$ and not occupied $B(\neg\text{occ})$. White values represent *true*, black values *false*.

contained in the freespace mask, the resulted binary mask after the *close* transformation will be masked by the *not occupied* mask later. With the same argumentation as for the free mask, a morphological *erode* transformation [36] is applied to the *not occupied* mask. This *erode* transformation also ensures a certain safety margin around occupied obstacles.

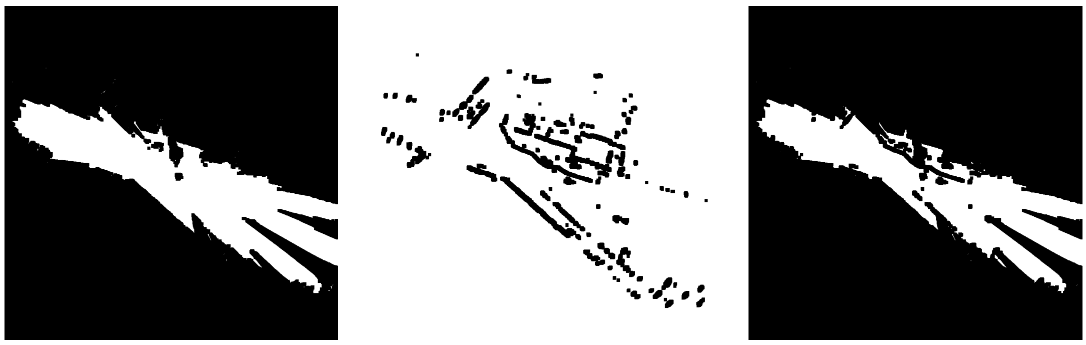
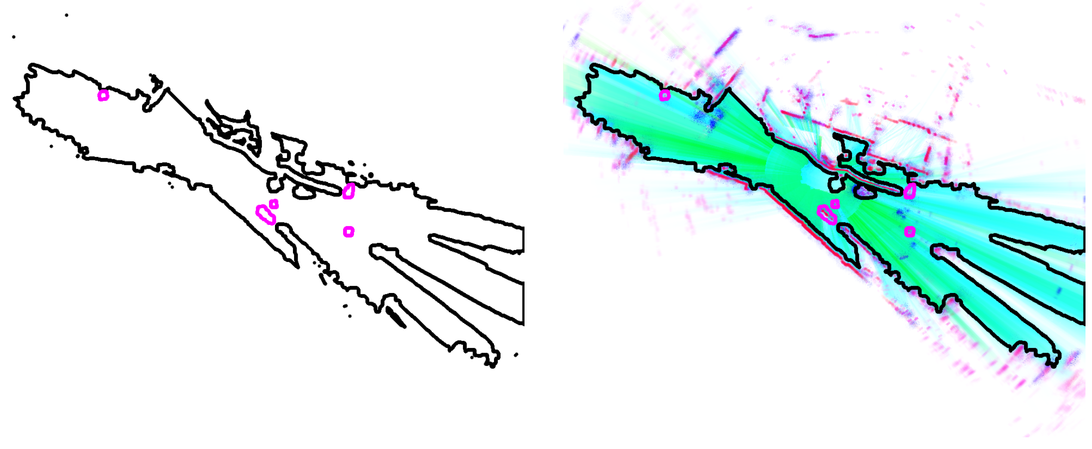

(a) Extracted freespace, after applying the morph operation. **(b)** Extracted occupied space, after applying the morph operation. **(c)** Final binary map for freespace contour extraction.

Figure 4.4: Freespace and occupied areas after applying the morph operations and combining both masks.

Figure 4.4 shows the two binary masks $B(\text{free})$ and $B(\neg\text{occ})$ after applying the morphological transformations as well as the combined mask $B = B(\text{free}) \wedge B(\neg\text{occ})$.



(a) Extracted Contours. **(b)** Final contours drawn on the input grid.

Figure 4.5: Extracted freespace contours. Contours indicating freespace are drawn in black, whereas the occupied "holes" are shown in purple.

On the final binary mask B, a state of the art contour extraction method (e.g. [37]) is applied to get the freespace contours as shown in figure 4.5a. As freespace areas, which can not be reached from the current ego position, are of no interest for autonomous driving functions, those contours are filtered out. Figure 4.5b shows the final result drawn on the original input data.

5 Object Tracking

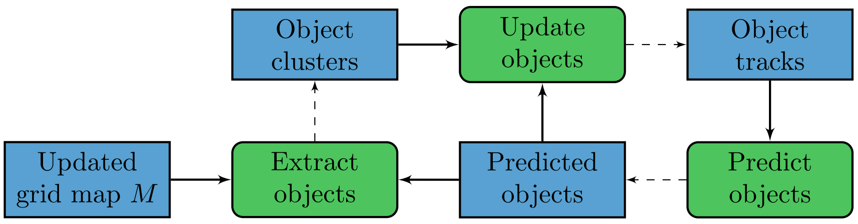


Figure 5.1: System overview for object tracking

The object tracking part takes the updated grid map M as input. This input contains information about the estimation of dynamic/static occupied as well as velocity vectors for each cell (see chapter 3). From the given input data, cells are extracted and clustered to form the object measurements. This process is described in detail in section 5.1. The extracted object measurements serve as input for the actual object tracker, described in section 5.2. Figure 5.1 illustrates the architecture of the object tracking.

The overall goal of this part of the work is to provide autonomous driving functions or ADAS system a sparse and established format to deal with dynamic objects: a *dynamic object list*. Another advantage of this output format compared to the grid map itself lies in the higher *smoothness* of the object tracks due to the additional use of a vehicle kinematics model. Furthermore, predictions of dynamic objects can be handled more efficiently than predictions of the grid map.

5.1 Object Extraction

The object extraction part takes the updated grid map (see 3) as input (see fig. 5.2) and tries to extract object clusters. These object clusters serve as input measurements for the actual object tracking. Additionally we use predicted object tracks from the last cycle as supportive input hints. As the object tracking is focused on tracking dynamic objects, the extraction part only extracts static cells, if they provide information regarding an already existing object track.

Cell Extraction In the first step the belief values for being occupied and being dynamic occupied are extracted (figures 5.3a and 5.3b):

$$\begin{aligned} bel(\text{occ}) &= M(S) + M(D) + M(SD) \\ bel(\text{dyn.occ}) &= M(D). \end{aligned}$$

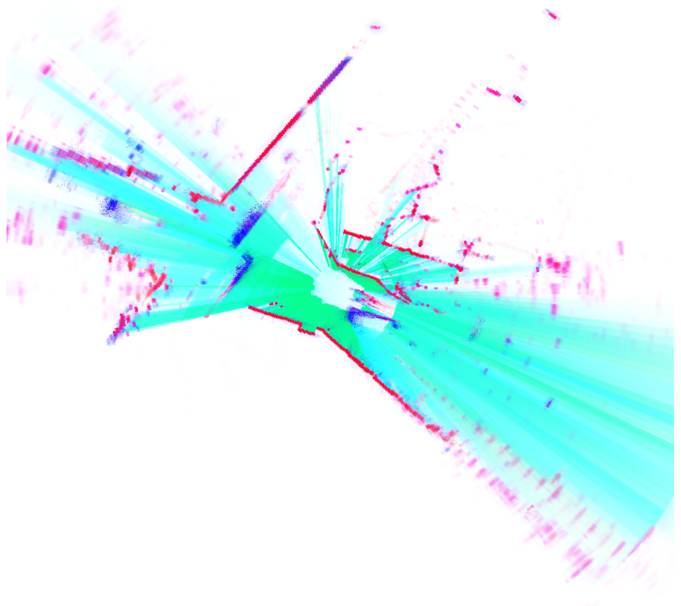


Figure 5.2: Input: Dynamic grid map. Green illustrates freespace $M(F)$, red static occupancy $M(S)$, blue dynamic occupancy $M(D)$, cyan free or dynamic occupancy $M(FD)$, purple static or dynamic occupancy $M(SD)$ and white unknown cells $M(FSD)$.

In order to extract the object clusters we generate binary images from the extraction by applying simple thresholding (figures 5.3c and 5.3d):

$$\begin{aligned} B(\text{occ}) &= (bel(\text{occ}) > \tau_{\text{occ}}) \\ B(\text{dyn.occ}) &= (bel(\text{dyn.occ}) > \tau_{\text{dyn.occ}}) \,. \end{aligned}$$

Clustering On the binary mask of being occupied $B(\text{occ})$ (fig. 5.3c) we perform a state-of-the-art connected component algorithm (e.g. [38]) in order to get our cell clusters, figure 5.4a shows an example of resulting clusters. The generated clusters also contain a lot of static structures (e.g. buildings). Therefore we filter out all clusters, which contain less than 25% of cells being included in the input hints or being considered as dynamic occupied cells, as defined with $B(\text{dyn.occ})$ (figure 5.4b). Keeping static clusters, which are located where we expect already tracked objects, allows us to keep tracking objects, which become temporarily static, e.g. when they stop in front of a traffic light.

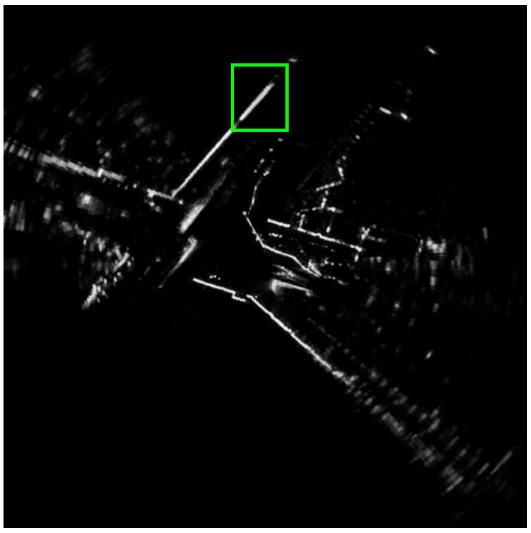
The filter method using only occupancy clusters with a certain amount of dynamic cells instead of only extracting dynamic occupied cells has the advantage of reducing false positives in the object extraction process. The green rectangles in figures 5.3 and 5.4 show a (static) wall as an example. Parts of the wall have been estimated as dynamic occupied (due to wrong sensor measurements), but due to the high amount of static occupancy estimation in the structure, this cluster is not considered a dynamic object.

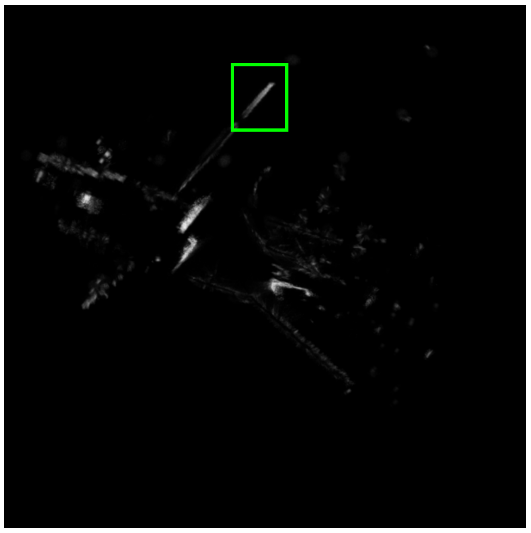
Bounding Box Estimation For the estimation of the cluster's bounding box, we first estimate the cluster's velocity vector by taking a weighted mean value of all cells c in the cluster C:

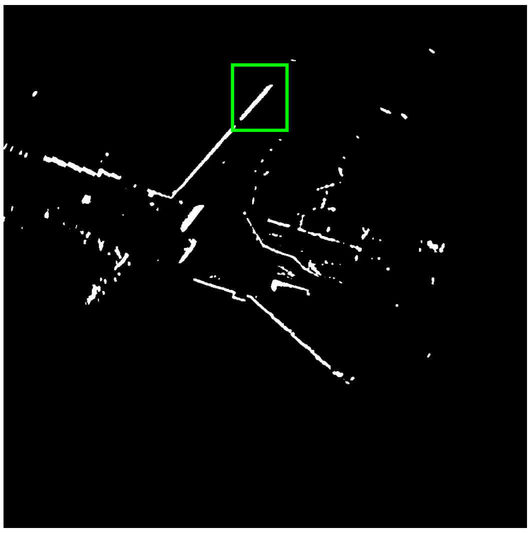
$$\hat{v}_C = \eta \cdot \sum_{c \in C} (M_c(D) \cdot v_c)$$

We use the orientation of the estimated velocity vector as orientation of the estimated bounding box. In cases, where no velocity can be estimated or the absolute value of the estimated velocity vector is very low, we use a rotating calipers approach [39] to find a rotated rectangle enclosing all cells of the cluster, with minimum area.

Once the orientation of the resulting bounding box is fixed, the estimation of the length and width is a trivial task, which leads to the final extracted object measurement as shown in figure 5.4c.


(a) Extracted occupancy belief *bel*(occ).


(b) Extracted dynamic occupancy belief *bel*(dyn.occ).


(c) Binary mask of being occupied B(occ).

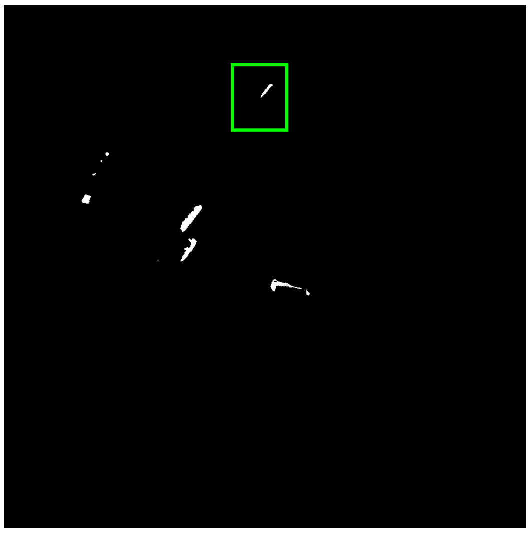
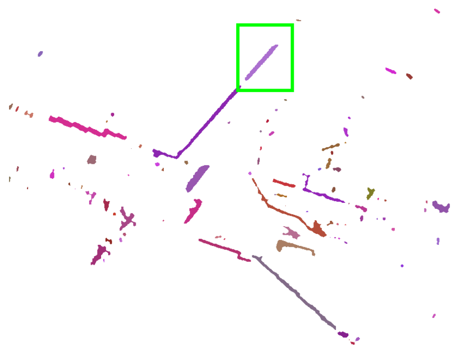
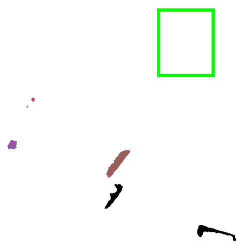

(d) Binary mask of being dynamic occupied B(dyn.occ).

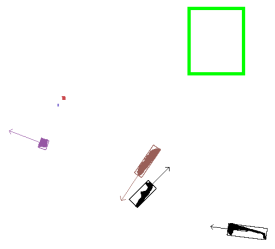
Figure 5.3: Extracted values and binary masks for being occupied and dynamic occupied areas. Black pixels corresponds to values of 0 or *false*, white pixels to values of 1 or *true*. The green rectangle highlights a true negative object extraction (see also fig. 5.4).



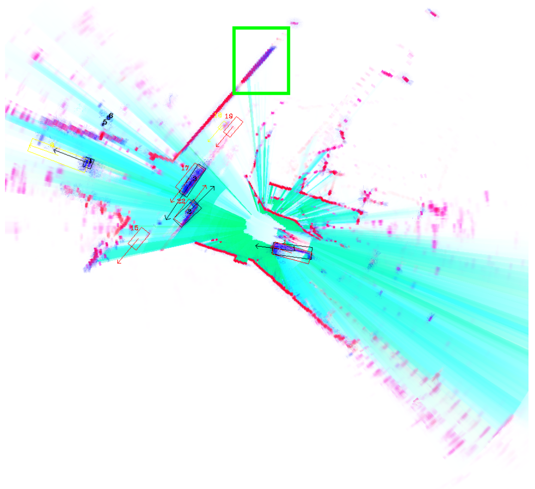
(a) Cell clusters using the connected component method.



(b) Cell clusters after filtering



(c) Cell clusters with estimated bounding box and velocity vector.



(d) Extracted objects (black) and ground-truth object list (red) drawn on the input grid.

Figure 5.4: Extracted cell clusters, before and after the filtering for dynamic clusters and final measurement result. The green rectangle highlights a true negative object extraction.

5.2 Object Tracking

After the object extraction step, described in the previous section, the extracted object measurements are used as input for the actual object tracking part. The goal of the object tracking is to enhance the quality of the object list and to reduce false positives. Furthermore, the object tracking assigns track ids, which allows association of the same object over time.

5.2.1 Object Track

For the object tracking itself we use an extended Kalman filter (EKF) with the following state vector:

$$\boldsymbol{x} = \begin{pmatrix} x \\ y \\ v \\ \Psi \\ a \\ \dot{\Psi} \end{pmatrix},$$

with x, y being the object's center position in world coordinates [m], v being the absolute velocity [m/s], Ψ being the object's orientation, a being the longitudinal acceleration [m/s^2] and $\dot{\Psi}$ being the change of the orientation over time [1/s]. Additional following information are updated separately and are therefore stored next to the object state:

- ID: unique track identifier,
- Last match: time of the last update,
- Existence probability: Probability that the object really exists,
- Length: object's length in meter and
- Width: object's width in meter.

5.2.2 Association

The goal of the association step is to decide which measurement corresponds to which predicted object. This association is required for selecting the correct measurements for updating the predicted tracks, as well as to decide which measurements are not related to any tracked objects and therefore create new object tracks.

Due to the preprocessing of the dynamic grid (chapter 3) and the clustering and object extraction (section 5.1) we already get good object measurements, especially in terms of their position (see 6.3.3). Therefore we use the intersection over union (IoU) as metric to calculate our distance measure between object tracks and measurements. To compute

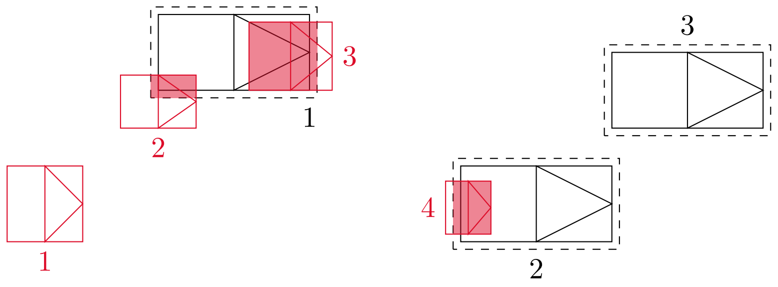


Figure 5.5: Three arbitrary object tracks shown in black and four object extractions shown in red. The intersection area of tracks with measurements is filled with red.

the intersection over union of one predicted object i and the measurement j, the area of overlap has to be divided by the area of the union:

$$\mathrm{IoU}_{ij} = \frac{A_i \cap A_j}{A_i + A_j - (A_i \cap A_j)}. \tag{5.1}$$

In order to compensate a certain amount of error in the prediction, we add a fixed margin μ to the tracked object length and width for the IoU calculation, figure 5.5 shows object predictions in black with the added margin as dashed bounding boxes. The measurements are drawn in red and the intersecting area is also filled in red. The considered areas for the predicted objects and measurements are given by their bounding boxes defined by the length l and width w.

We defined the distance between a predicted object i and the measurement j as the opposite of the IoU:

$$d_{i,j} = 1 - \mathrm{IoU}_{ij}. \tag{5.2}$$

With the distances between all predicted objects and all measurements we choose a greedy algorithm for the actual association:

Algorithm 2: Association algorithm for object updates

```
1 while $d_{i,j} < 1$ exists do
2     Select $\hat{\imath}, \hat{\jmath} = \arg\min_{i,j} d_{i,j}$;
3     Update tracked object $\hat{\imath}$ with measurement $\hat{\jmath}$;
4     Prevent further updates of object $\hat{\imath}$: $d_{\hat{\imath},j} = 1$;
5     Prevent further updates with measurement $\hat{\jmath}$: $d_{i,\hat{\jmath}} = 1$;
6 end
```

All object measurements without any overlap with predicted tracks are used to initialize new object tracks.

Taking the example of figure 5.5 the distance matrix between the 3 predicted object tracks and the 4 measurements will look like

$$D = \begin{pmatrix} 1 & 0.94 & 0.81 & 1 \\ 1 & 1 & 1 & 0.87 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \tag{5.3}$$

resulting in object 1 being updated with measurement 3 and object 2 being updated using measurement 4. Object 3 is not receiving any update and with measurement 1 a new track will be initialized. Measurement 2 will be ignored, because measurement 3 is the one with a smaller distance to object 1.

5.2.3 Track Management

Within the track management the lifetime of object tracks is managed. In this section we describe how new object tracks are created and how existing object tracks are deleted or not considered anymore. Therefore the object track's existence probability is used.

Track Birth We create a new object track, if an object measurement cannot be associated to any existing track and the measurement is not considered to be a static object. We want to avoid using the object tracking framework for arbitrary static objects, which are already covered by the grid map. Object measurements are considered to be static if their estimated absolute velocity is below a certain threshold or if the sum of static mass S is exceeding the sum of the dynamic mass D within the cells in the measurement.

For object measurements not classified as static, a new object track is instantiated with the measurement acting as initial state vector, length and width for the new track. Additionally, the track is assigned a new unique ID and its existence probability $P(E)$ is set to an initial value of 0.5.

Existence Probability The existence probability of each track is updated during the objects update cycle. The goal is to estimate the object existence probability E given the object measurements up to the current time $Z^{1:t}$. This probability estimation corresponds to the estimation of the occupancy of a single cell in the Bayesian occupancy grid (chapter 2.1). Therefore we can use the same update rule in logarithmic version as in eq. 2.3:

$$l\left(E|Z^{1:t}\right) = \underbrace{l\left(E|Z^{t}\right)}_{\text{Update term}} + \underbrace{l\left(E|Z^{1:t-1}\right)}_{\text{Previous estimation}} . \tag{5.4}$$

In our implementation we use two constant terms for updating the existence probability of an object track. If the track is associated with one of the current measurements, the existence probability is increased, otherwise the existence probability is decreased.

Objects not exceeding a probability threshold are suppressed until the threshold is reached by consecutive measurements. This means a new initiated track has to be confirmed by some consecutive measurement to track associations before the track will be part of the algorithm's output object list. For tracks to be considered as object hints in the extraction step (section 5.1) an even higher threshold has to be reached. This is, because we only want to influence the extraction step, if we are certain about an object's existence in order to prevent false-positives from confirming themselves.

Track Death Object tracks with an existence probability below a predefined threshold are removed from the list and not considered anymore. Additionally we remove tracks existing for a given time interval, but without any change in their positions. The second case prevents static objects in the track list, which have been added due to a wrong dynamic classification by the dynamic grid (chapter 3).

5.2.4 Update

During the track update step the predicted object track will be updated with the information of the associated object measurement.

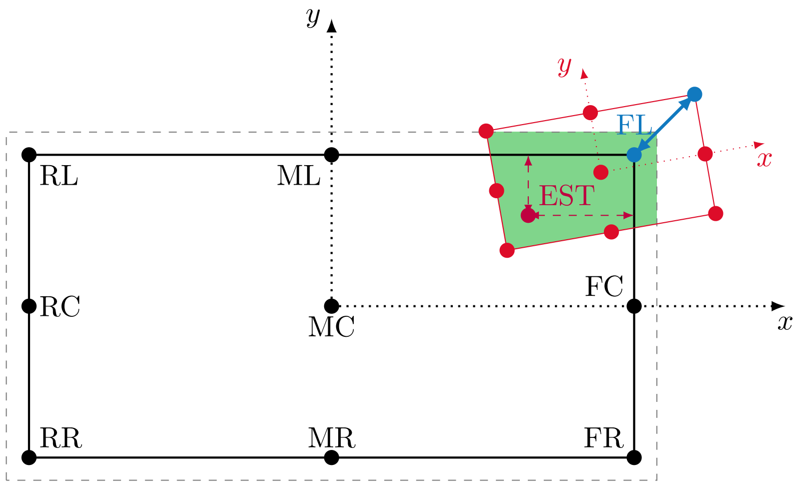


Figure 5.6: Object is shown in black with its 9 reference points: (R)ear/(M)iddle/(F)ront - (L)eft/(C)enter/(R)ight. The dashed gray rectangle shows the area of the object considered for the intersection over union calculation. The red bounding box shows a potential measurement for that object. The best matching reference point is highlighted in blue, whereas the intersection area with the object is shown in green. The measurement estimation point EST is shown in purple.

Reference Point In the general case the measured object will not cover the complete object, but only a part of it. Normally only one side/edge or a corner of that object can be detected at a given time step. Therefore it is not possible to simply update the object's center with the measured position. To compensate this effect, we use nine reference points on the object track and measurement: the four corners, the center of the four sides and the center of the bounding box (see figure. 5.6). When updating the object we use the reference point, which has the smallest Euclidean distance between the object track and the measurement.

Length and Width Updates The length and width of the tracked objects are not part of the EKF state, as their update is treated differently. We do expect to see only part of the objects at a time, meaning that the measured dimension of the object is not normally distributed and therefore not suited to be tracked with a Kalman Filter. In our implementation we use some sort of exponential smoothing, where the smoothing factor depends firstly on the age of the tracked object a_{object} and secondly on the measurement having a larger or smaller length/width than our tracked object:

$$\alpha\left(a_{\text{track}}, d_{\text{track}}, d_{\text{meas}}\right) = \begin{cases} \frac{1}{a_{\text{track}}} & , d_{\text{track}} \leq d_{\text{meas}} \\ \frac{1}{10 \cdot a_{\text{track}}} & , d_{\text{track}} > d_{\text{meas}} \end{cases},$$

where d is one of the dimensions, either length or width. The update of the track's dimension is then calculated as follows:

$$d_{\text{track}}^{t+1} = (1-\alpha)\, d_{\text{track}}^{t} + \alpha \cdot d_{\text{meas}}$$

State Update The update of the object state $\boldsymbol{x^t}$ and the corresponding covariance matrix $\boldsymbol{P^t}$ is defined by the EKF in the following way:

$$\boldsymbol{y^{t+1}} = \boldsymbol{z^{t+1}} - h\left(\boldsymbol{\bar{x}^{t+1}}\right) \qquad \text{(Innovation)} \tag{5.5}$$

$$\boldsymbol{S^{t+1}} = \boldsymbol{H^{t+1}}\boldsymbol{\bar{P}^{t+1}}\left(\boldsymbol{H^{t+1}}\right)^T + \boldsymbol{R^{t+1}} \qquad \text{(Innovation covariance)} \tag{5.6}$$

$$\boldsymbol{K^{t+1}} = \boldsymbol{\bar{P}^{t+1}}\left(\boldsymbol{H^{t+1}}\right)^T\left(\boldsymbol{S^{t+1}}\right)^{-1} \qquad \text{(Kalman gain)} \tag{5.7}$$

$$\boldsymbol{x^{t+1}} = \boldsymbol{\bar{x}^{t+1}} + \boldsymbol{K^{t+1}}\boldsymbol{y^{t+1}} \qquad \text{(Updated state)} \tag{5.8}$$

$$\boldsymbol{P^{t+1}} = \boldsymbol{\bar{P}^{t+1}} - \boldsymbol{K^{t+1}}\boldsymbol{H^{t+1}}\boldsymbol{\bar{P}^{t+1}}, \qquad \text{(Updated state covariance)} \tag{5.9}$$

with h being a non-linear observation function, which estimates the measurement vector $\boldsymbol{z^{t+1}}$ given the predicted state vector $\boldsymbol{\bar{x}^{t+1}}$. $\boldsymbol{H^{t+1}}$ presents the Jacobian matrix of the function h. The predicted state covariance matrix is given as $\boldsymbol{\bar{P}^{t+1}}$ and the observation covariance matrix is given as $\boldsymbol{R^{t+1}}$.

In order to update our object state vector according to the EKF update rule, we have to define the observation function h, which has to estimate the measurement's center based on the state vector $\boldsymbol{\bar{x}^{t+1}}$ and the additional information stored in our object track. We use the information of the best matched reference point r and the length and width of the tracked object $(l_{\text{track}}, w_{\text{track}})$ as well as the measurement bounding box $(l_{\text{meas}}, w_{\text{meas}})$ as additional parameters for h.

The goal is to estimate the measurement at the estimation point EST (compare figure 5.6), which is located at the same offset to the reference point of the tracked object, as it has on the measurement bounding box. We therefore define a helper function p (eq. 5.10), which returns the coordinate of the reference point relative to the object/measurement's center point.

$$p(l,w,r) = \begin{cases} \left(\frac{l}{2}, \frac{w}{2}\right)^T & \text{, if r} = \text{FL} \\ \left(0, \frac{w}{2}\right)^T & \text{, if r} = \text{ML} \\ \left(-\frac{l}{2}, \frac{w}{2}\right)^T & \text{, if r} = \text{RL} \\ \left(-\frac{l}{2}, 0\right)^T & \text{, if r} = \text{RC} \\ \left(-\frac{l}{2}, -\frac{w}{2}\right)^T & \text{, if r} = \text{RR} \\ \left(0, -\frac{w}{2}\right)^T & \text{, if r} = \text{MR} \\ \left(\frac{l}{2}, -\frac{w}{2}\right)^T & \text{, if r} = \text{FR} \\ \left(\frac{l}{2}, 0\right)^T & \text{, if r} = \text{FC} \\ (0,0)^T & \text{, if r} = \text{MC} \end{cases} \tag{5.10}$$

The estimation point EST (in object coordinates), given the best matching reference point r, can then be expressed as:

$$\text{EST}_{\text{relative}} = \begin{pmatrix} e_x \\ e_y \end{pmatrix} \tag{5.11}$$

$$= p\left(l_{\text{track}}, w_{\text{track}}, r\right) - p\left(l_{\text{meas}}, w_{\text{meas}}, r\right). \tag{5.12}$$

Transforming this track-relative coordinate into world coordinates (eq. A.2) allows us to formulate the final observation function h:

$$h\left(\bar{\boldsymbol{x}}^{\boldsymbol{t+1}}\right) = \begin{pmatrix} \tilde{x}^{t+1} \\ \tilde{y}^{t+1} \\ \tilde{v}^{t+1} \\ \tilde{\Psi}^{t+1} \end{pmatrix} \tag{5.13}$$

$$= \begin{pmatrix} \cos\left(\bar{\Psi}^{t+1}\right) e_x - \sin\left(\bar{\Psi}^{t+1}\right) e_y + \bar{x}^{t+1} \\ \sin\left(\bar{\Psi}^{t+1}\right) e_x + \cos\left(\bar{\Psi}^{t+1}\right) e_y + \bar{y}^{t+1} \\ \bar{v}^{t+1} \\ \bar{\Psi}^{t+1} \end{pmatrix}. \tag{5.14}$$

The corresponding Jacobian $\boldsymbol{H}^{\boldsymbol{t+1}}$ is then defined as:

$$\boldsymbol{H}^{\boldsymbol{t+1}} = \begin{pmatrix} 1 & 0 & 0 & -\sin\left(\bar{\Psi}^{t+1}\right) e_x - \cos\left(\bar{\Psi}^{t+1}\right) e_y & 0 & 0 \\ 0 & 1 & 0 & \cos\left(\bar{\Psi}^{t+1}\right) e_x - \sin\left(\bar{\Psi}^{t+1}\right) e_y & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}. \tag{5.15}$$

For the observation noise covariance matrix $\boldsymbol{R}^{\boldsymbol{t+1}}$ we use a diagonal matrix with predefined variances for the single measurement components.

5.2.5 Prediction

To predict the object state $\boldsymbol{x^t}$ and the corresponding covariance matrix $\boldsymbol{P^t}$ to time $t+1$ the standard prediction rule of the EKF applies:

$$\begin{aligned}
\bar{\boldsymbol{x}}^{\boldsymbol{t+1}} &= f\left(\boldsymbol{x^t}, \delta_t\right) && \text{(Predicted state)} && (5.16)\\
\bar{\boldsymbol{P}}^{\boldsymbol{t+1}} &= \boldsymbol{F^t}\boldsymbol{P^t}\left(\boldsymbol{F^t}\right)^T + \boldsymbol{Q^t}, && \text{(Predicted state covariance)} && (5.17)
\end{aligned}$$

with f being a non-linear prediction function, δ_t being the time difference between the previous timestamp and the predicted one, $\boldsymbol{F^t}$ being the Jacobian of the prediction function evaluated at $\boldsymbol{x^t}$ and $\boldsymbol{Q^t}$ being the prediction noise covariance matrix.

We use a Constant Turn-Rate & (longitudinal) Acceleration (CTRA) motion model for our prediction function. The derivation of the motion model is described in appendix B and leads to following prediction function:

$$f\left(\boldsymbol{x^t}, \delta_t\right) = \begin{pmatrix} \bar{x}^{t+1} \\ \bar{y}^{t+1} \\ \bar{v}^{t+1} \\ \bar{\Psi}^{t+1} \\ \bar{a}^{t+1} \\ \bar{\dot{\Psi}}^{t+1} \end{pmatrix} \tag{5.18}$$

$$\bar{x}^{t+1} = \begin{cases} x^t + \frac{\bar{v}^{t+1}\cdot\left(\sin\left(\bar{\Psi}^{t+1}\right)-\sin\left(\Psi^t\right)\right)+a^t\sin(\Psi^t)\cdot\delta_t}{\dot{\Psi}^t} - \frac{a^t\left(\cos\left(\Psi^t\right)-\cos\left(\bar{\Psi}^{t+1}\right)\right)}{\left(\dot{\Psi}^t\right)^2} & \dot{\Psi}^t \neq 0 \\ x^t + \cos(\Psi^t)\left(v^t\delta_t + \frac{1}{2}a^t\delta_t^2\right) & \dot{\Psi}^t = 0 \end{cases} \tag{5.19}$$

$$\bar{y}^{t+1} = \begin{cases} y^t + \frac{\bar{v}^{t+1}\cdot\left(\cos\left(\Psi^t\right)-\cos\left(\bar{\Psi}^{t+1}\right)\right)-a^t\cos\left(\Psi^t\right)\cdot\delta_t}{\dot{\Psi}^t} - \frac{a^t\left(\sin\left(\Psi^t\right)-\sin\left(\bar{\Psi}^{t+1}\right)\right)}{\left(\dot{\Psi}^t\right)^2} & \dot{\Psi}^t \neq 0 \\ y^t + \sin(\Psi^t)\left(v^t\delta_t + \frac{1}{2}a^t\delta_t^2\right) & \dot{\Psi}^t = 0 \end{cases} \tag{5.20}$$

$$\begin{aligned}
\bar{v}^{t+1} &= v^t + a^t \cdot \delta_t && (5.21)\\
\bar{\Psi}^{t+1} &= \Psi^t + \dot{\Psi}^t \cdot \delta_t && (5.22)\\
\bar{a}^{t+1} &= a^t && (5.23)\\
\bar{\dot{\Psi}}^{t+1} &= \dot{\Psi}^t. && (5.24)
\end{aligned}$$

The Jacobian $\boldsymbol{F^t}$ is defined as:

$$\boldsymbol{F^t} = \begin{pmatrix} \frac{\partial\bar{x}^{t+1}}{\partial x^t} & \cdots & \frac{\partial\bar{x}^{t+1}}{\partial\dot{\Psi}^t} \\ \vdots & \ddots & \vdots \\ \frac{\partial\bar{\dot{\Psi}}^{t+1}}{\partial x^t} & \cdots & \frac{\partial\bar{\dot{\Psi}}^{t+1}}{\partial\dot{\Psi}^t} \end{pmatrix}, \tag{5.25}$$

with the partial derivatives listed in B.3. For the prediction noise covariance matrix $\boldsymbol{Q^t}$ we use a diagonal matrix with predefined variances for the single state components.

6 Evaluation & Validation

In this chapter we present the data we used for evaluating the proposed sensor fusion framework and how we extracted ground-truth data to compare against the output of the implemented algorithm. This chapter consists of three sections. The first section 6.1 discusses the data source we used for evaluation and the extraction for our ground-truth data. The second section 6.2 presents the methods used for evaluation, and lastly, 6.3 contains the results of our proposed method and shows the advantages of single features.

6.1 Evaluation Data

Our evaluation is based on the *Mini subset* of ten scenes provided by *nuScenes: A multimodal dataset for autonomous driving* [40]. The dataset contains raw senor data of three radar and one lidar sensor together with the position of the ego vehicle as well as 3-D object annotations and classified map data. Therefore, this dataset is well suited to test our sensor models and fusion approach.

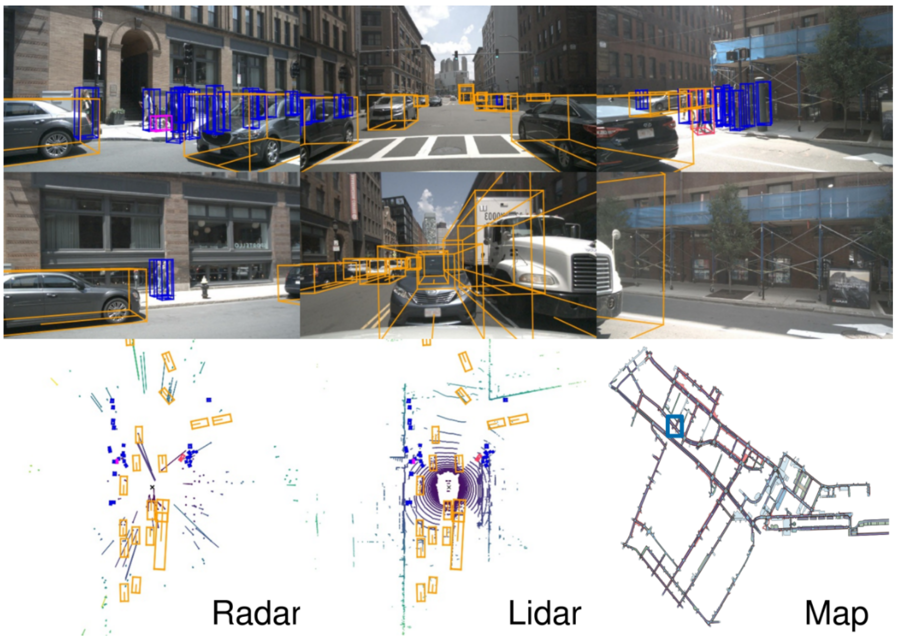


Figure 6.1: An example from the nuScenes dataset [40]. This example shows the annotated 3-D objects projected into the image of the six cameras. The bottom row shows a bird's eye view of the radar and lidar data with the annotated objects as well as an extraction of the corresponding map.

Name	Location	Description
scene-0061	Singapore	Parked truck, construction, intersection, turn left, following a van
scene-0103	Boston	Many peds right, wait for turning car, long bike rack left, cyclist
scene-0553	Boston	Wait at intersection, bicycle, large truck, peds crossing crosswalk, ped with stroller
scene-0655	Boston	Parking lot, parked cars, jaywalker, bendy bus, gardening vehicles
scene-0757	Boston	Arrive at busy intersection, bus, wait at intersection, bicycle, peds
scene-0796	Singapore	Scooter, peds on sidewalk, bus, cars, truck, fake construction worker, bicycle, cross intersection, car overtaking us
scene-0916	Singapore	Parking lot, bicycle rack, parked bicycles, bus, many peds, parked scooters, parked motorcycle
scene-1077	Singapore	Night, big street, bus stop, high speed, construction vehicle
scene-1094	Singapore	Night, after rain, many peds, PMD, ped with bag, jaywalker, truck, scooter
scene-1100	Singapore	Night, peds in sidewalk, peds cross crosswalk, scooter, PMD, difficult lighting

Table 6.1: Overview of the scenes contained in the *mini-dataset*

Each provided scene consists of 20 seconds of driving with annotated frames every 0.5 second. The *mini-dataset* already includes recordings from two different locations (Boston and Singapore) and a variety of different scenarios and/or conditions, as constant driving, busy intersections, day and night-time, etc. Figure 6.1 shows an annotated sample as example. Table 6.1 lists all scenes included in the dataset and used throughout our evaluation.

6.1.1 Reference Grids

For the evaluation of the sensor models (2.3) and our fused grid map (3), we generated so called *reference grids* in order to enable a cell-wise comparison with the results of our proposed algorithms.

Freespace and Occupancy To generate reference grids based on the annotated data from the data-set, we first define areas labeled as *drivable_area* and *walkway* (figure 6.3)

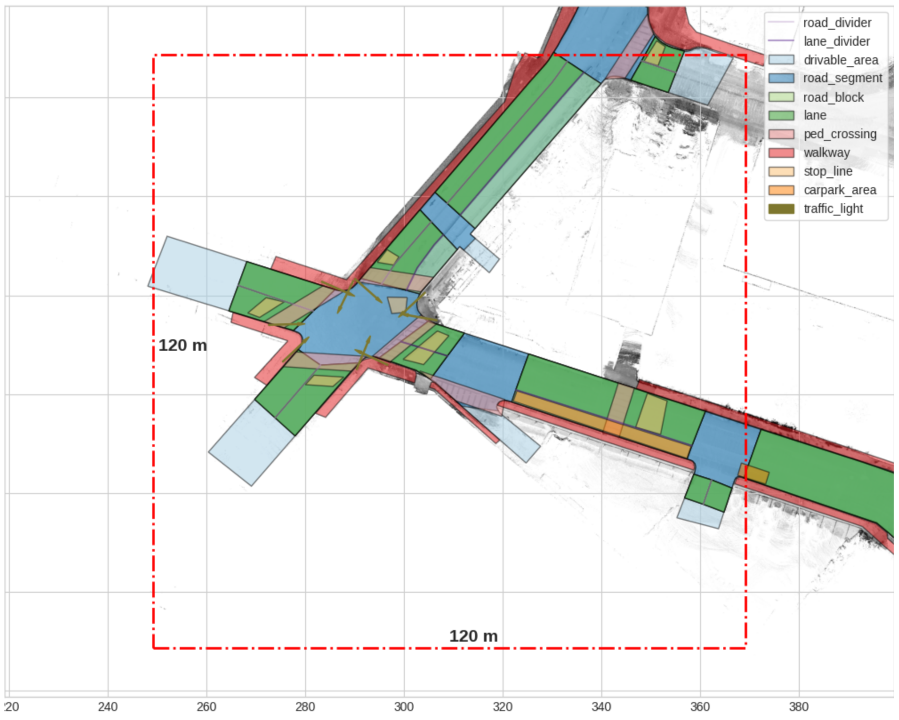
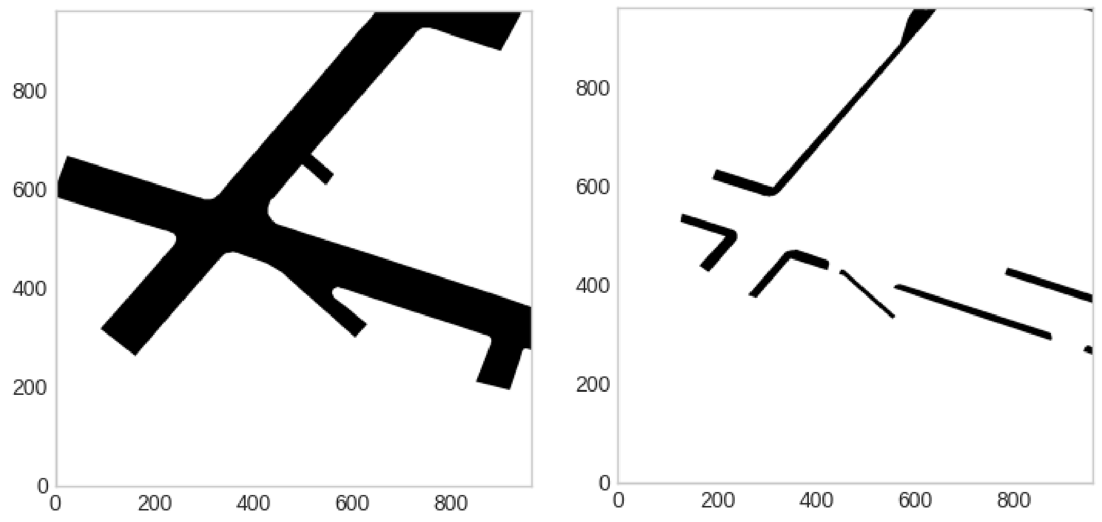


Figure 6.2: Example of the provided map data for a given location.

to be our reference freespace. Note, that we assume that area to be free (if currently not occupied by an object), but do not make any statement about other areas. In particular, we do not state that other areas do not contain freespace. In the second step we extracted all annotated 3-D objects and generate an occupancy grid out of them (figure 6.4a). Subtracting the cells occupied by objects from the static freespace bitmap leads to our final reference bitmap of the current freespace (figure 6.4b).

Velocity Additionally to the reference bitmaps for freespace and occupancy we also generate reference grids for the velocity of single cells. The only locations, where the reference velocity can be calculated, are those cells, which are currently occupied by an object (see fig. 6.4a). From the dataset the location and orientation of object o at time t is given as T_o^t and α_o^t. Given the location of cell c in world coordinates X_c^W, the location can be transformed into the vehicle coordinate system of object o at the current time t, denoted as $X_c^{V,t}$ (see A.5). Computing the reverse transformation (eq. A.5), but using the object's future position T_o^{t+1} and α_o^{t+1}, results in the position, where the content of the cell c is moving to: $X_c^{W,t+1}$. Taking the difference in position over the time difference



(a) Extracted *drivable area.* (b) Extracted *walkway.*

Figure 6.3: Extracted bitmask for *drivable area* and *walkway* based on the map extraction shown in 6.2.

gives us the reference velocity of cell c at the current time t:

$$\vartheta_c^t = \frac{X_c^{W,t+1} - X_c^W}{\Delta_t}. \tag{6.1}$$

Figures 6.4c and 6.4d show the orientation and absolute speed value of the extracted reference velocity information, used in the evaluation. Applying a threshold on the expected cell velocities, static occupied cells can be distinguished from dynamic ones. This leads us to the final reference grid as shown in figure 6.5.

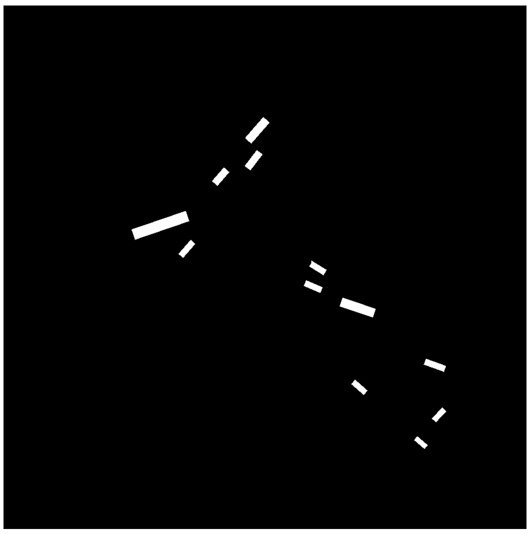
6.1.2 Ground-Truth Object List

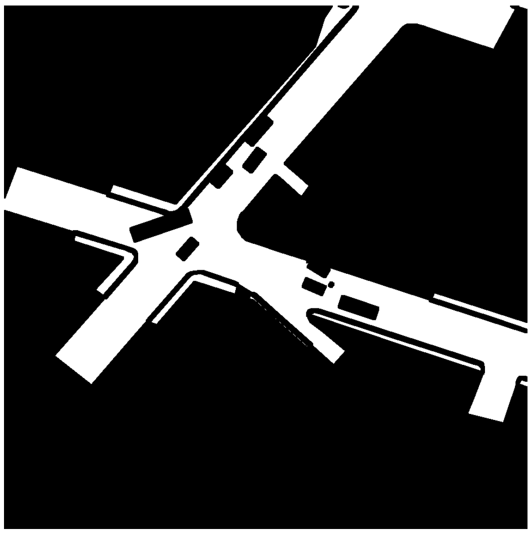
To evaluate the object extraction (5.1) and tracking (5.2) methods we use the so called *annotations* from the nuScenes dataset. Those annotations, available for each annotated frame every 0.5 second, provide bounding boxes defining the position of objects seen in that frame.

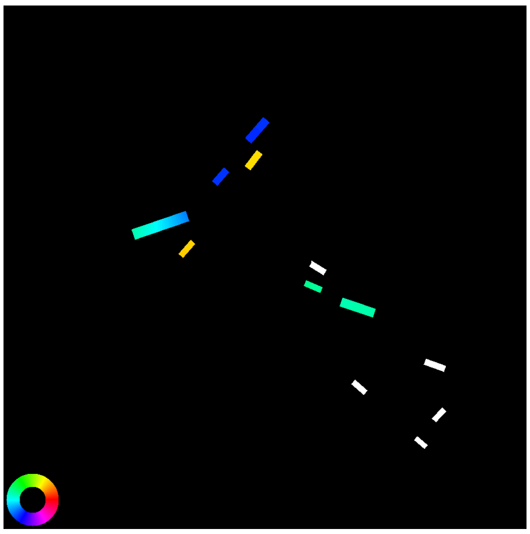
The relevant informations for our evaluation are:

- translation: the position of the bounding box center in global coordinates,
- size: the dimension of the bounding box in length and width and
- rotation: the orientation of the bounding box.

Additionally to the already provided attributes, we estimate the object's velocity vector and rotation rate, based on the current and future/past pose of the object.


(a) Occupied cells based on annotated objects.


(b) Current freespace based on map data and annotated objects.


(c) Orientation of occupied cells. The angles are encoded by the color scheme shown in the lower left corner.

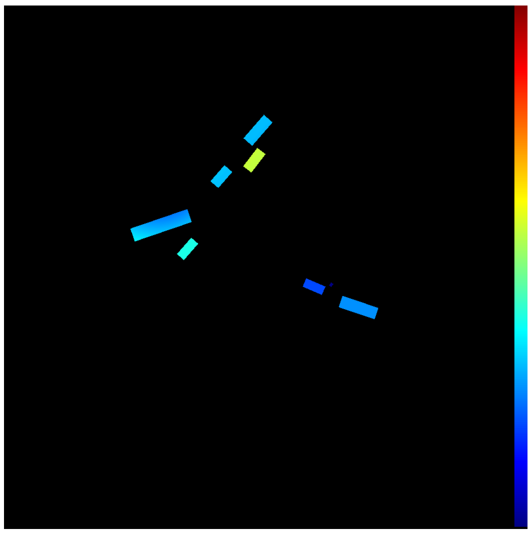

(d) Speed of occupied cells. The color encoding (right bar) covers a speed range from 0 (blue) to 20 m/s (red).

Figure 6.4: Reference grids based on the map data and annotated objects in the nuScenes dataset.

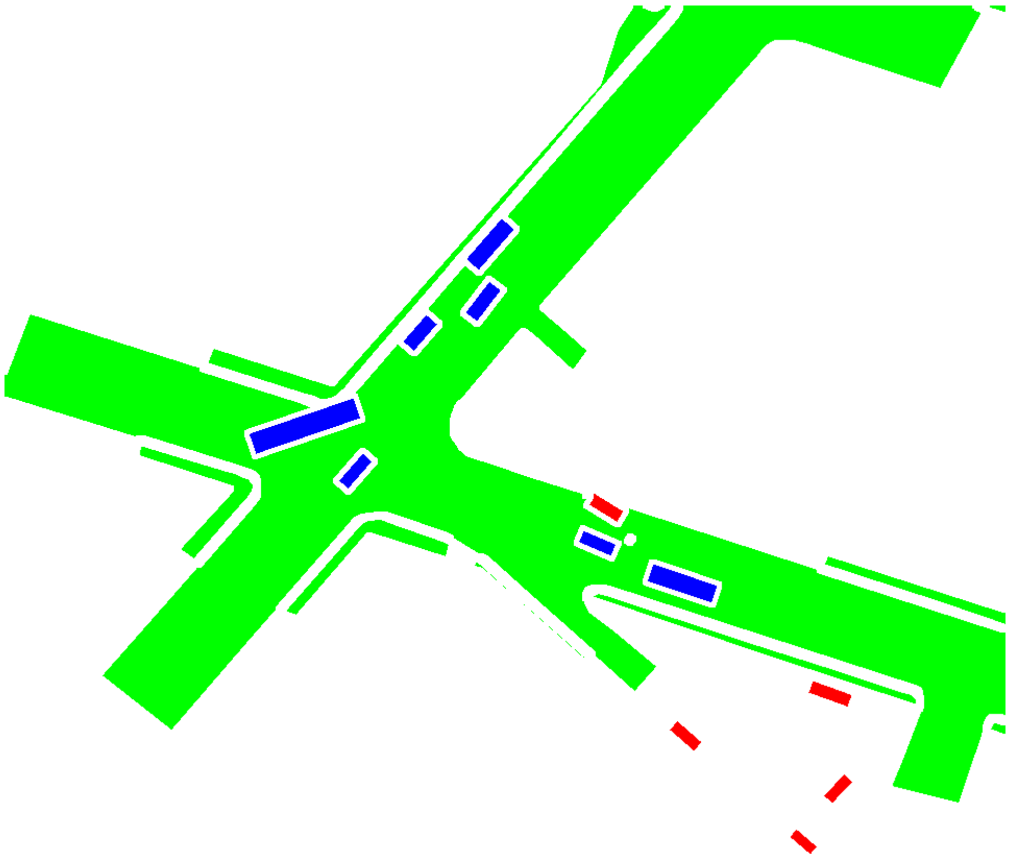


Figure 6.5: Ground-truth map $\mathcal{M}$ with freespace (F) shown in green, static occupancy (S) in red and dynamic occupancy (D) in blue. Unknown area (FSD) is shown in white.

6.2 Evaluation Method

In this section we describe how we evaluate our algorithm using the reference data described in the previous section (6.1). The main focus of the evaluation is to get a metric for comparison of the different processing steps rather then to have an absolute benchmark.

6.2.1 Grid Evaluation

Mass Estimations A rather simple, but powerful, evaluation method for the sensor models (2.3), as well as for the dynamic grid (3), is to compare the ground-truth map (fig. 6.5) with the estimated cell masses.

For every possible ground-truth classification $\vartheta \in \{F, S, D\}$, we calculate the sum of the estimated mass values $\hat{\vartheta} \in \{F, S, D, FD, SD\}$ of all cells c:

$$\Sigma_{\vartheta}\left(\hat{\vartheta}\right) = \sum_{c|\mathcal{M}_c=\vartheta} M\left(\hat{\vartheta}\right), \tag{6.2}$$

with the ground-truth reference map $\mathcal{M}_c$ and the map to be evaluated: M. We call this sum divided by the maximal possible value our *detection score*, which we use for comparison and interpretation of our results:

$$\chi_{\vartheta}\left(\hat{\vartheta}\right) = \frac{\Sigma_{\vartheta}\left(\hat{\vartheta}\right)}{\sum_{c|\mathcal{M}_c=\vartheta} 1}. \tag{6.3}$$

These scores for every possible combination of expected mass (ground-truth) and estimated mass from the algorithms is summed up for all timestamps of each scene and over all scenes. Additionally we calculate the detections scores for different ranges around the vehicle.

Figure 6.6 shows the number of cells (and therefore also the maximal possible detection score) depending on the distance to the vehicle. As expected we see an almost quadratic increase of the number of cells with the distance. The plot also shows that the number of freespace cells is a few magnitudes higher than the number of occupied cells. The number of static occupied cells is also three times higher than the number of dynamic occupied cells.

Velocity Estimations In addition to the described evaluation method for the mass estimations, we also calculate the difference in the estimated velocity v and the reference one V for every cell, where a reference velocity is available:

$$\delta_v = \sqrt{(v - V)^T (v - V)}. \tag{6.4}$$

In order to calculate our velocity scores, we count the number of cells, where the velocity error is below 1, 2 and 4 m/s. With this method we get the percentage of cells with a velocity error of less than δ_v. The advantage of this method compared to others, e.g. calculating the mean error, is that the score cannot get worse, if more cells are getting an estimation.

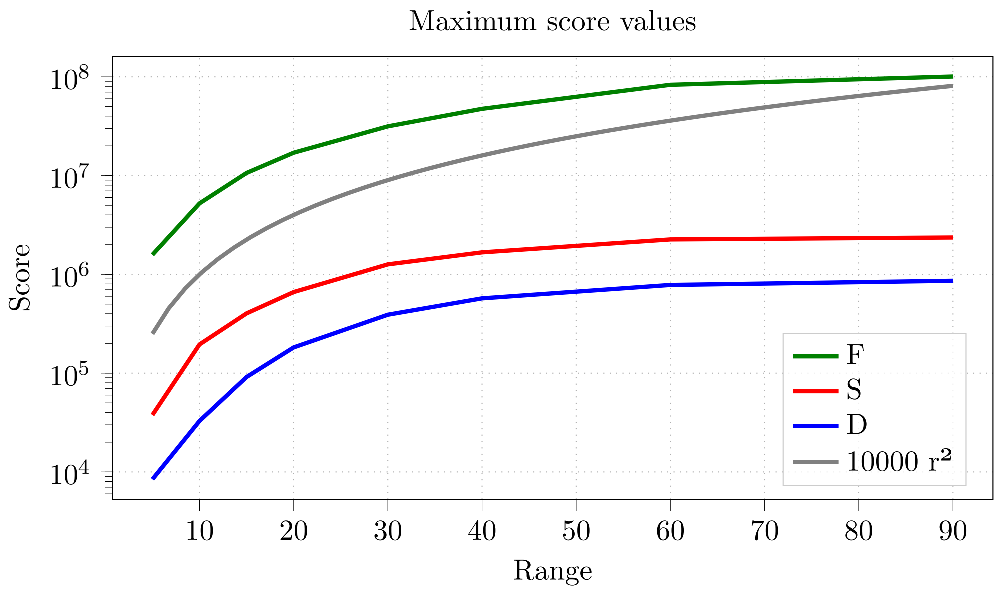


Figure 6.6: Maximum score values over different ranges. All 10 scenes summed up (C.1). The gray line acts only as orientation for a quadratic increase over range.

6.2.2 Object Evaluation

This section describes how the object list from our grid extraction or object tracking algorithm is evaluated with the given ground-truth object list.

Initialization The proposed dynamic grid (chapter 3) and the object tracking (chapter 5.2) are accumulating current and past data, by using predict/update cycles. Using such approaches leads to a certain initial phase of the algorithms. As we do not want to focus on the initialization phase during the evaluation, we skip the first 2 seconds of the scenes in the evaluation data.

Object Association For the association between the estimated objects $\hat{o} \in \hat{O}$ and the ground-truth objects $o \in O$ we use the same intersection over union (IoU) approach as described in 5.2.2, but with a higher margin $\mu = 2\,\mathrm{m}$. Associated objects are counted as true positive and their attributes are compared as described in the next paragraph. Estimated objects with no associated ground-truth object are counted as false-positive, whereas each ground-truth object without any associated estimated object is counted as false-negative.

In case that more than one estimated object is associated with a ground-truth object, only the match with the highest IoU ratio is used for the comparison evaluation. All the other matches are neither considered as true-positive nor as false-positive, but are ignored for the further evaluation. The most common case, where one ground-truth object could be associated with more estimated ones, is if the ground-truth object is partly hidden, e.g. when only its front and rear part can be sensed.

Sensitivity and Precision To compare the number of objects being detected (and associated) with the number of ground-truth objects, we count the following values:

- *true positive (tp)*: The number of correctly detected objects (output object can be associated with a ground-truth object).
- *false negative (fn)*: Number of ground-truth objects, which were not associated with any output object, i.e. undetected objects.
- *false positive (fp)*: An output object, which were not associated with any ground-truth object, i.e. detected objects, which do not exist.

Additionally we calculate the *sensitivity*, *precision* and the *F₁ score* (harmonic mean of sensitivity and precision) as follows:

$$\text{sensitivity} = \frac{\text{tp}}{\text{tp} + \text{fn}} \tag{6.5}$$

$$\text{precision} = \frac{\text{tp}}{\text{tp} + \text{fp}} \tag{6.6}$$

$$\text{F}_1 \text{ score} = 2 \cdot \frac{\text{precision} \cdot \text{sensitivity}}{\text{precision} + \text{sensitivity}} \tag{6.7}$$

Object Filtering Since with our approach no new object tracks are created for static objects, we do not consider the static ground-truth objects in the evaluation. Only if the ground-truth object has moved previously, but is now standing still (e.g. car stopping at an intersection) it is, however, included in the evaluation. We consider the objects with no previous movement as part of the static environment and therefore they are excluded from the freespace estimation (chapter 4), as well as from the dynamic object list.

Comparison To compare the ground-truth object with its matched estimated object, we simply look at the difference of their attributes in terms of:

- position,
- orientation,
- speed,
- length and
- width.

In terms of the position the Euclidean distance between the best matched reference point is chosen (see fig. 5.6). The evaluation results in sections 6.3.3 and 6.3.4 will state the median ($P_{50\%}$) error as well as the 25% ($P_{25\%}$) and 75% percentile ($P_{75\%}$).

6.3 Results

This section shows the qualitative results of our proposed algorithm. We chose a cell-size of 0.125 m $\times$ 0.125 m to generate those results. With the grid dimension set to 960 $\times$ 960 cells, this leads to an area of 120 m $\times$ 120 m around the vehicle. The total number of used particles was configured to 921 600, which means one particle per cell on average.

6.3.1 Sensor Models

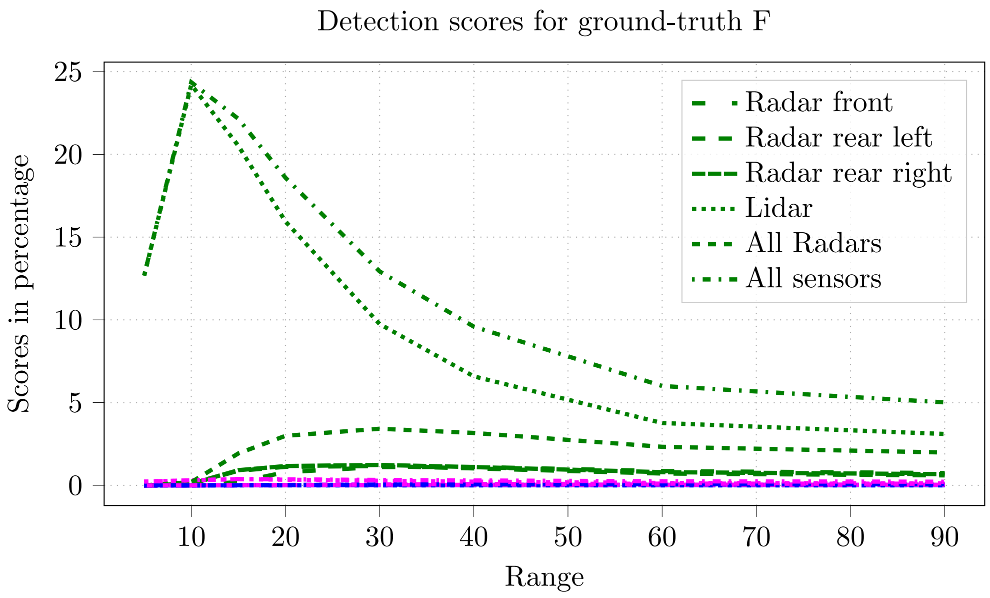


Figure 6.7: Detection scores for freespace of different radar and lidar sensors. All 10 scenes summed up (C.2). The line style shows the different sensor grids as shown in the legend, the color encodes the type of detection score: green: F, blue: D, purple: SD.

Freespace The most obvious observation of the freespace results (fig 6.7) is that the detection score of freespace estimation is declining with the range. This is especially the case for the lidar sensor (dotted line). This result is expected due to the design of our lidar sensor model (2.3.2). With higher distances the number of rays traversing a cell is reduced, either due to occlusion of targets on their way or simply due to the angular measurement principle, leading to higher uncertainty in the freespace estimation.
The increase in the beginning can be explained with the fact, that the lidar sensor has some defined vertical opening angle and therefore the area next to the vehicle cannot be sensed.

The radar sensors (2.3.3, shown as dashed lines here) on the other side, do not estimate any freespace up to their configured minimum range (in this case 10 m around the vehicle). For ranges greater than that the area covered by the sensors increases quadratically, explaining the increase for ranges between 10 m and 30 m, but same as with the lidar sensors, with higher distance the uncertainty in the freespace estimation is increasing, leading to a nearly constant freespace detection score for distances greater than 30 m.

The best freespace estimation is achieved with the combined output of the sensor models (2.3.4). The combination of all three radar sensors (dashed lines) is outperforming each single radar freespace estimation (other forms of dashed lines), which in turn is outperformed by the fusion of all four sensor estimates (dashed/dotted line).

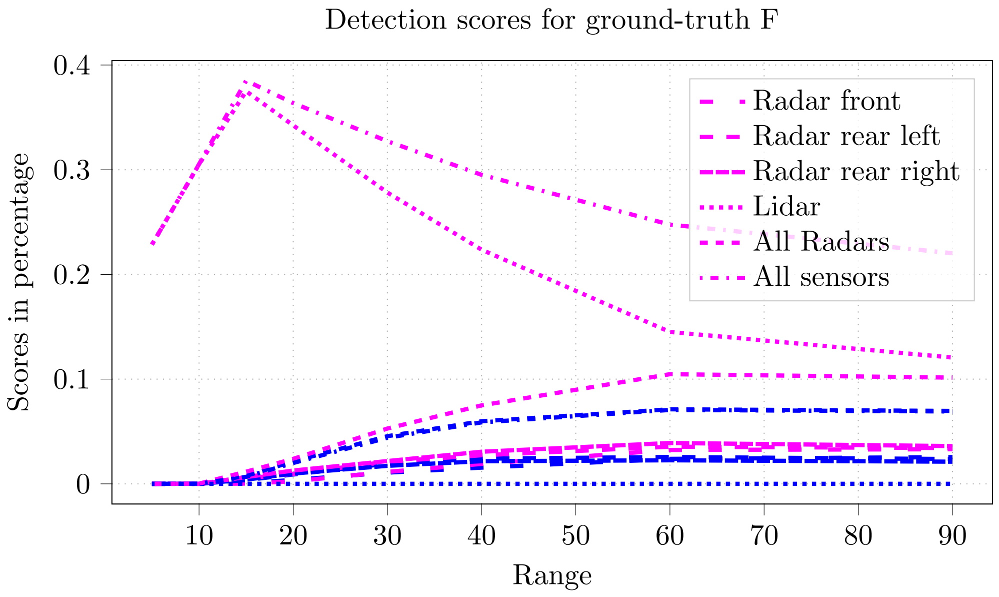
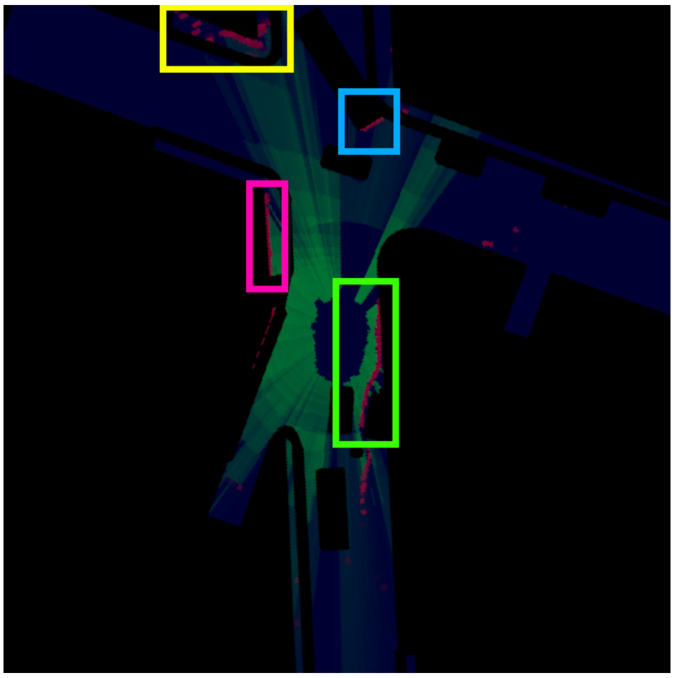


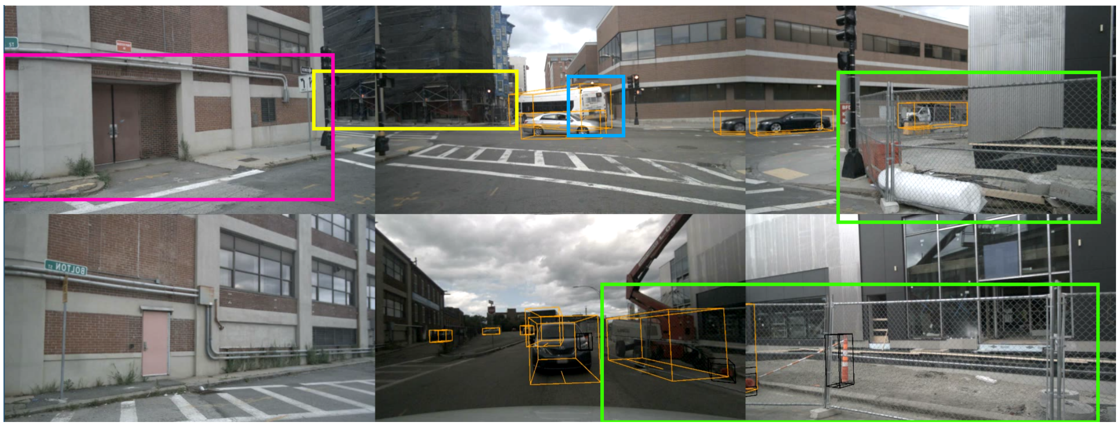
Figure 6.8: Detection scores for freespace of different radar and lidar sensors, without the freespace estimation. For a complete picture see figure 6.7. The line style shows the different sensor grids as shown in the legend, the color encodes the type of detection score: blue: D, purple: SD.

The number of false estimations of the freespace is very low, so it can be barely seen in figure 6.7, therefore figure 6.8 shows the same data, but without the correct estimation of freespace. We basically see the same curves in the graph as with the correct freespace estimation, but at a much smaller level. Again the lidar acts as the dominant sensor, as specified in the sensor fusion (2.3.4). The false classifications are mainly caused by inaccurate sensor measurements (or mismatches between sensor data and reference data), non permanent obstacles (e.g. scaffolding or fences on walkways), houses adjacent to walkways, wrong classified ground-points (e.g. curbstone not classified as ground), or small objects (e.g. cones) not labeled in the ground-truth data. Examples of some wrong classifications can be seen in figure 6.9.

In summary we see good results in the estimation of the freespace area, with a very small amount of wrong classifications.



(a) Output of the fused sensor grid. Blue background shows the reference freespace, green shows the (correct) estimated freespace, whereas red shows the “wrong” estimated occupancy.



(b) Top row: left, middle and right front camera images. Bottom row: left, middle and right rear camera images.

Figure 6.9: Example output of scene *scene-0757* at $t = 6.81\,\text{s}$. Four different sources of wrong occupancy estimation in freespace are marked with colored rectangles. Yellow: scaffolding on the walkway, purple: wall next to the walkway, blue: small mismatch between measured distance and reference data, green: fence on the street/walkway.

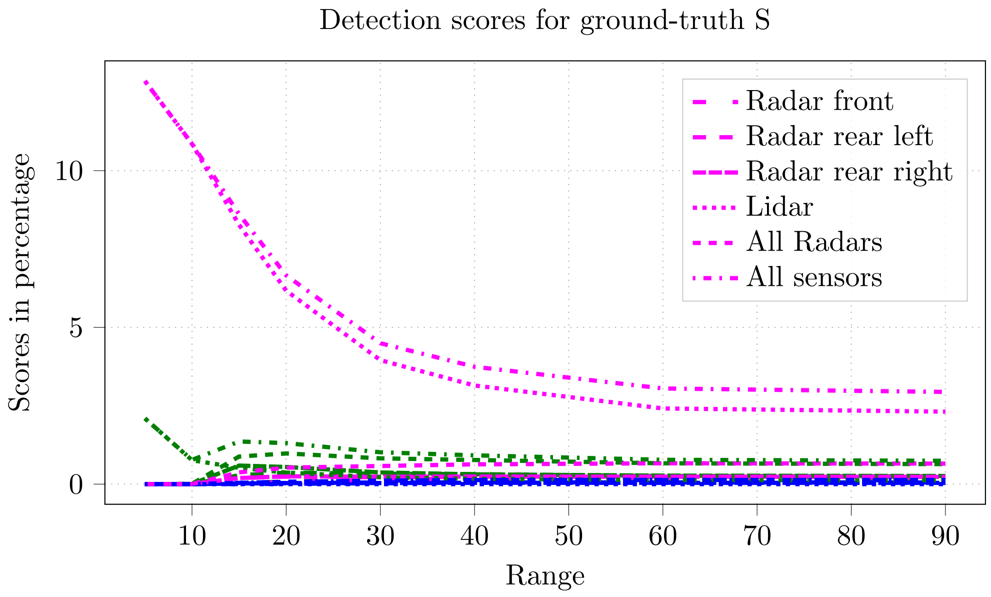


Figure 6.10: Detection scores for static occupancy of different radar and lidar sensors. All 10 scenes summed up (C.3). The line style shows the different sensor grids as shown in the legend, the color encodes the type of detection score: green: F, blue: D, purple: SD.

Static Occupancy Since none of our proposed sensor models (2.3) is estimating static occupancy S, we can only evaluate the estimation of static or dynamic occupancy SD and dynamic occupancy D against freespace F. Figure 6.10 depicts the detection scores in case of static occupied cells. What we can observe here again is the high accuracy of the lidar sensor (dotted lines) compared to the radar sensors, while also delivering a very low score for a wrong estimation of freespace. Similar to the freespace case, the limited range of the lidar sensor can be seen here as well. The main reason for the poor estimation performance compared to the freespace is, that the sensor can only see one or two edges of a static object, whereas the reference map classifies the whole area of the object as being static occupied (figure 6.12).

Figure 6.11 shows the detection scores only for the radar sensors and the fusion of those. Merely the front-looking radar sensor has a higher detection score for occupancy ($SD + D$) than for freespace. Because the field of views of the radar sensors are mainly not overlapping, the combined radar output differs only in a larger covered area, hence performing very similar to the single sensors. Therefore the radar sensor is not performing well in case of the static occupied reference cells.

Using the combined sensor grid (lidar and radar sensors) results in a better detection score of SD compared to the single technologies over all ranges, but has also the drawback of a higher freespace F detections score than using the lidar sensor only.

In summary the combined sensor grid again provides the best results and delivers a decent detection score for the static occupied reference cells.

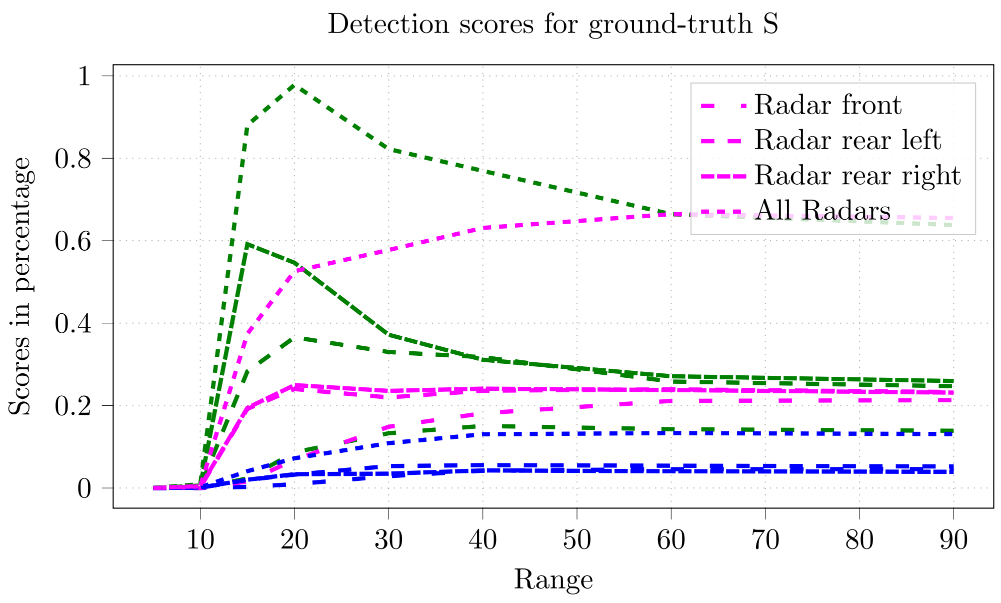
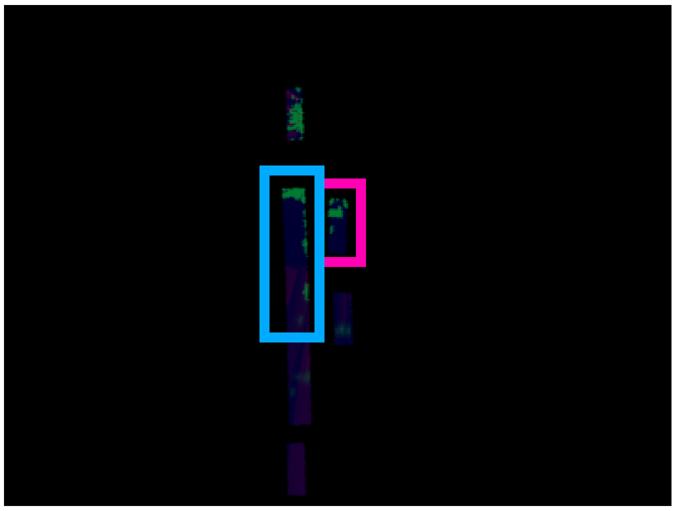
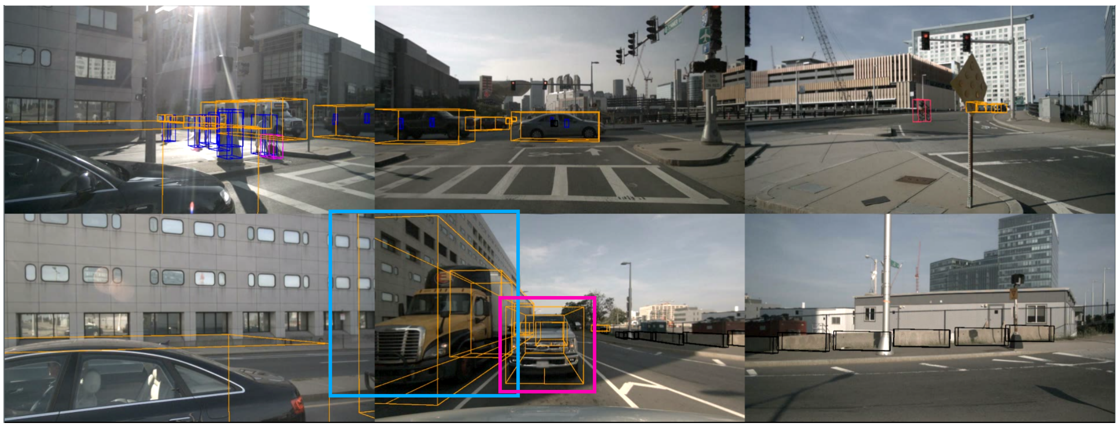


Figure 6.11: Detection scores for static occupancy of different radar sensors. All 10 scenes summed up. For a complete picture see figure 6.10. The line style shows the different sensor grids as shown in the legend, the color encodes the type of detection score: green: F, blue: D, purple: SD.



(a) Output of the fused sensor grid. Blue background shows the reference static occupancy, green shows the correct estimated static occupancy, whereas red shows the wrong estimated freespace.



(b) Top row: left, middle and right front camera images. Bottom row: left, middle and right rear camera images.

Figure 6.12: Example output of scene *scene-0553* at $t = 3.00\,\text{s}$. Two examples of low static occupancy estimation are marked with colored rectangles. Blue: truck behind the ego vehicle, purple: car directly behind the ego vehicle.

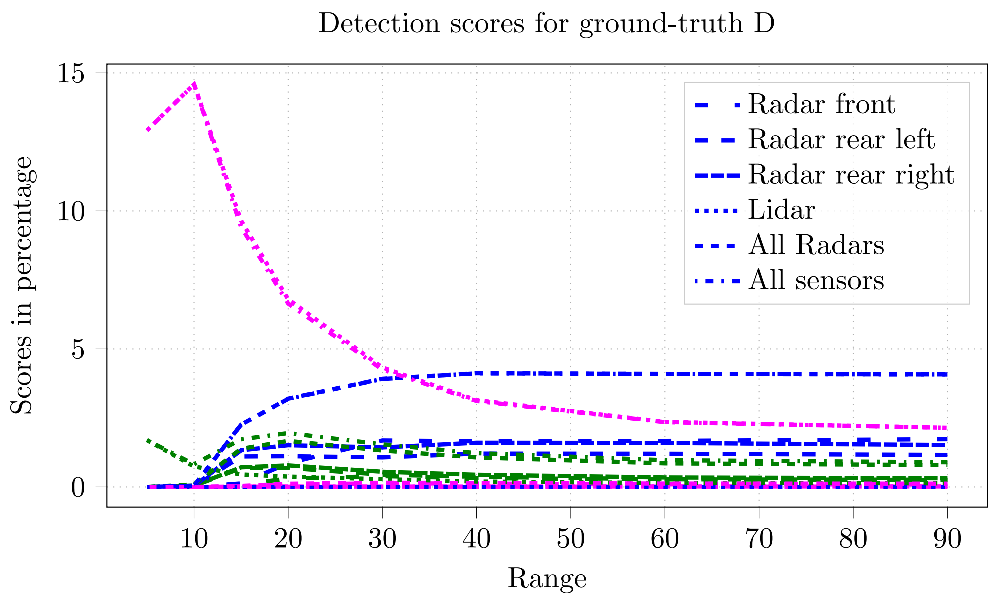
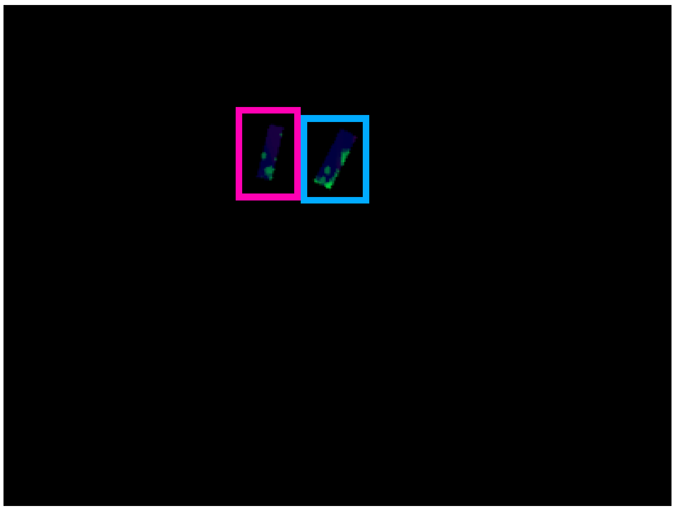


Figure 6.13: Detection scores for dynamic occupancy of different radar and lidar sensors. All 10 scenes summed up (C.4). The line style shows the different sensor grids as shown in the legend, the color encodes the type of detection score: green: F, blue: D, purple: SD.

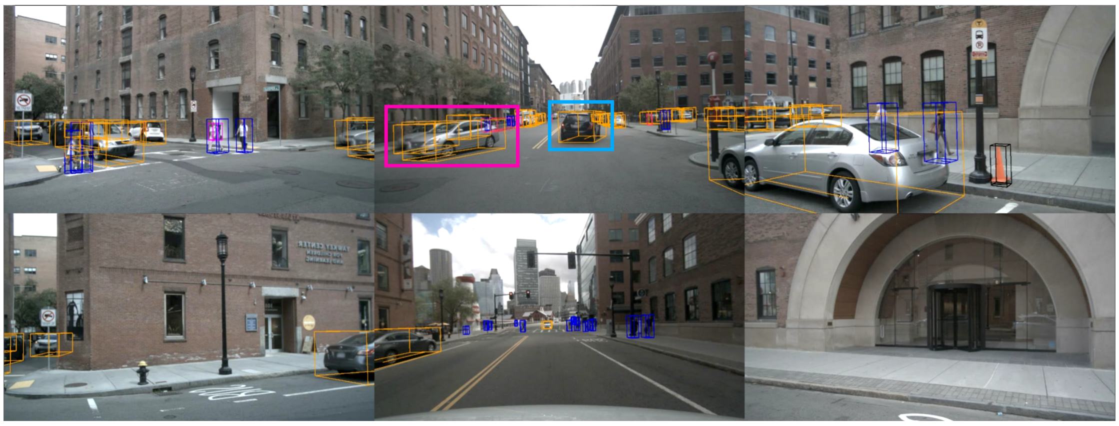
Dynamic Occupancy In the near range ($< 30\,\text{m}$) where the lidar sensor has the higher impact, the detection scores for static or dynamic occupied SD are the dominant ones. As the lidar sensor cannot distinguish between static or dynamic, this result is the expected one. The radar sensors on the other hand can classify most dynamic objects correctly and therefore the detection score for those is the dominant contribution of the radar sensors. Both estimates are dominating the wrong freespace estimation.

Again, the reason for the general low detection scores, compared to those of the reference freespace cells, is the visibility of merely the edges of the dynamic objects (figure 6.14).

With the combination of the lidar and radar sensor grids, the good occupancy estimation of the lidar in the close-range is combined with the dynamic classification of the radar sensors. This leads to a superior result compared to the single sensor technologies.



(a) Output of the fused sensor grid. Blue background shows the reference dynamic occupancy, green shows the correct estimated occupancy ($D+SD$), whereas red shows the wrong estimated freespace.



(b) Top row: left, middle and right front camera images. Bottom row: left, middle and right rear camera images.

Figure 6.14: Example output of scene *scene-0103* at $t = 9.31\,\mathrm{s}$. Two examples of low dynamic occupancy estimation are marked with colored rectangles. Both showing that only small parts of the objects are sensed by the sensor models.

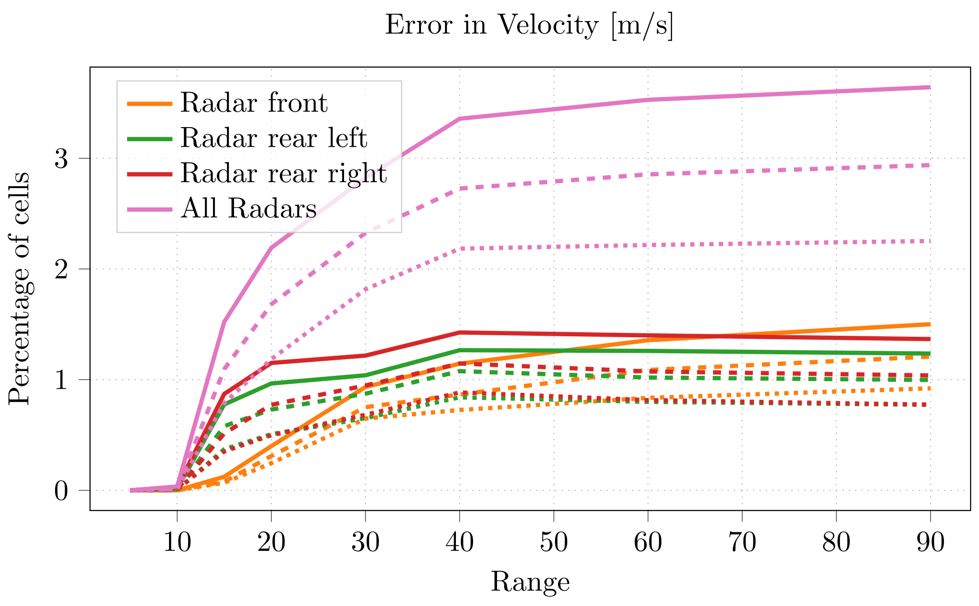


Figure 6.15: Detection scores for dynamic occupancy of different radar and lidar sensors. All 10 scenes summed up. Dotted lines show the cell percentage with a maximal velocity error of 1 m/s, dashed lines with 2 m/s and solid lines with 4 m/s. (C.5)

Velocity Estimation Figure 6.15 shows the velocity estimation score of the single radar sensors, as well as for the combined grid. As the lidar sensor is not providing any velocity information at all, evaluating the lidar or the combined sensor grid in terms of velocity estimation is meaningless.

The low percentage of “correctly” estimated cells has various reasons. One reason is that at most two edges of an object can be sensed, as already mentioned. Additionally not the entire surroundings of the vehicle are within the field of view of the radar sensors. Another reason is that the radar sensors are only measuring the radial velocity component, and hence the tangential component is completely missing.

The combined sensor grid of the three radar sensor receives better velocity scores, but almost exclusively due to the combined sensor field of view.

6.3.2 Dynamic Grid

6.3.2.1 Comparison Against Sensor Grids

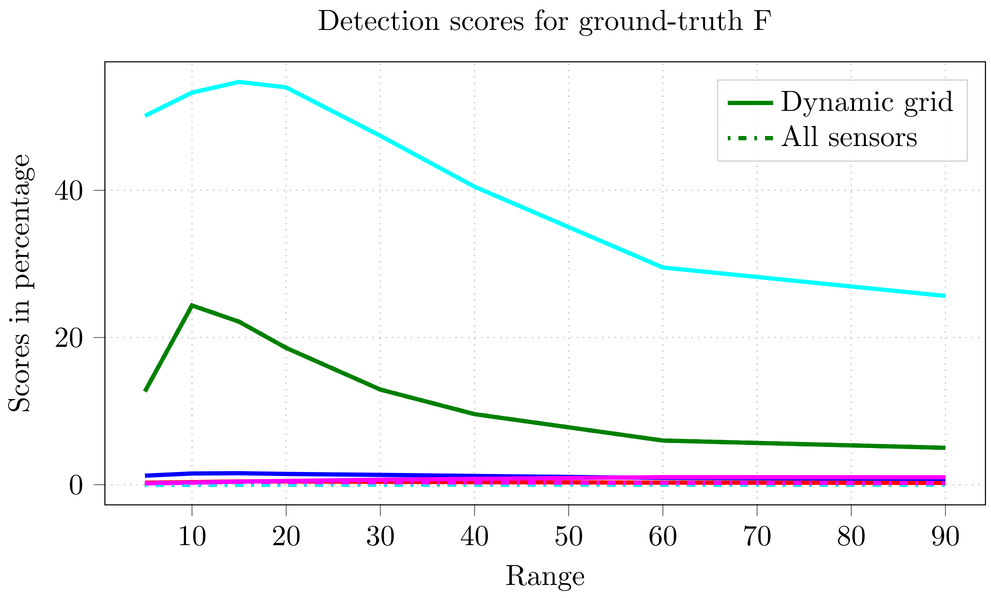


Figure 6.16: Detection scores for freespace of the input sensor grid and the dynamic grid. All 10 scenes summed up (C.6). The line style shows the different evaluation input as shown in the legend, the color encodes the type of detection score: green: F, red: S, blue: D, cyan: FD, purple: SD.

Freespace Figure 6.16 shows the detection score for freespace areas of the (combined) sensor grid, as well as for the dynamic occupancy grid (3). The estimation of being freespace F of the dynamic grid is almost equal to the sensor grid, as expected by the given combination rule (3.2.1.1). The highest detection score is given by the estimation of freespace or dynamic occupancy FD, which results in the accumulation of freespace measurements. With increasing distance to the vehicle, the uncertainty of freespace estimation decreases, leading to lower detection scores.

Estimation of occupancy in case of our reference freespace occurs only with a very low detection score and is caused by wrong sensor measurements or, in case of D, by wrong prediction of dynamic occupancy.

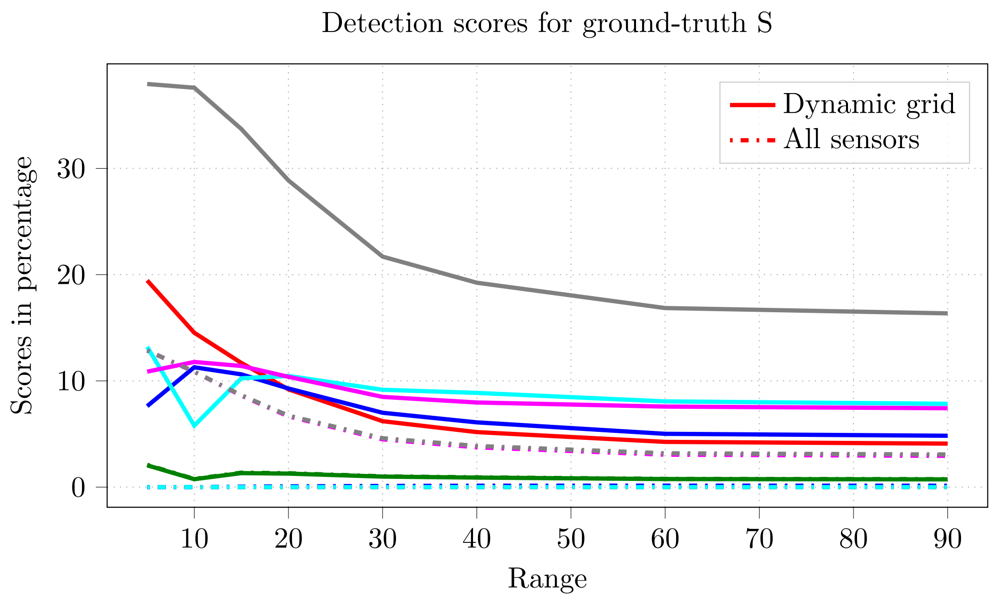


Figure 6.17: Detection scores for static occupancy of the input sensor grid and the dynamic grid. All 10 scenes summed up. (C.7). The line style shows the different evaluation input as shown in the legend, the color encodes the type of detection score: green: F, red: S, blue: D, cyan: FD, purple: SD, gray: sum of S, D and SD.

Static Occupancy The detection scores in case of static occupancy are shown in figure 6.17. The estimations for being free or dynamic occupied FD and static or dynamic occupied SD are getting the highest detection scores, with a small exception of S for smaller ranges. The accumulated grid suffers from the same visible edge issue as the sensor grids, but to a lower extent. Due to driving past static objects more than two edges can be sensed.

Combining the different estimations of being occupied $S + D + SD$ (shown in gray) would get the highest detection scores. This means that static occupied cells are estimated as occupied in most cases. We want to note that static structures (e.g. buildings) are not part of our reference data, however we do see good results for the static occupancy estimation as well.

The estimation of being static occupied S of the dynamic grid is getting better detections scores than the occupancy estimation of the sensor grid. This result is also expected since the accumulation of static environments is the most common use-case of occupancy grid maps.

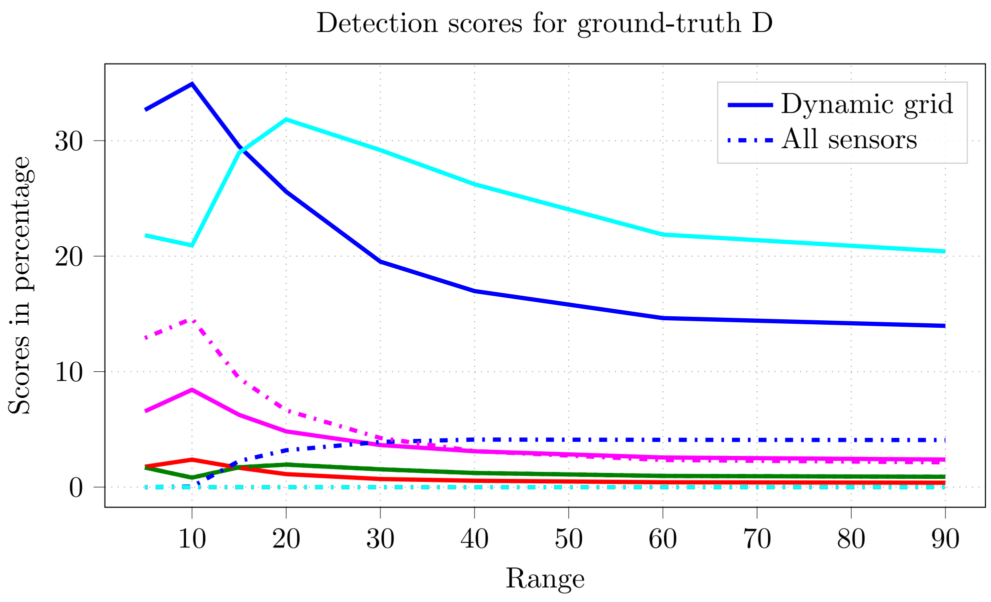


Figure 6.18: Detection scores for dynamic occupancy of the input sensor grid and the dynamic grid. All 10 scenes summed up (C.8). The line style shows the different evaluation input as shown in the legend, the color encodes the type of detection score: green: F, red: S, blue: D, cyan: FD, purple: SD.

Dynamic Occupancy For dynamic occupied cells, the detection scores of D of the dynamic grid outperforms the complete occupancy estimation of the sensor grid (fig. 6.18). The high detection scores of being free or dynamic occupied FD are again caused by the visibility of only one or two edges of an object. If an objects is moving into an area that was sensed as free before, only the edges can be sensed as occupied, whereas a large amount of the occupying object will be located in an area which is estimated as FD. The estimation of being freespace F almost equals the estimation of the sensor grid and therefore does not require any additional explanation here.

The estimation of being dynamic occupied D of the dynamic occupancy grid works well. Since the dynamic estimation is mainly based on the prediction step, which also includes a velocity estimation, this conclusion leads us directly to the next paragraph: the evaluation of the estimated velocity.

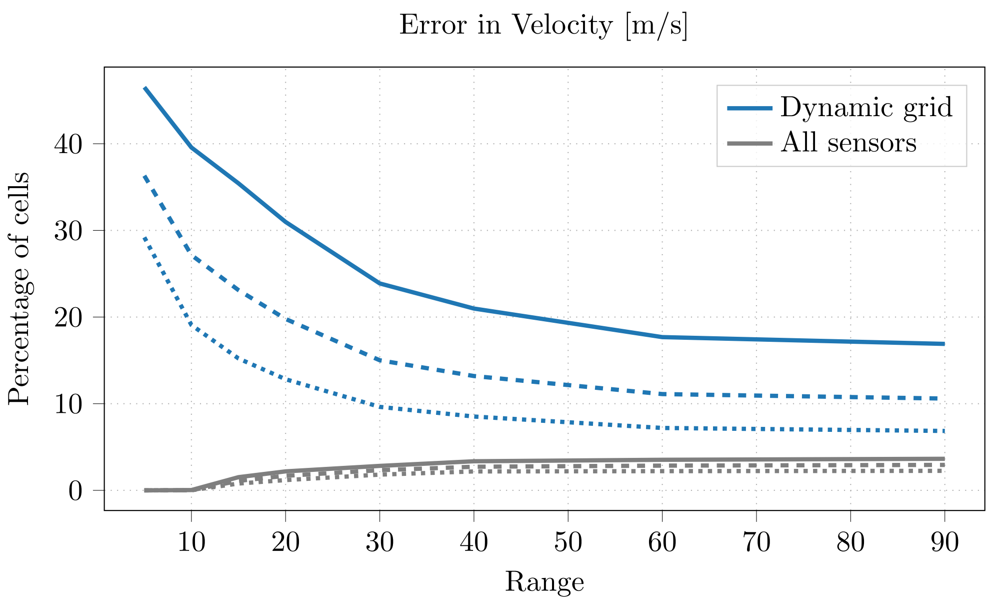


Figure 6.19: Percentage of cells with an estimated velocity within a given error compared to the reference value. Only cells with a reference velocity are considered. Dotted lines show the cell percentage with a maximal velocity error of 1 m/s, dashed lines with 2 m/s and solid lines with 4 m/s. (C.9)

Velocity Estimation Figure 6.19 compares the estimated velocity of the cells with the measured velocity of the sensor grid. Since the measurement of velocity is very limited, as discussed in the previous section, the estimated velocity of the proposed dynamic grid clearly outperforms the sensor grid in every range. This observation shows that the velocity estimation works regardless of the provided velocities of the sensor grids.

Summary The comparison of the output of the dynamic occupancy grid with the sensor grid shows the advantage of the dynamic occupancy over using the sensor grids alone. In terms of the ground-truth freespace, the detection scores for freespace are superior by a factor of three, whereas the wrong classification of freespace stays very low. Additionally in terms of ground-truth static occupancy we see a similar outperformance compared to the frame by frame input of the sensor grid. The major reason for both enhancements of the classification of freespace and static occupancy can be explained by the accumulation of non dynamic parts of the environment.

For the dynamic occupancy and velocity estimation the prediction step, using the particles to transport information, is the key factor. The proposed predict and update cycle on grid level leads to an estimation output a single frame sensor cannot reach. As lidar sensors cannot measure velocity at all and the radar sensors can only measure one of two velocity components, the velocity estimation in the dynamic grid is a key advantage.

6.3.2.2 Impact of the Orientation Prior

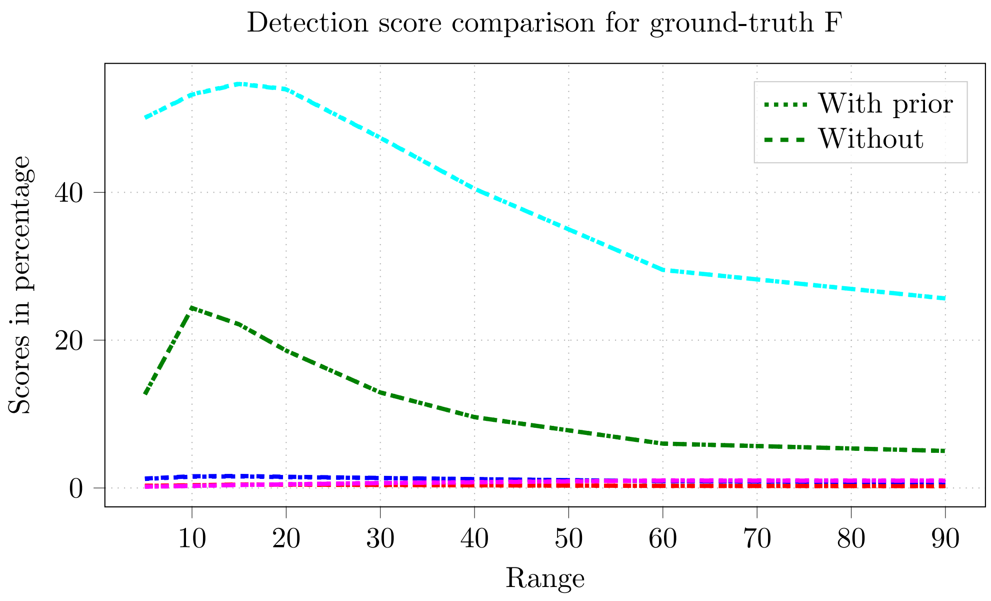


Figure 6.20: Detection scores in percentage for freespace of dynamic grid, with and without using the online orientation prior. All 10 scenes summed up (C.10). The line style indicates whether the online generated orientation prior is used or not. The color encodes the type of detection score: green: F, red: S, blue: D, cyan: FD, purple: SD.

Freespace In terms of the detection score of freespace, we do not see any significant difference in the dynamic grid output between using the online generated orientation prior information and not using it.

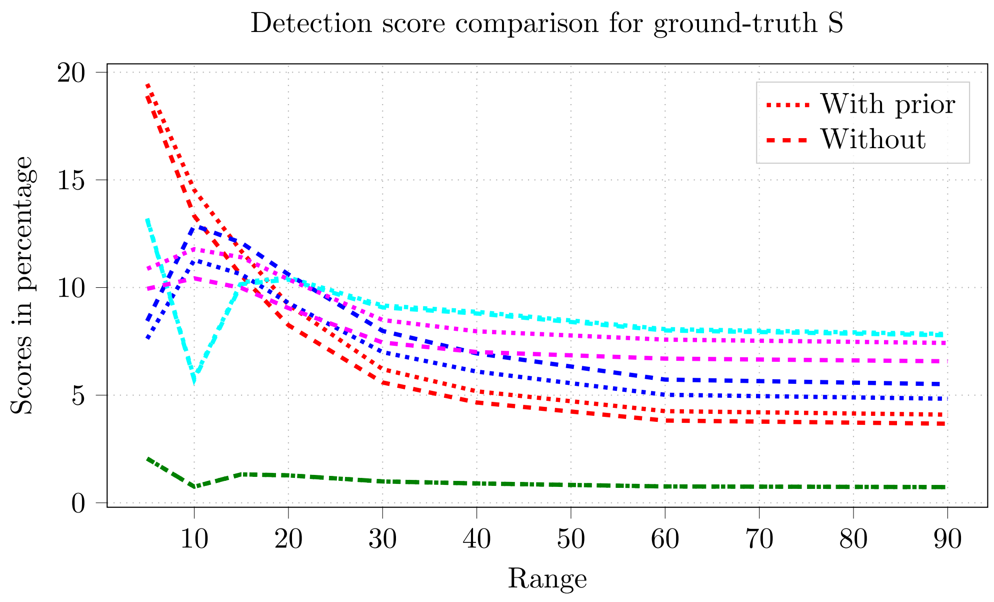


Figure 6.21: Detection scores in percentage for static occupancy of dynamic grid, with and without using the online orientation prior. All 10 scenes summed up (C.11). The line style indicates whether the online generated orientation prior is used or not. The color encodes the type of detection score: green: F, red: S, blue: D, cyan: FD, purple: SD.

Static Occupancy In the case of static occupied cells, we observe a small increase in the static occupied S score, if the online orientation prior is used. The detection scores for (static or dynamic) occupied also improve if the online orientation prior is used. On the other side the detection scores for wrong classification of being dynamic occupied D is lower, than without having the prior information. This leads to the conclusion, that more cells are correctly estimated as static S or (static or dynamic SD) occupied by using the proposed online orientation prior.

For cells being incorrectly classified as F or FD, the orientation prior has no significant impact.

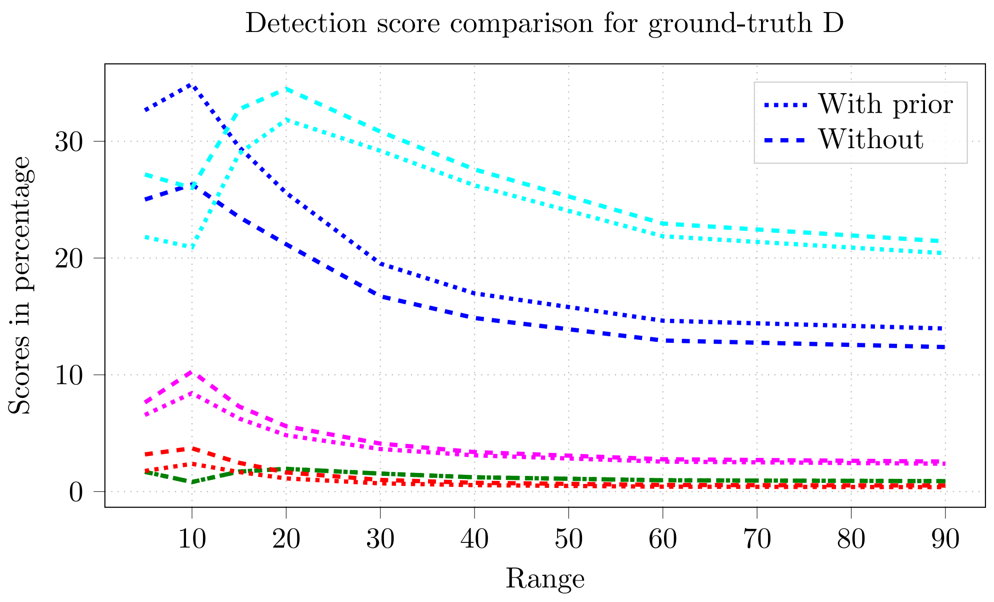


Figure 6.22: Detection scores in percentage for dynamic occupancy of dynamic grid, with and without using the online orientation prior. All 10 scenes summed up (C.12). The line style indicates whether the online generated orientation prior is used or not. The color encodes the type of detection score: green: F, red: S, blue: D, cyan: FD, purple: SD.

Dynamic Occupancy The evaluation of the detection scores for dynamic occupied cells shows that the detection score for D is significantly higher with the orientation prior information than without it. At the same time the estimation as being FD is decreased noticeably. The detection scores for S and SD are slightly lower than without using the prior information.

The wrong estimation of being freespace F is not affected by the online orientation prior.

For the dynamic occupied cells, we can reach the same conclusion as in the static occupied case: the online orientation prior leads to better classification of occupied cells.

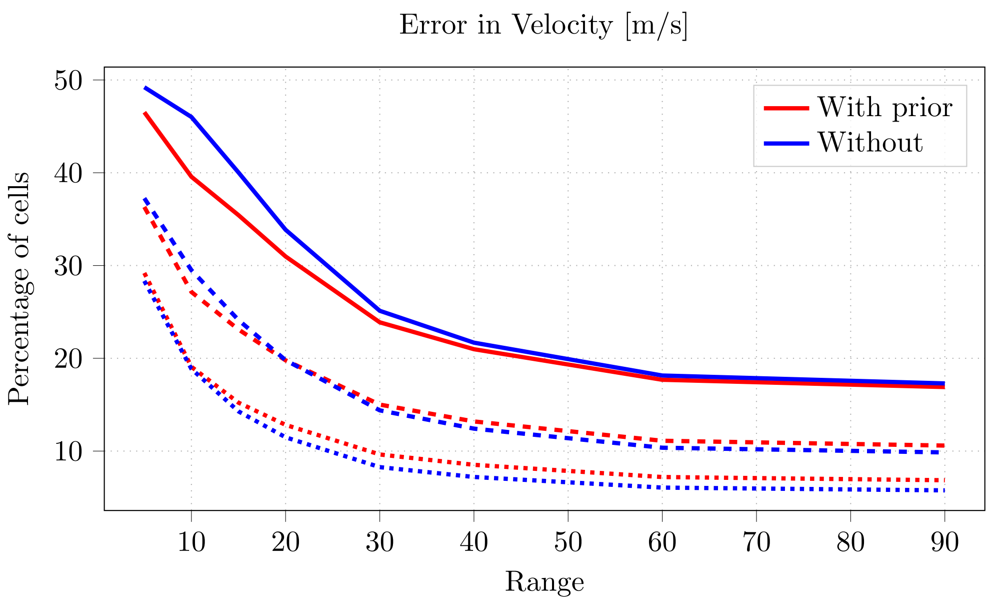


Figure 6.23: Percentage of cells with an estimated velocity within a given error compared to the reference value. Only cells with a reference velocity are considered. Dotted lines show the cell percentage with a maximal velocity error of 1 m/s, dashed lines with 2 m/s and solid lines with 4 m/s. (C.13)

Velocity Estimation The comparison of the percentage of cells with a given error in velocity gives us a slightly more differentiated picture. Although the percentage of cells with a speed error below 1 m/s is slightly higher with the orientation prior, the percentage of cells within a higher velocity error is smaller than without using the prior information.

6.3.3 Object Extraction

In this section we will evaluate the proposed object extraction method (5.1). Additionally we show the impact of the online generated orientation prior (3.2.3) on the extracted objects.

Comparison Value For the comparison of the position, orientation, speed and dimension of the objects we use the 25%-percentile $P_{25\%}$, the median value $P_{50\%}$, the 75%-percentile $P_{75\%}$ and the mean value of the error. Additionally we show the "comparison" percentile value for the extracted objects using the orientation prior. This value is the q-percentile error P_q of the extracted objects using the prior. The value q is defined by the number of true positives of the extracted objects without the orientation prior: n_- and the ones with the prior information: n_+:

$$q = 50\% \cdot \frac{n_-}{n_+}. \tag{6.8}$$

Using this definition of the q-percentile error of the extracted objects with the prior information allows a comparison with the median error of the objects generated without the orientation prior. The median error of the objects, without using the prior, defines the maximal error of the $\frac{n_-}{2}$ best extracted objects. The q-percentile error of the extracted objects, with using the prior information, also gives us the maximal error of the $\frac{n_-}{2}$ best objects, whereas the median error of the "prior" objects is giving us the maximal error of the $\frac{n_+}{2}$ best objects. In the general case we expect $n_+ > n_-$, as the orientation prior should lead to more true positives.

Confusion Matrix Figure 6.24 shows the number of true positives, false positives and false negatives for different ranges. The additional input of the online generated prior leads to more true positives (and hence less false negatives) at the cost of significantly increased number of false positives. This effect is also clearly visible in the sensitivity and precision scores (fig 6.25), where the method with the orientation prior is leading to smaller precision, but higher sensitivity. With the F_1 score being better through all ranges we conclude a positive effect of the online generated orientation prior for the object extraction.

In close areas ($< 10m$) and far areas ($> 40m$) the F_1 score of the extracted objects with the orientation prior is at most 50%. These results are showing room for further improvements. One approach to decrease the number of false positives and false negatives and simultaneously increase the number of true positives is to use this data as input for an object tracking algorithm (see section 5.2). The results of the object tracking are listed in 6.3.4.

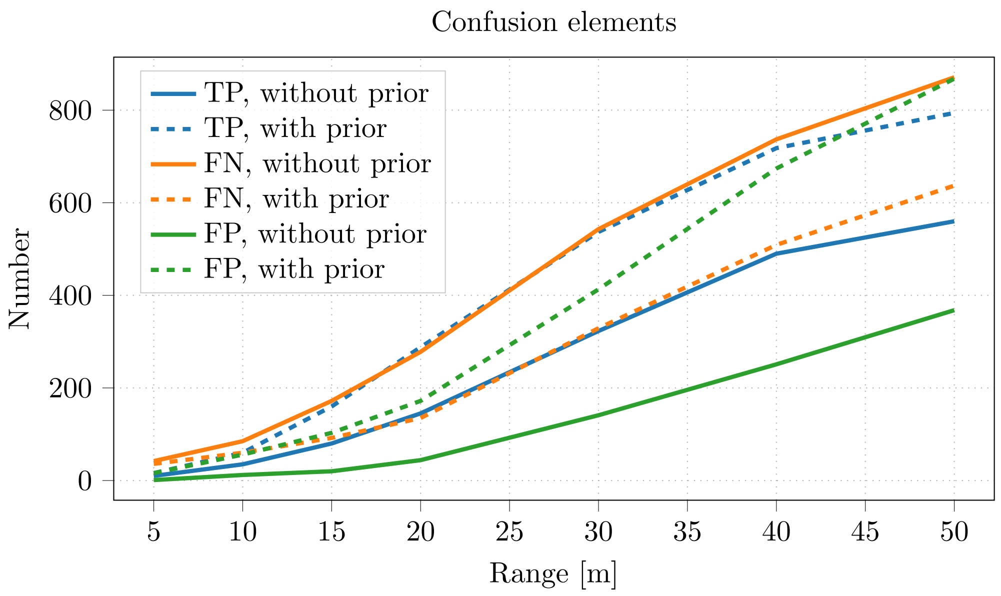


Figure 6.24: True positives, false negatives and false positives over different ranges. All 10 scenes summed up (C.14).

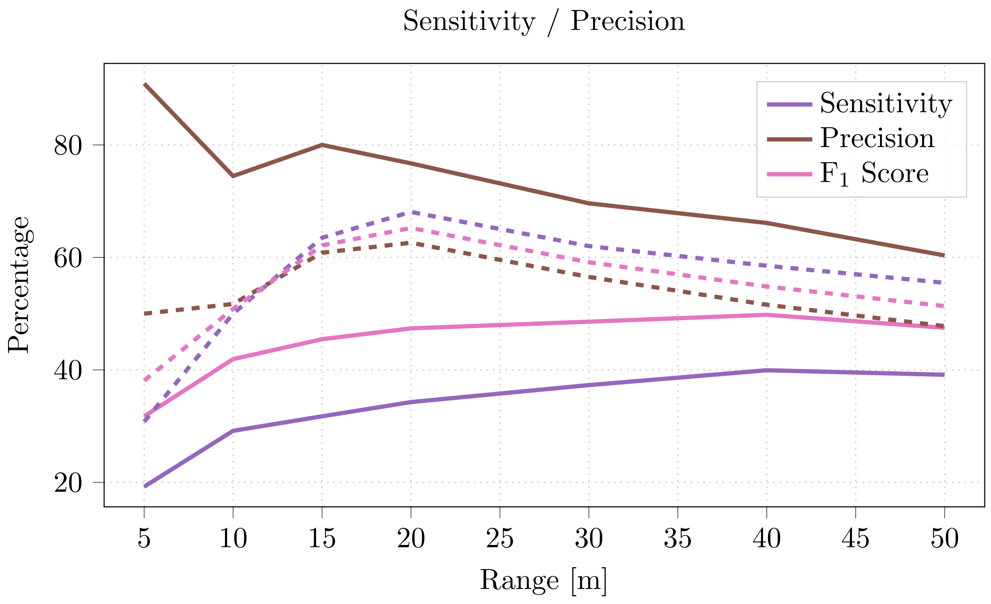


Figure 6.25: Sensitivity, precision and F1 score over different ranges. Solid lines show the result without using the orientation prior, dashed lines show the result using the orientation prior. All 10 scenes summed up (C.14).

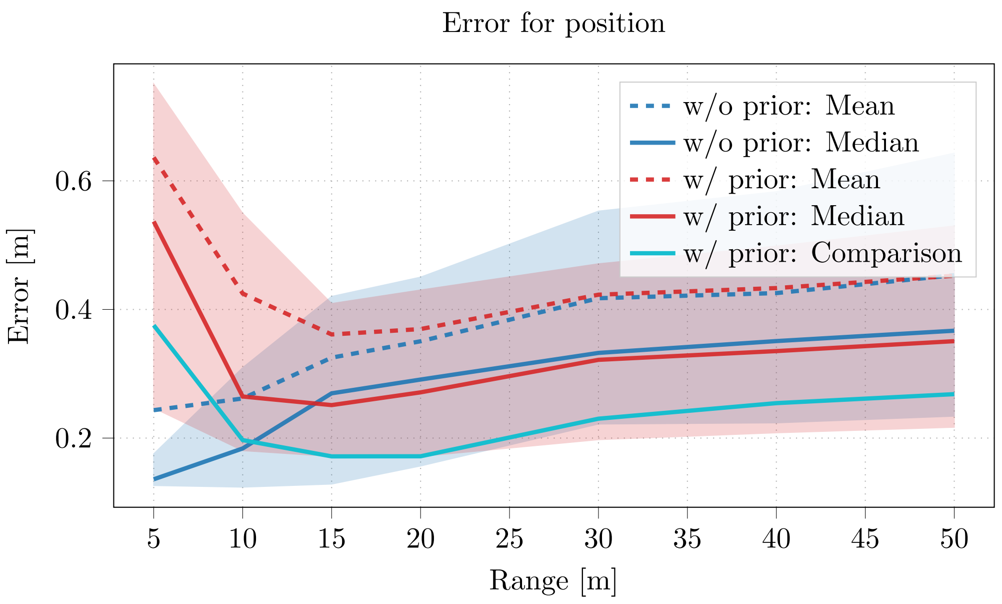


Figure 6.26: Position error of extracted objects over different ranges. All 10 scenes summed up. (C.15)

Position With a mean error of roughly 40 cm and a median error around 35 cm (fig. 6.26) the position accuracy of the extracted object is very good over all ranges, regardless whether the orientation prior is used or not. The chosen association method using intersection over union (5.2.2) relies on a good position. Taking into account the various sources of errors, such as ego position, sensor data, cell prediction and cell discretization, a mean position error of 40 cm shows that the presented approach is working. Overall, the object extraction with using the orientation prior provides slightly better results in terms of position estimation than the one without using this information. The small increase in the mean error is caused by the higher number of associated true positives, as indicated by the P_q value.

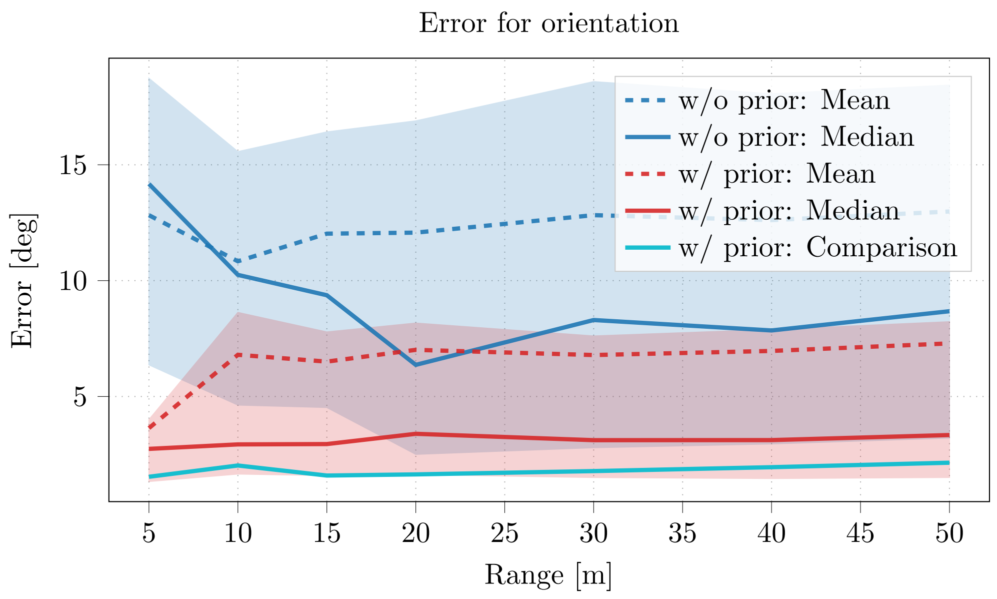


Figure 6.27: Orientation error of extracted objects over different ranges. All 10 scenes summed up. (C.16)

Orientation The orientation error of the object extraction using the orientation prior information is significantly better than without using this information (fig. 6.27). The median error of 3° of the extractions using the prior information shows a very good orientation measurement on at least half of the measurements. The corresponding mean error of 7° indicates outliers with a bad orientation estimation. Nevertheless, with 75% of the object measurements having an orientation error of less than 8°, the quality of the object extraction should be sufficient for most use-cases, at least in terms of orientation. Using the online generated orientation prior dramatically increases the orientation estimation quality of the extracted objects.

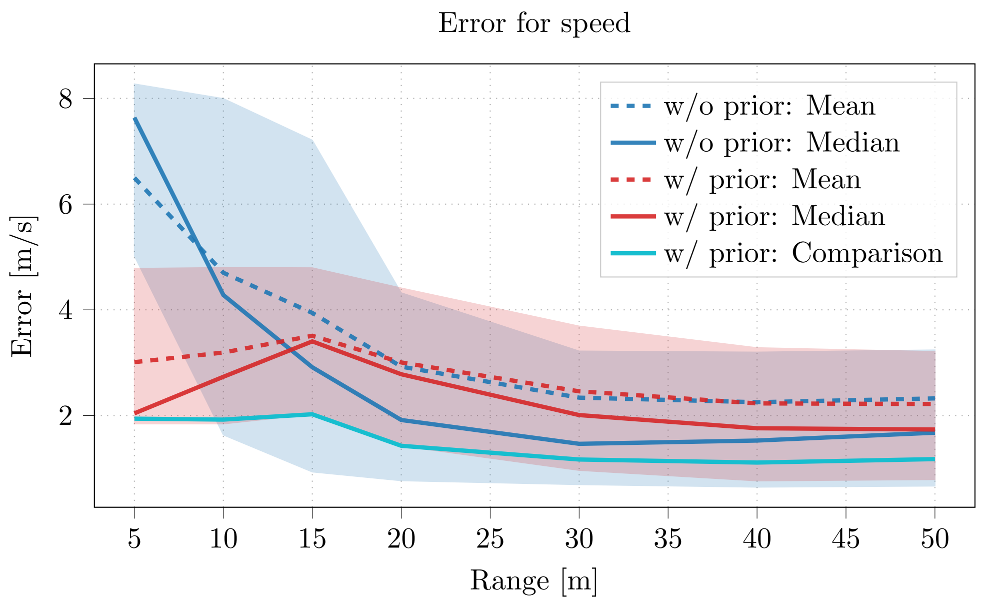


Figure 6.28: Speed error of extracted objects over different ranges. All 10 scenes summed up. (C.17)

Speed The object extraction using the online generated orientation prior information shows a better speed estimation in the near field, whereas the extracted objects without using the prior information show a better mean error in speed at range above 20 m. The P_q error of the objects using the prior information is better than the $P_{50\%}$ error of the objects without using the additional information. This indicates that the worse result of the extraction using the prior is due to the higher number of associated true positives.

The plot shows that the grid's speed estimation works well, regardless of the usage of the online generated orientation prior information. Nevertheless a mean error in speed of approximately 2.5 m/s is not sufficient for driver assistance systems or autonomous driving. Therefore the speed estimation of the dynamic objects has to be enhanced by the object tracking.

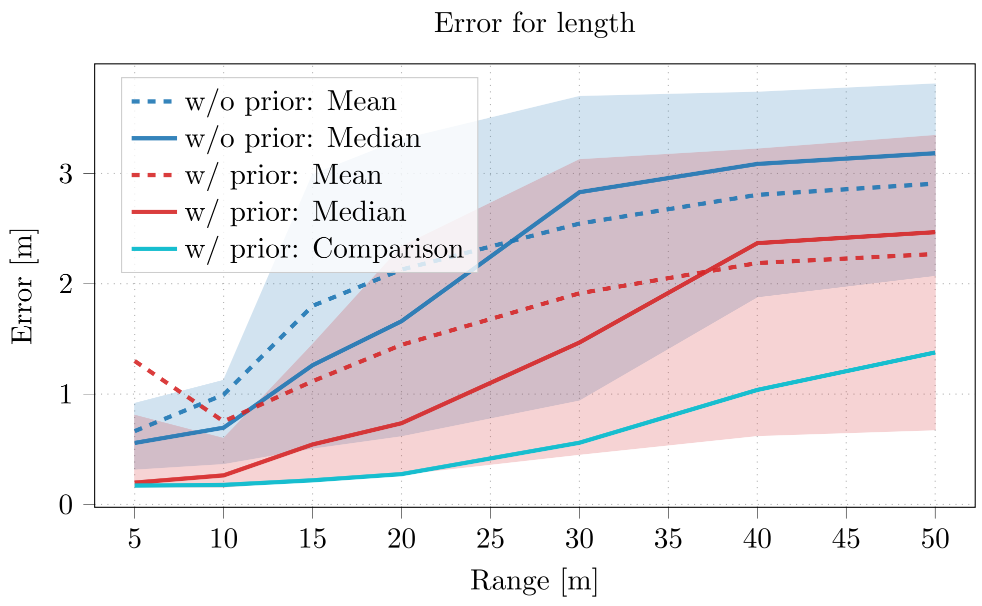


Figure 6.29: Length error of extracted objects over different ranges. All 10 scenes summed up. (C.18)

Length & Width Both errors, in length and width (fig. 6.29 and 6.30), are increasing with larger distances to the ego vehicle. Although this effect is more visible in the length error (as the objects are longer than wide), the reason for both is the same. With a higher distance to the sensor, the probability that only parts of the object can be sensed is higher. The main reasons for that are: The object is leaving the sensors field of view, the object is (partly) hidden by other objects, or the object is sensed too small due to the polar sensing principle (sec. 2.3.1). The object tracking should also lead to better results in terms of the object dimension. For most autonomous driving functions the shown quality in length and width estimation should be sufficient. Especially in the close range ($< 15\,\text{m}$), where the correct dimension estimation is the most important, the extracted objects have a good dimension estimation.

The usage of the online generated prior shows superior results, probably caused by the better orientation estimation, resulting in a more correct orientation of the bounding box, which in turn defines the estimated dimension of the object.

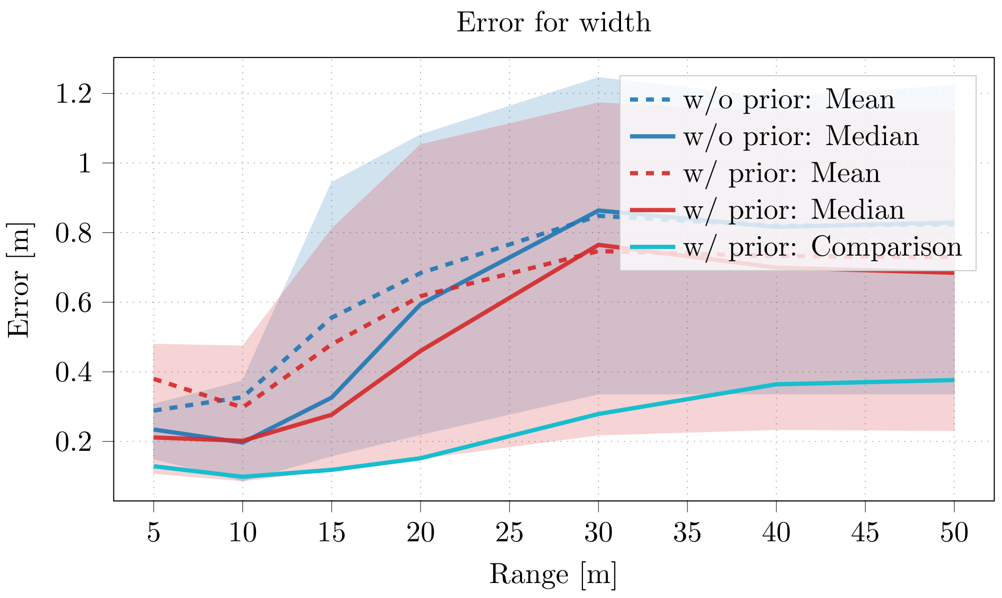


Figure 6.30: Width error of extracted objects over different ranges. All 10 scenes summed up. (C.19)

6.3.4 Object Tracking

The object tracking algorithm (section 5.2) uses the extracted objects as input data. Therefore, we compare the results of the tracking algorithm with the object extraction results of the previous section (6.3.3). In this section we are only using the extracted objects using the online generated orientation prior information, since the previous section showed its superior performance.

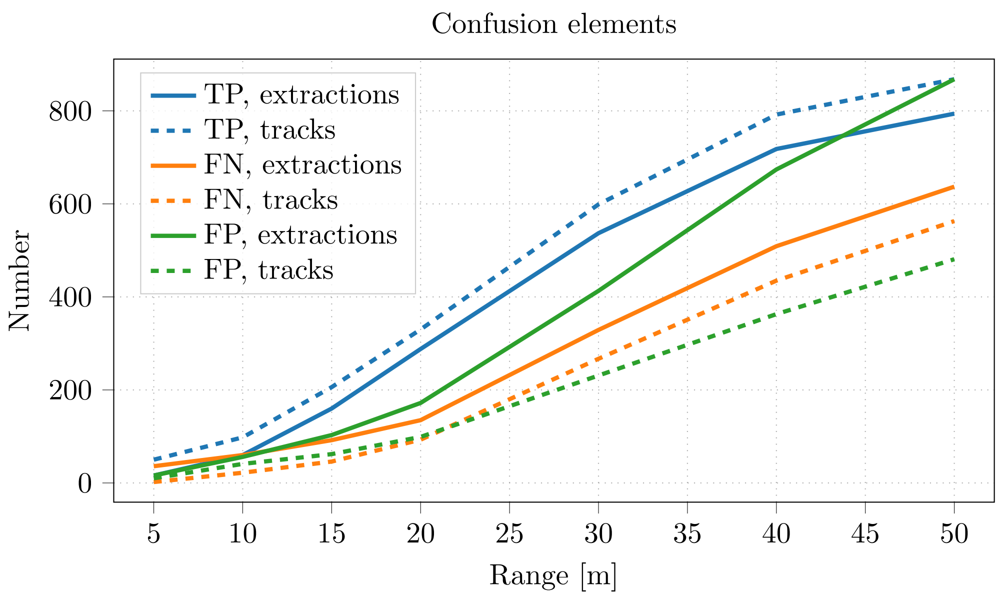


Figure 6.31: True positives, false negatives and false positives for object extractions and tracked objects. All 10 scenes summed up (C.20).

Confusion Matrix Throughout all different ranges, the object tracking algorithm is generating more true positives (and therefore less false negatives) than only applying the object extraction method (fig. 6.31), leading to a better sensitivity (fig. 6.32). In terms of the false positives, applying the object tracking algorithm leads to a reduction of nearly 50% for the complete range. Most false positives are introduced by wrong sensor measurements. As this inconsistencies are not very stable over time, the object management is able to filter the majority of those false positives. This reduction of false positives leads to the better precision as shown in fig. 6.32.

In terms of true positives, false positives and false negatives, the additionally object tracking algorithm outperforms the approach of only extracting objects from the grid. The sole drawback at this point is the additional delay in detecting new objects, introduced by the added initial phase of the tracking algorithm.

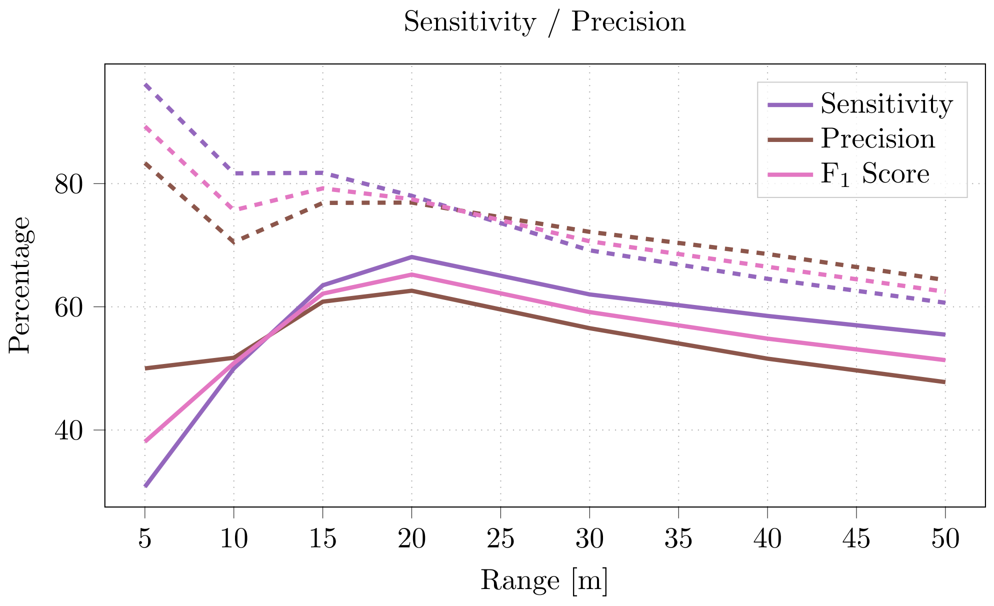


Figure 6.32: Sensitivity and precision rate for object extractions and tracked objects. Solid lines show the result of extracted objects, dashed lines show the result of tracked objects. All 10 scenes summed up (C.20).

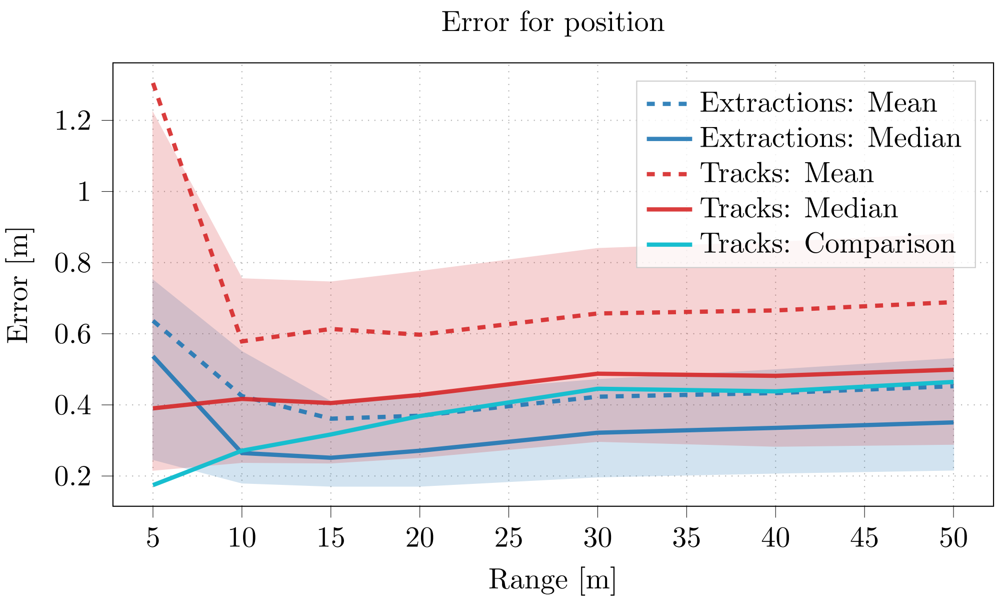


Figure 6.33: Position error of extracted and tracked objects. All 10 scenes summed up. (C.21)

Position The error in position of the tracked objects with a median of ca. 0.5 m and a mean of 0.7 m is significant higher than the error of the object extractions. The only conclusion here is that our prediction model (5.2.5) is introducing a new source of positioning error.

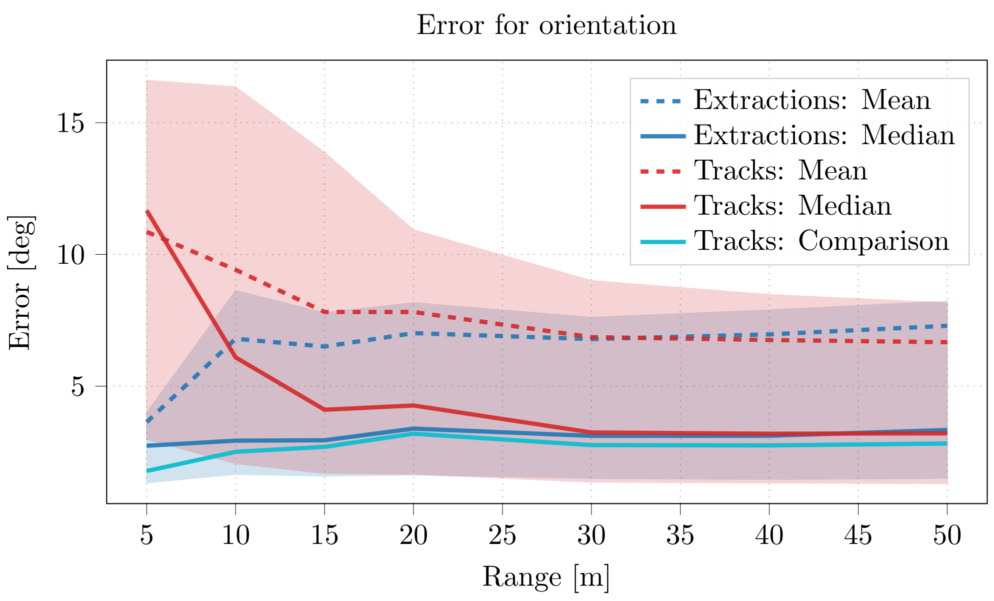


Figure 6.34: Orientation error of extracted and tracked objects. All 10 scenes summed up. (C.22)

Orientation The estimated orientation of the tracked objects have the same error range as the extracted object measurements (median: 3.5° mean: 7.0°) for the ranges of 30 m and above (fig. 6.34). In the closer surroundings of the vehicle the orientation error of the tracked objects is larger than the one of the extracted objects. The comparison percentile of the tracked objects is showing a smaller error of the tracked objects in the near field. The reason for this can be explained as follows: The few correct extracted objects close to the vehicle have a very precise orientation estimation, but the majority of the objects are not detected by the extraction at all. These additional true positives of the object tracking algorithm have a worse orientation estimation and therefore are leading to higher orientation errors of the tracked objects in the near field.

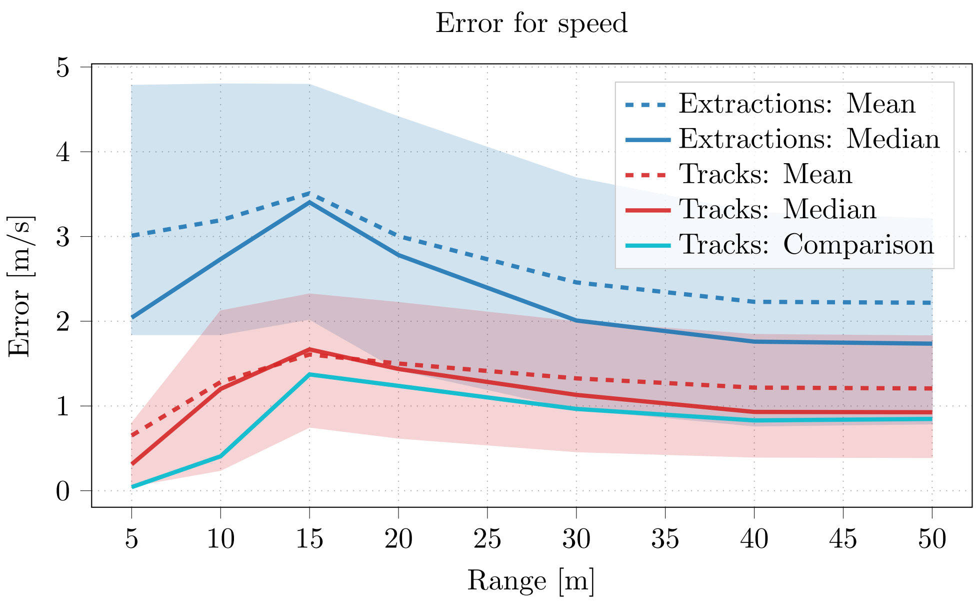


Figure 6.35: Speed error of extracted and tracked objects. All 10 scenes summed up. (C.23)

Speed In terms of speed estimation of the objects, the object tracking approach outperforms the extraction-only approach significantly (fig. 6.35). With a median and mean error of 1 m/s, or 1.5 m/s respectively, the estimated speed signal is well suited for driving functions. In terms of speed estimation the advantage of using an additional object tracking method is clearly visible.

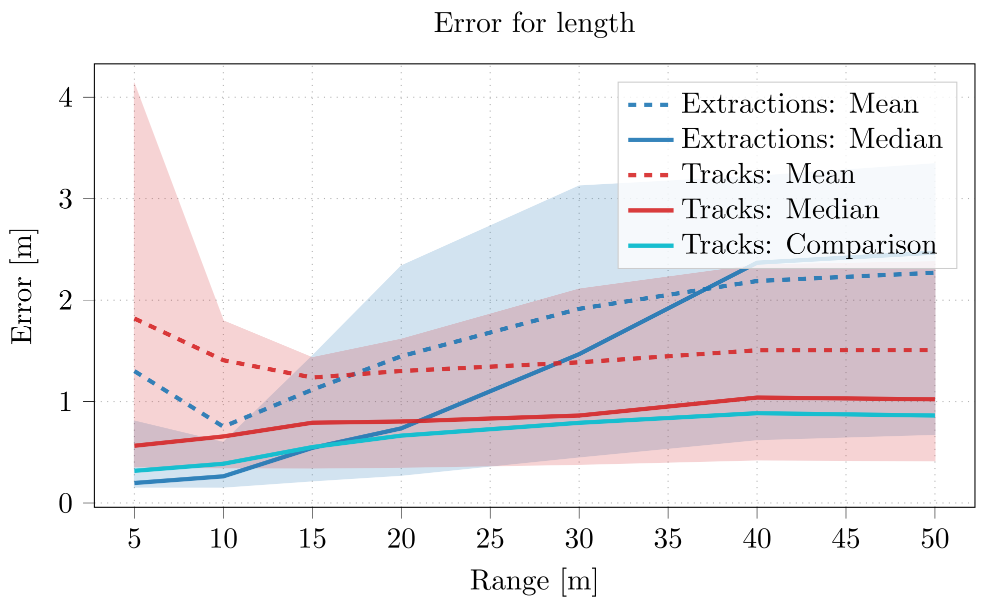


Figure 6.36: Length error of extracted and tracked objects. All 10 scenes summed up. (C.24)

Length & Width The estimated object dimensions of the tracked objects is quite consistent over the different ranges (figures 6.36 and 6.37). This consistency is expected by the restriction in terms of length and width updates during the tracking process (see 5.2.4). Only in the near field the dimension errors of the tracked objects are higher compared to the extracted ones. This effect can be explained with the comparison percentile value: Due to the higher sensitivity of the tracked objects the error is increased. On the other hand the dimension estimation of the extracted object is very good in this area. Nevertheless the dimension estimation error of the tracked object is within an acceptable range.

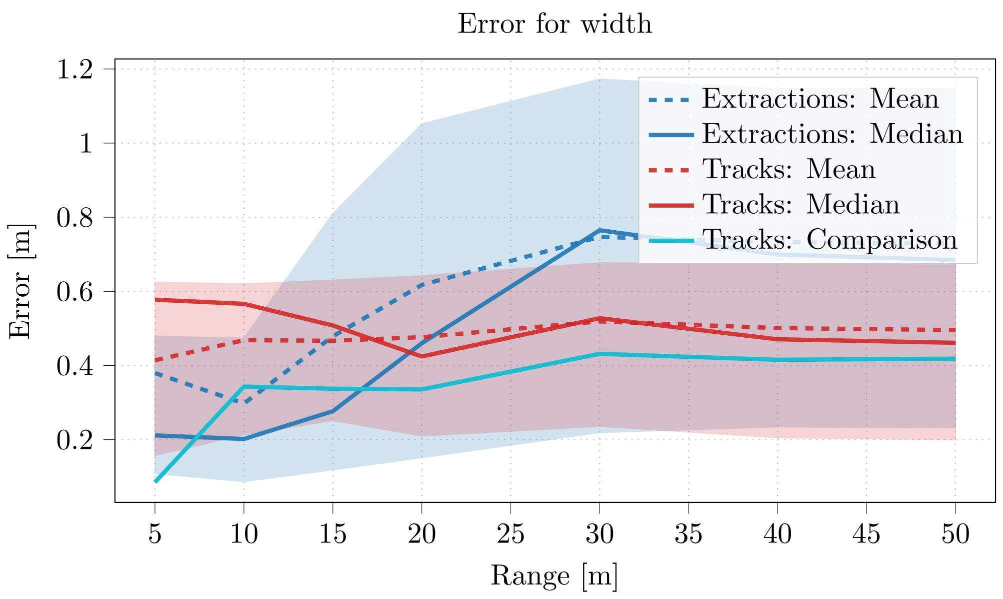


Figure 6.37: Width error of extracted and tracked objects. All 10 scenes summed up. (C.25)

Summary Using an additionally object tracking filter on top of the extracted objects leads to a significant improvement of the detected objects. Especially in terms of sensitivity and precision, the usage of the object tracking filter is superior. The object tracking allows to reduce false positives and stabilizes the existing object tracks. Additionally using the object tracking allows a much better estimation of the speed values.

6.4 Runtime

This section shows the results of our runtime measurements. Our implementation is divided into two parts. The first part, the prediction and update of the dynamic grid, is implemented using CUDA and runs on the *graphics processing unit* (GPU). The processing in this part consists of:

- *Prediction* of the previous map to the current timestamp:
  - Static prediction (3.2.2.1)
  - Dynamic prediction, including particle re-sampling (3.2.2.2)
  - Combination of both prediction (3.2.2.3)
  - Calculation of the online generated orientation prior (3.2.3)
- *Update* of the predicted map with new sensor data:
  - Copy of the sensor data to device (GPU) memory
  - Calculation of the single sensor grids (2.3)
  - Fusion of the sensor grids into a common one (2.3.4)
  - Update of the mass values of the dynamic grid (3.2.1.1)
  - Update of the particle weights based on velocity measurements (3.2.1.2)

We reference this two steps as *Prediction* and *Update* in this section and the total runtime of both steps as *GPU processing.*

The second part of our implementation is implemented using C++ and runs on the *central processing unit* (CPU). The processing in the CPU part consists of:

- *Copy* of the updated dynamic grid from GPU memory to CPU memory
- *Freespace extraction* (4)
- *Object tracking* including the extraction (5)

We use the italic naming of those three steps again as reference in this section. The total runtime of all three steps is referenced as *CPU processing.*

Since both parts, the GPU and CPU processing, are running on different computing resources, they can easily be parallelized. While the CPU part is processing the data

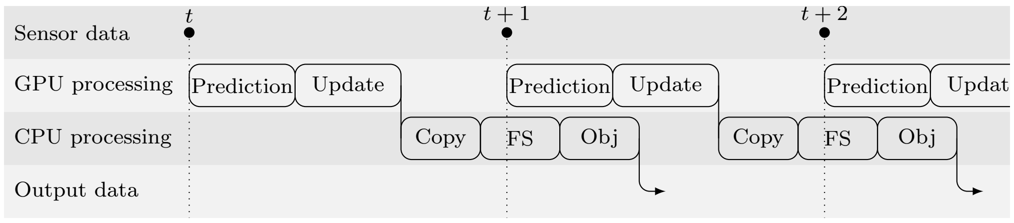


Figure 6.38: Timing of the processing steps and their distribution between GPU and CPU. The row labeled *Sensor data* shows the arrival of new sensor data, the bottom row *Output data* indicates the transmitting of the extracted freespace and object list to the driving function.

of time t, the GPU part can already process the data of the next timestamp $t + 1$ (see figure 6.38).

The runtime measurements presented in this section, are using the same configuration as in previous evaluation (cellsize of 0.125 m, gridsize of 120 m, one particle per cell on average, three radar sensors and one lidar sensor).

6.4.1 Developer Notebook

In this subsection we present the runtime of the algorithm on a modern developer notebook.

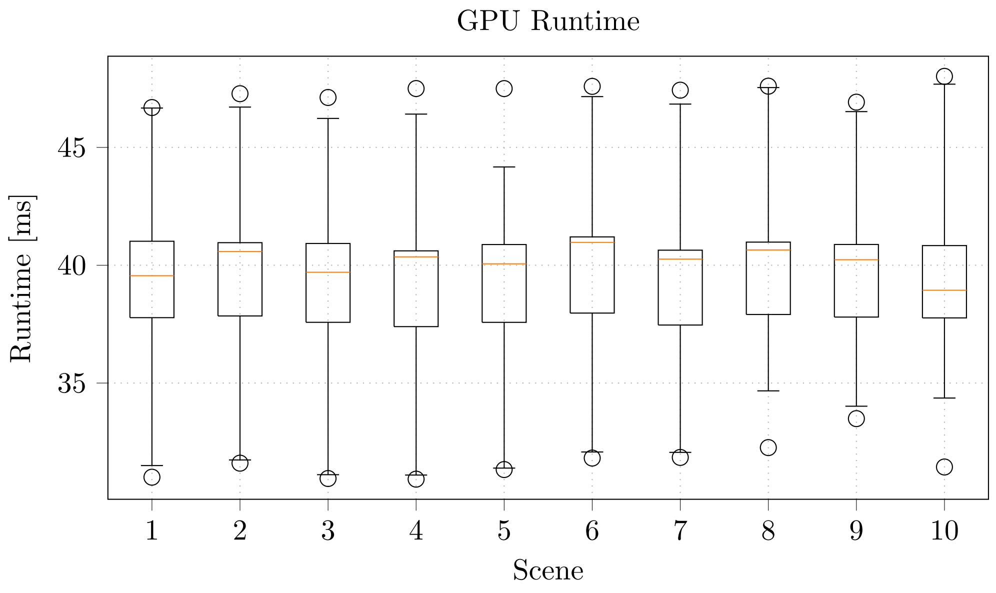


Figure 6.39: Box and whisker plots of the GPU processing runtime on the developer notebook for the different scenes. The box extends from the 25% to the 75% percentile, whereas the whiskers extend from the 0.3% to the 99.7% percentile. The median is displayed as the orange line.

Figure 6.39 shows the runtime of the *GPU processing* over the various scenes. Although the scenes are quite different, the runtime stays relatively constant. We attribute this to the fact that the number of operations performed in the *GPU processing*, especially in the *Prediction*, is almost constant regardless of the environment. Figure 6.40 supports this thesis by showing the runtime of the *GPU processing* over time for one scene. The prediction runtime stays constant, whereas the update runtime has some variations. The differences in the update runtime can be explained by the varying number of measurements on the sensor input.

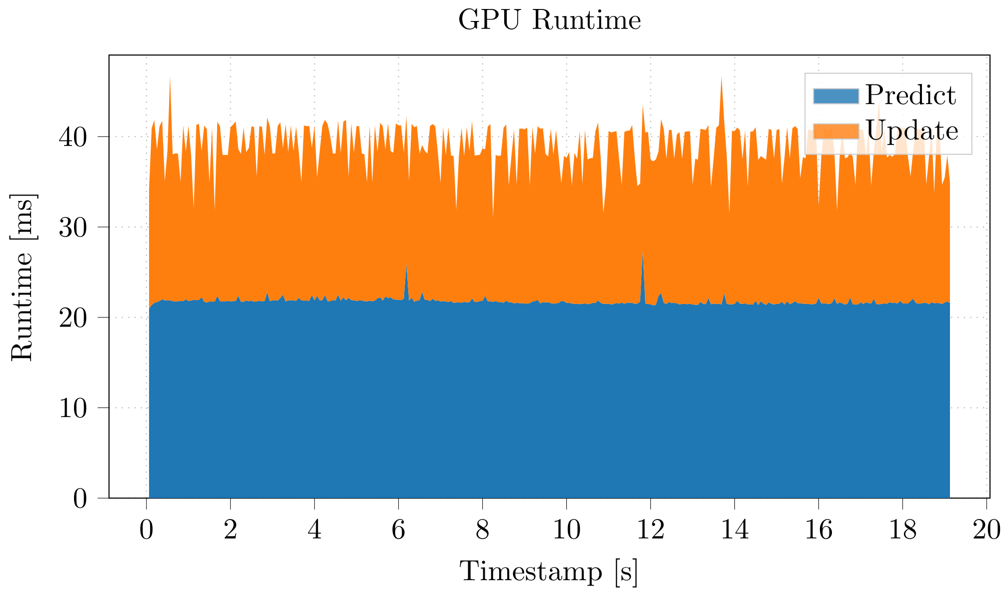


Figure 6.40: Runtime of the GPU processing parts of scene 1 over time on the developer notebook.

The runtime of the *CPU processing* is shown in figures 6.41 and 6.42. We see a relative constant runtime over the different scenes as well as over time for one scene. We want to point out, that the data transfer from the GPU to the CPU memory of the dynamic grid output takes a significant amount of time.

In summary we can clearly see that the *GPU processing* as well as the *CPU processing* take less than 50 ms of time. This runtime performance allows the algorithm to run at 20 Hz using the chosen sensor configuration on a standard notebook hardware.

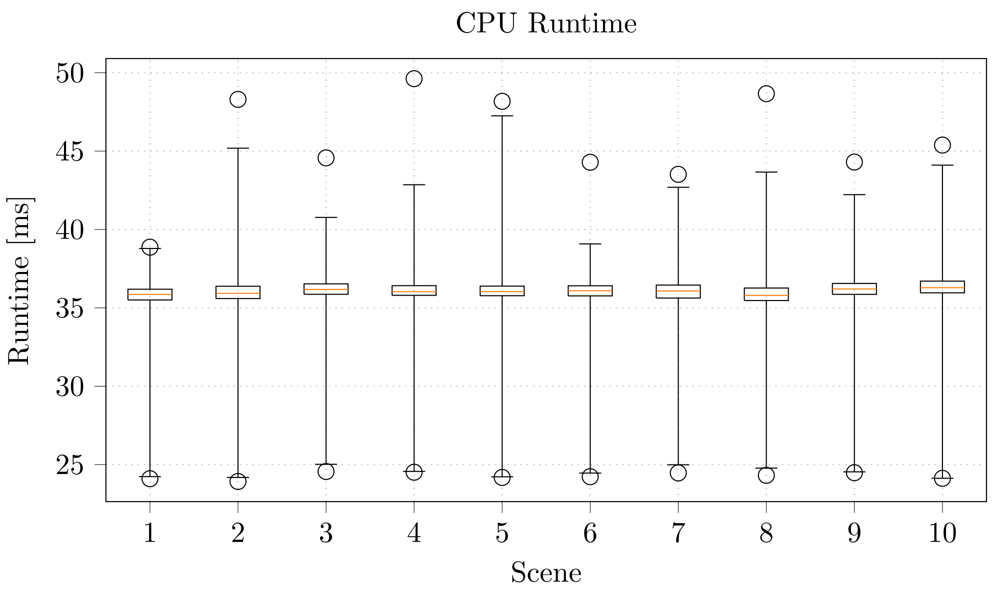


Figure 6.41: Box and whisker plots of the CPU processing runtime on the developer notebook for the different scenes. The box extends from the 25% to the 75% percentile, whereas the whiskers extend from the 0.3% to the 99.7% percentile. The median is displayed as the orange line.

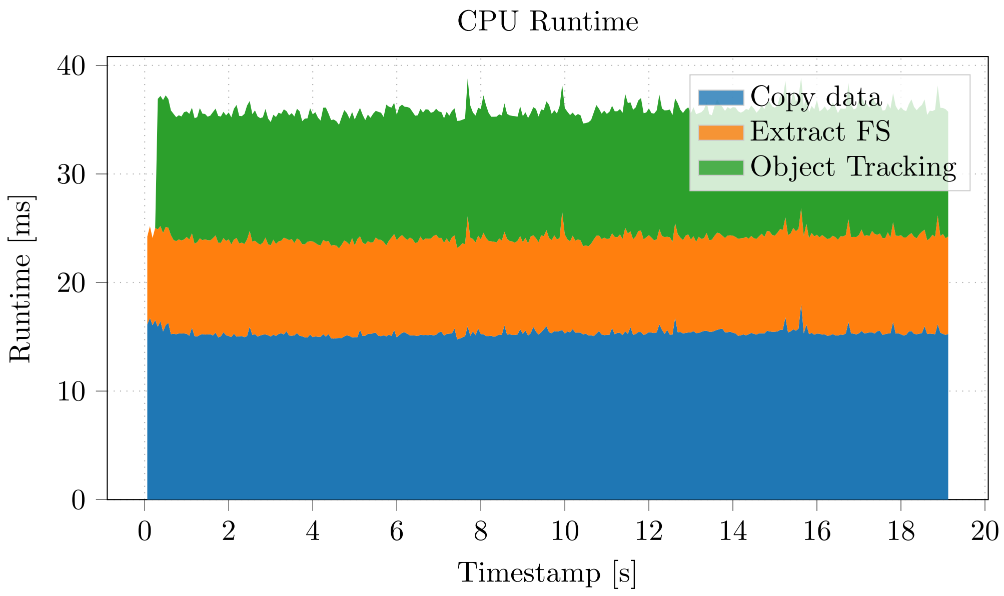


Figure 6.42: Runtime of the CPU processing parts of scene 1 over time on the developer notebook.

6.4.2 Electronic Control Unit (ECU)

The runtime evaluation presented in this section was carried out on a prototype of an upcoming high performance ECU. Although we use the same dataset as input, we reduced the grid size from 120 m × 120 m to 108 m × 108 m here. Otherwise the configuration and algorithms are unchanged.

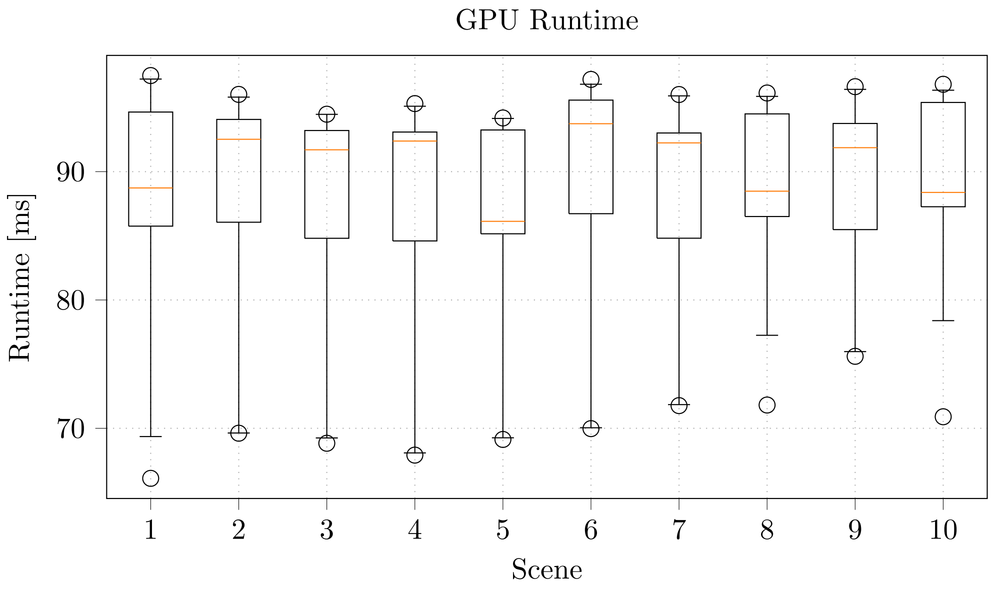


Figure 6.43: Box and whisker plots of the GPU processing runtime on the ECU for the different scenes. The box extends from the 25% to the 75% percentile, whereas the whiskers extend from the 0.3% to the 99.7% percentile. The median is displayed as the orange line.

Figures 6.43 and 6.44 show the same runtime characteristics of the *GPU processing* on the ECU as discussed in the previous section on the developer notebook: the runtime of the processing is very constant over different scenes and time. The only difference is the absolute runtime on the ECU: we see a runtime of nearly 100 ms, whereas the notebook runtime was below 50 ms.

The runtime measurements of the *CPU processing* (figure 6.45 and 6.46) show also the same behavior as on the developer notebook, with the absolute runtime being the only difference.

With a runtime of below 100 ms, for both the GPU and CPU processing, we achieve a frame-rate of 10 Hz on an ECU with our algorithm. Note that this runtime on the ECU contains the sensor fusion of one lidar and three radar sensor on a grid map, as well as the freespace and object tracking algorithm.

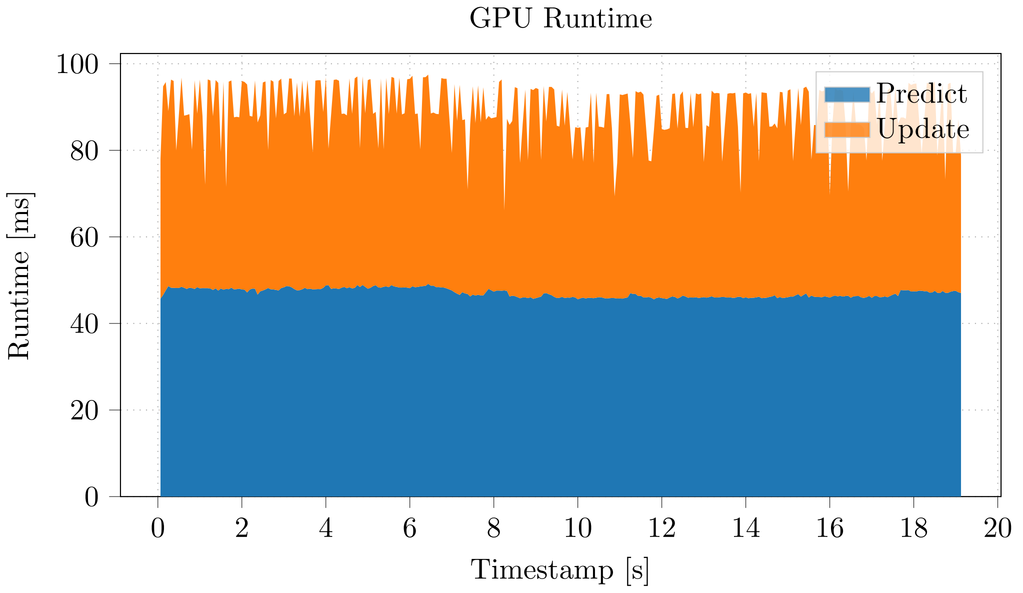


Figure 6.44: Runtime of the GPU processing parts of scene 1 over time on the ECU.

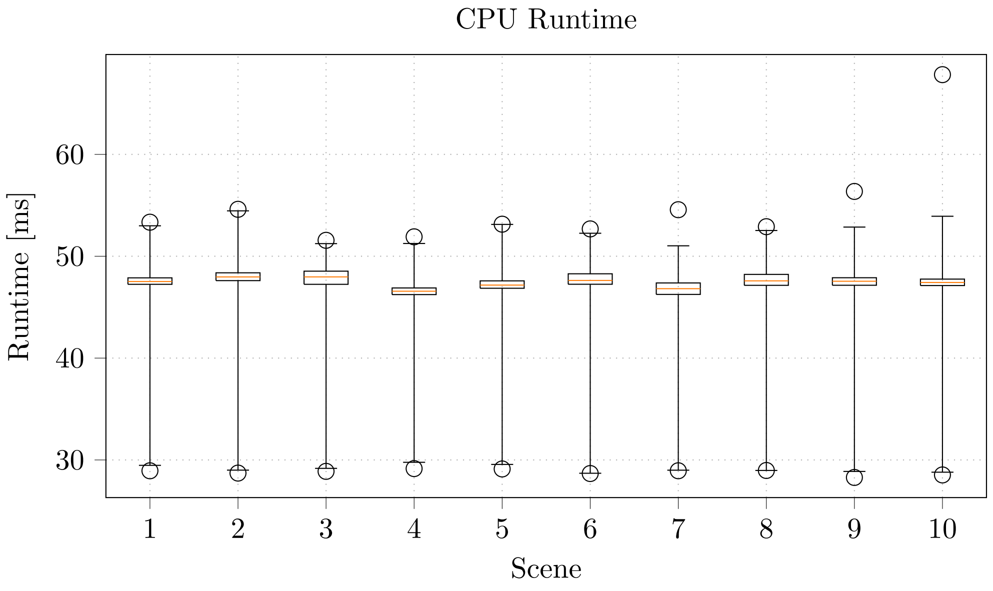


Figure 6.45: Box and whisker plots of the CPU processing runtime on the ECU for the different scenes. The box extends from the 25% to the 75% percentile, whereas the whiskers extend from the 0.3% to the 99.7% percentile. The median is displayed as the orange line.

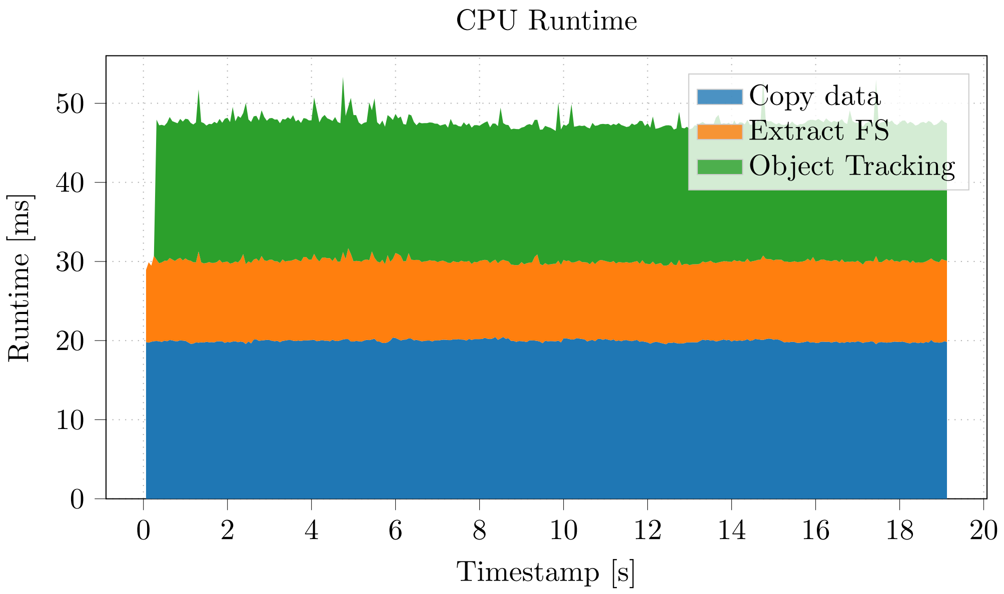


Figure 6.46: Runtime of the CPU processing parts of scene 1 over time on the ECU.

7 Conclusion & Outlook

Environment perception plays a crucial role in advanced driver assistance systems and autonomous driving. Therefore it is essential to invest in research and development of perception algorithms. Every processing step in the perception pipeline, from the sensor preprocessing, to feature extraction, to fusion and to object tracking, has to be optimized in order to get the best results for the driving functions. Even with new sensor generations, sensor detections will always contain certain errors, such as measurement noise and misdetections. A powerful perception pipeline has to handle such circumstances to provide a good environment representation.

In this work we presented a grid-based fusion approach for multiple radar and lidar sensors. This grid-based method makes no, or only very few, assumptions about the environment and is therefore usable in a very wide range of applications. Our approach is estimating the occupancy as well as the velocity of each cell using particles. The presented algorithm architecture is highly scalable regarding the number of used sensors.

We presented a grid-based fusion approach, which also acts as an abstraction layer for the sensor data. The dynamic grid can handle arbitrary sensor data, as long as it can provide its output as an occupancy grid. For the two most used range based sensors, radar and lidar, we presented such inverse sensor models. These sensor models estimate the occupancy of grid cells using single sensor measurements.

Furthermore we introduced the so called *online generated orientation prior* and showed how it enhances the perception output. Using this type of prior information during particle initialization allows a faster convergence of the particles, which are used to estimate the cell velocities. This orientation information is calculated online, so no further input for the algorithm is required.

Although we think that a grid-based representation of the environment is well suited for autonomous driving tasks, e.g. collision avoidance or path planning, we understand the demand of a more compressed representation. Therefore we presented two extraction methods to provide well established interfaces for the environment representation: A freespace contour list for the static environment and an object list for the dynamic environment. To improve the quality of the object list, we proposed a standard object tracking method using an extended Kalman filter, taking advantage of the knowledge about a vehicle motion model.

The proposed method in this work is well suited to be parallelized for running on graphics processing units (GPUs). Since the prediction and update of the single cells are independent of each other, each cell (or particle) can be calculated in parallel. In the runtime evaluation we showed that our proposed algorithm is real time capable on upcoming electronic control units (ECU). Future ECUs (e.g. [41]) will allow even faster execution of such parallelized algorithms, allowing the fusion of more sensors and/or using bigger grid maps.

We validated the proposed method using an open data set and showed that this grid-based fusion works with a set of four sensors and that the sensor fusion outperforms single sensor configurations. The evaluation also showed that there is still room for further improvements. Therefore we are looking forward to upcoming publications in this area.

Our proposed algorithm is based on range based sensors only, especially radar and lidar sensors. A possible addition is the integration of camera sensors. The integration of stereo cameras should be possible similar to the lidar implementation, because both sensors are providing a dense point cloud. Using mono cameras on the other hand is more challenging, since the estimated range of detected objects is very noisy. A big advantage of camera sensors is their classification capability. Therefore we expect cameras to improve the dynamic/static classification as supplementary sensors to radar and lidar sensors.

Another interesting research question is how to extract lane information from the grid map. The proposed freespace extraction can act as a starting point for that. Furthermore it would be interesting whether the online generated orientation prior can help to improve the lane estimation. In order to improve the velocity estimation of the cells, an extension of the online generated orientation prior by the absolute speed value could possibly lead to better results.

Bibliography

[1] A. J. Hawkins. Waymo tells riders that ’completely driverless’ vehicles are on the way. `https://www.theverge.com/2019/10/10/20907901/waymo-driverless-cars-email-customers-arizona`, 2019. [Online; accessed 2021-01-17].

[2] K. Wiggers. Uber gets California DMV license to test self-driving cars on public roads. `https://venturebeat.com/2020/02/05/uber-acquires-dmv-license-to-test-self-driving-cars-on-public-california-roads/`, 2020. [Online; accessed 2021-01-17].

[3] A. AG. The new Audi A8 - conditional automated at level 3. `https://www.audi-mediacenter.com/en/on-autopilot-into-the-future-the-audi-vision-of-autonomous-driving-9305/the-new-audi-a8-conditional-automated-at-level-3-9307`, 2017. [Online; accessed 2020-02-12].

[4] J. Golson and D. Bohn. All new Tesla cars now have hardware for ‘full self-driving capabilities’. `https://www.theverge.com/2016/10/19/13340938/tesla-autopilot-update-model-3-elon-musk-update`, 2016. [Online; accessed 2021-01-17].

[5] W. C. Timo Daum, Andreas Knie. Vom Fahren zum Gefahrenwerden. Automatisierte Shuttles – der europäische Weg? `https://www.heise.de/hintergrund/Vom-Fahren-zum-Gefahrenwerden-Automatisierte-Shuttles-der-europaeische-Weg-6034403.html`, 2021. [Online; accessed 2021-10-06].

[6] S. Aktuell. Friedrichshafen: Selbstfahrende Shuttlebusse in Innenstadt. `https://www.swr.de/swraktuell/baden-wuerttemberg/friedrichshafen/projekt-selbstfahrende-shuttlebusse-mit-zf-friedrichshafen-beteiligung-geht-in-neue-runde-100.html`, 2021. [Online; accessed 2021-10-06].

[7] C. Thompson. New details about the fatal Tesla Autopilot crash reveal the driver’s last minutes. `https://www.businessinsider.com/details-about-the-fatal-tesla-autopilot-accident-released-2017-6?r=DE&IR=T`, 2017. [Online; accessed 2020-10-05].

[8] S. Dent. Uber self-driving car involved in fatal crash couldn’t detect jaywalkers. `https://www.engadget.com/2019/11/06/uber-self-driving-car-fatal-accident-ntsb/`, 2019. [Online; accessed 2020-02-12].

[9] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, 2:116–12, 1985.

[10] R. Garcia, O. Aycard, T. Vu, and M. Ahrholdt. High level sensor data fusion for automotive applications using occupancy grids. *2008 10th International Conference on Control, Automation, Robotics and Vision*, pages 530–535, 2008. `doi:10.1109/ICARCV.2008.4795574`.

[11] M. Bouzouraa and U. Hofmann. Fusion of occupancy grid mapping and model based object tracking for driver assistance systems using laser and radar sensors. *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 294–300, 2010. `doi:10.1109/IVS.2010.5548106`.

[12] R. Jungnickel and F. Korf. Object tracking and dynamic estimation on evidential grids. *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2310–2316, 2014. `doi:10.1109/ITSC.2014.6958060`.

[13] T. Yuan, D. Nuss, and G. Krehl. Fundamental Properties of Dynamic Occupancy Grid Systems. *IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 153–156, 2015. `doi:10.1109/CAMSAP.2015.7383759`.

[14] C. Chen, C. Tay, C. Laugier, and K. Mekhnacha. Dynamic environment modeling with gridmap: A multiple-object tracking application. *9th International Conference on Control, Automation, Robotics and Vision, 2006, ICARCV '06*, pages 1–6, 2006. `doi:10.1109/ICARCV.2006.345399`.

[15] T. Gindele, S. Brechtel, J. Schroder, and R. Dillmann. Bayesian occupancy grid filter for dynamic environments using prior map knowledge. In *2009 IEEE Intelligent Vehicles Symposium*, pages 669–676, June 2009. `doi:10.1109/IVS.2009.5164357`.

[16] A. Nègre, L. Rummelhard, and C. Laugier. Hybrid sampling bayesian occupancy filter. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 1307–1312, June 2014. `doi:10.1109/IVS.2014.6856554`.

[17] G. Tanzmeister and D. Wollherr. Evidential grid-based tracking and mapping. *IEEE Transactions on Intelligent Transportation Systems*, 18(6):1454–1467, 2017. `doi:10.1109/TITS.2016.2608919`.

[18] S. Steyer, G. Tanzmeister, and D. Wollherr. Grid-based environment estimation using evidential mapping and particle tracking. *IEEE Transactions on Intelligent Vehicles*, 3(3):384–396, 2018. `doi:10.1109/TIV.2018.2843130`.

[19] F. Homm, N. Kaempchen, J. Ota, and D. Burschka. Efficient occupancy grid computation on the gpu with lidar and radar for road boundary detection. In *2010 IEEE Intelligent Vehicles Symposium*, pages 1006–1013, 2010. `doi:10.1109/IVS.2010.5548091`.

[20] K. Werber, M. Rapp, J. Klappstein, M. Hahn, J. Dickmann, K. Dietmayer, and C. Waldschmidt. Automotive radar gridmap representations. In *2015 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 1–4. IEEE, 2015.

[21] M. Slutsky and D. Dobkin. Dual inverse sensor model for radar occupancy grids. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1760–1767, 2019. `doi:10.1109/IVS.2019.8813772`.

[22] L. Lindenmaier and V. Tihanyi. Comparison of different tracking approaches on pre-fused data for automotive perception system. In *2019 IEEE 19th International Symposium on Computational Intelligence and Informatics and 7th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Sciences and Robotics (CINTI-MACRo)*, pages 000199–000204, 2019. `doi:10.1109/CINTI-MACRo49179.2019.9105224`.

[23] C. Diehl, E. Feicho, A. Schwambach, T. Dammeier, E. Mares, and T. Bertram. Radar-based dynamic occupancy grid mapping and object detection. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2020. `doi:10.1109/ITSC45102.2020.9294626`.

[24] S. Steyer, G. Tanzmeister, C. Lenk, V. Dallabetta, and D. Wollherr. Data association for grid-based object tracking using particle labeling. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3036–3043, 2018. `doi:10.1109/ITSC.2018.8569511`.

[25] S. Steyer, C. Lenk, D. Kellner, G. Tanzmeister, and D. Wollherr. Grid-based object tracking with nonlinear dynamic state and shape estimation. *IEEE Transactions on Intelligent Transportation Systems*, 21(7):2874–2893, 2020. `doi:10.1109/TITS.2019.2921248`.

[26] S. J. Steyer. *Grid-based object tracking.* Dissertation, Technische Universität München, München, 2021.

[27] J. Wessner and W. Utschick. Extending occupancy grid mapping for dynamic environments. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 701–707, 2018. `doi:10.1109/IVS.2018.8500362`.

[28] J. Wessner and W. Utschick. Online orientation prior for dynamic grid-maps. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 818–823, 2021. `doi:10.1109/IV48863.2021.9575374`.

[29] Y. Jazi. Grid-based object tracking in autonomous driving. Master's thesis, University of Passau, 1 2020.

[30] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* The MIT Press, 2005.

[31] Velodyne Lidar, Inc. Puck lidar sensor, high-value surround lidar. `https://velodynelidar.com/products/puck/`, 2020. [Online; accessed 2020-12-30].

[32] M. Himmelsbach, F. v. Hundelshausen, and H. . Wuensche. Fast segmentation of 3d point clouds for ground vehicles. In *2010 IEEE Intelligent Vehicles Symposium*, pages 560–565, June 2010. `doi:10.1109/IVS.2010.5548059`.

[33] C. Wolff. The radar equation - radartutorial. `https://www.radartutorial.eu/01.basics/The%20Radar%20Range%20Equation.en.html`, 2020. [Online; accessed 2020-12-30].

[34] I. J. H. Ender. Introduction to radar part i. page 53 ff., 2011.

[35] A. Huamán. Opencv: More morphology transformations. `https://docs.opencv.org/4.5.2/d3/dbe/tutorial_opening_closing_hats.html`, 2021. [Online; accessed 2021-07-25].

[36] A. Huamán. Opencv: Eroding and dilating. `https://docs.opencv.org/4.5.2/db/df6/tutorial_erosion_dilatation.html`, 2021. [Online; accessed 2021-07-25].

[37] S. Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[38] C. Grana, D. Borghesani, and R. Cucchiara. Optimized block-based connected components labeling with decision trees. *IEEE Transactions on Image Processing*, 19(6):1596–1609, 2010.

[39] G. T. Toussaint. Solving geometric problems with the rotating calipers. In *Proc. IEEE Melecon*, volume 83, page A10, 1983.

[40] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[41] J. Wienrich. Zf proai: The source of vehicle intelligence - zf. `https://www.zf.com/mobile/en/stories_30336.html`, 2021. [Online; accessed 2021-12-14].

A Coordinate Transformations

A.1 Location Transformation

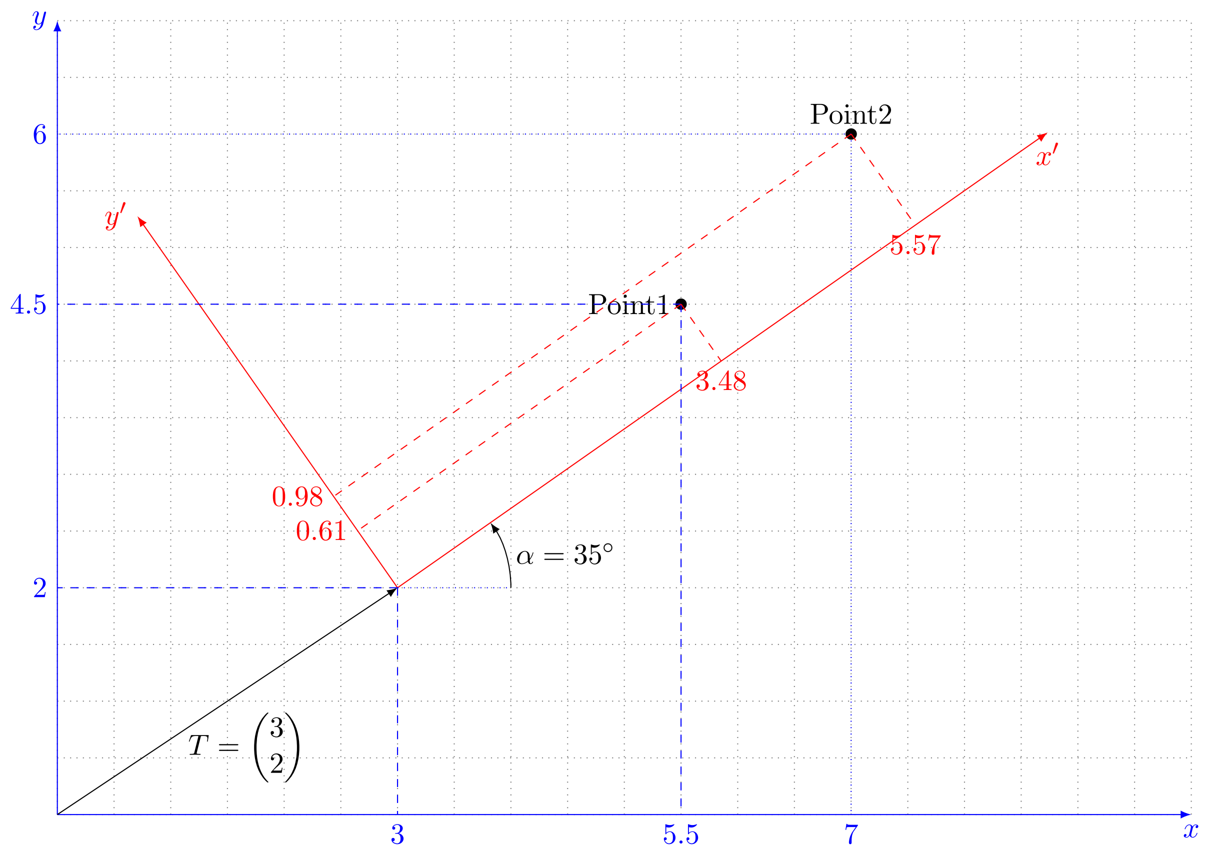


Figure A.1: Two example points shown in an outer (blue) and an inner (red) coordinate system

Inner to Outer System

This section describes the coordinate transformation from an inner to an outer coordinate system. This transformation corresponds to a transformation of the red coordinate system into the blue one shown in figure A.1. First we have to rotate the inner system by an angle of α clockwise, which corresponds to a point rotation by α anti-clockwise.

Therefore the rotation matrix $R_{i\to o}$ is:

$$R_{i\to o} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \tag{A.1}$$

After the rotation the coordinates have to be shifted to match the outer coordinate system by adding the origin offset T:

$$X_o = R_{i\to o}\ X_i + T \tag{A.2}$$

This equation can be rewritten to one matrix multiplication using the transformation matrix $M_{i\to o}$:

$$M_{i\to o} = \begin{pmatrix} R_{i\to o} & T \\ 0 & 1 \end{pmatrix} \tag{A.3}$$

$$\begin{pmatrix} X_o \\ 1 \end{pmatrix} = M_{i\to o} \begin{pmatrix} X_i \\ 1 \end{pmatrix} \tag{A.4}$$

Example We use Point1 from figure A.1 as an example. The information required for the transformation is:

$$R_{i\to o} = \begin{pmatrix} \cos(35°) & -\sin(35°) \\ \sin(35°) & \cos(35°) \end{pmatrix} \approx \begin{pmatrix} 0.8192 & -0.5736 \\ 0.5736 & 0.8192 \end{pmatrix}$$

$$T = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

$$X_i = \begin{pmatrix} 3.48 \\ 0.61 \end{pmatrix}$$

Inserting these values in the transformation equation (A.2) results in the coordinates of that point in the outer (blue) system:

$$\begin{aligned} X_o &= \begin{pmatrix} 0.8192 & -0.5736 \\ 0.5736 & 0.8192 \end{pmatrix} \begin{pmatrix} 3.48 \\ 0.61 \end{pmatrix} + \begin{pmatrix} 3 \\ 2 \end{pmatrix} \\ &= \begin{pmatrix} 2.5 \\ 2.5 \end{pmatrix} + \begin{pmatrix} 3 \\ 2 \end{pmatrix} \\ X_o &= \begin{pmatrix} 5.5 \\ 4.5 \end{pmatrix} \end{aligned}$$

Outer to Inner System

To convert points from the outer (blue) to the inner (red) coordinate system, we apply the inverse operations (in reversed order) as in the previous section (see A.1). First we subtract the coordinate offset T and then we rotate by $-\alpha$:

$$X_i = R_{o\to i}\ (X_o - T)$$

$$X_i = R_{o\to i}\ X_o - R_{o\to i}\ T \tag{A.5}$$

with:

$$\begin{aligned} R_{o\to i} &= \begin{pmatrix} \cos(-\alpha) & -\sin(-\alpha) \\ \sin(-\alpha) & \cos(-\alpha) \end{pmatrix} \\ &= \begin{pmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{pmatrix} \\ &= R_{i\to o}^{-1} \end{aligned}$$

As in the opposite tranformation, this transformation formula (A.5) can also be written with only one matrix multiplication:

$$M_{o\to i} = \begin{pmatrix} R_{o\to i} & (-R_{o\to i}\, T) \\ 0 & 1 \end{pmatrix} \tag{A.6}$$

$$\begin{pmatrix} X_i \\ 1 \end{pmatrix} = M_{o\to i} \begin{pmatrix} X_o \\ 1 \end{pmatrix} \tag{A.7}$$

Example For this example we use Point2 of figure A.1:

$$\begin{aligned} R_{o\to i} &= \begin{pmatrix} \cos(35°) & \sin(35°) \\ -\sin(35°) & \cos(35°) \end{pmatrix} \approx \begin{pmatrix} 0.8192 & 0.5736 \\ -0.5736 & 0.8192 \end{pmatrix} \\ T &= \begin{pmatrix} 3 \\ 2 \end{pmatrix} \\ X_o &= \begin{pmatrix} 7 \\ 6 \end{pmatrix} \end{aligned}$$

Using the transformation equation A.5, we get:

$$\begin{aligned} X_i &= \begin{pmatrix} 0.8192 & 0.5736 \\ -0.5736 & 0.8192 \end{pmatrix} \begin{pmatrix} 7 \\ 6 \end{pmatrix} - \begin{pmatrix} 0.8192 & 0.5736 \\ -0.5736 & 0.8192 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix} \\ &= \begin{pmatrix} 9.176 \\ 0.9 \end{pmatrix} - \begin{pmatrix} 3.605 \\ -0.082 \end{pmatrix} \\ X_i &= \begin{pmatrix} 5.57 \\ 0.98 \end{pmatrix} \end{aligned}$$

A.2 Velocity Transformation

For the velocity transformation, the transformation equations (A.2 and A.5) have to be derived with respect to the time.

Inner to Outer System

The transformation formula for velocities from the inner (red) into the outer (blue) coordinate system is a derivative of A.2:

$$\begin{aligned}\dot{X}_o &= \frac{d}{dt}(R_{i\to o}\ X_i + T)\\ &= \dot{R}_{i\to o}\ X_i + R_{i\to o}\ \dot{X}_i + \dot{T}\end{aligned} \tag{A.8}$$

$\dot{R}_{i\to o}$ is the derivative of the rotation matrix and denotes the change in orientation:

$$\begin{aligned}\dot{R}_{i\to o} &= \frac{d}{dt}\begin{pmatrix}\cos(\alpha) & -\sin(\alpha)\\ \sin(\alpha) & \cos(\alpha)\end{pmatrix}\\ &= \begin{pmatrix}-\sin(\alpha)\ \dot{\alpha} & -\cos(\alpha)\ \dot{\alpha}\\ \cos(\alpha)\ \dot{\alpha} & -\sin(\alpha)\ \dot{\alpha}\end{pmatrix}\end{aligned}$$

The transformation of position and velocity can be combined into one single matrix multiplication:

$$\begin{aligned}\begin{pmatrix}X_o\\ \dot{X}_o\\ 1\end{pmatrix} &= N_{i\to o}\begin{pmatrix}X_i\\ \dot{X}_i\\ 1\end{pmatrix}\\ &= \begin{pmatrix}R_{i\to o} & 0 & T\\ \dot{R}_{i\to o} & R_{i\to o} & \dot{T}\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}X_i\\ \dot{X}_i\\ 1\end{pmatrix}\end{aligned}$$

Outer to Inner System

The derivation for the reverse transformation is analogously:

$$\begin{aligned}\dot{X}_i &= \frac{d}{dt}(R_{o\to i}\ X_o - R_{o\to i}T)\\ &= \dot{R}_{o\to i}\ X_o + R_{o\to i}\ \dot{X}_o - \dot{R}_{o\to i}\ T - R_{o\to i}\ \dot{T}\end{aligned} \tag{A.9}$$

with:

$$\begin{aligned}\dot{R}_{o\to i} &= \frac{d}{dt}\begin{pmatrix}\cos(\alpha) & \sin(\alpha)\\ -\sin(\alpha) & \cos(\alpha)\end{pmatrix}\\ &= \begin{pmatrix}-\sin(\alpha)\ \dot{\alpha} & \cos(\alpha)\ \dot{\alpha}\\ -\cos(\alpha)\ \dot{\alpha} & -\sin(\alpha)\ \dot{\alpha}\end{pmatrix} \neq \dot{R}_{i\to o}^{-1}\end{aligned}$$

Written as single matrix multiplication:

$$\begin{aligned}\begin{pmatrix}X_i\\ \dot{X}_i\\ 1\end{pmatrix} &= N_{o\to i}\begin{pmatrix}X_o\\ \dot{X}_o\\ 1\end{pmatrix}\\ &= \begin{pmatrix}R_{o\to i} & 0 & (-R_{o\to i}\ T)\\ \dot{R}_{o\to i} & R_{o\to i} & (-\dot{R}_{o\to i}\ T - R_{o\to i}\ \dot{T})\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}X_o\\ \dot{X}_o\\ 1\end{pmatrix}\end{aligned}$$

A.3 Acceleration Transformation

For the acceleration transformation, the velocity transformation equations (A.8 and A.9) have to be differentiated again with respect to time.

Inner to Outer System

The transformation formula for accelerations from the inner (red) into the outer (blue) coordinate system is a derivative of A.8:

$$\begin{aligned}\ddot{X}_o &= \frac{d}{dt}\left(\dot{R}_{i\to o}\ X_i + R_{i\to o}\ \dot{X}_i + \dot{T}\right)\\ &= \ddot{R}_{i\to o}\ X_i + \dot{R}_{i\to o}\ \dot{X}_i + \dot{R}_{i\to o}\ \dot{X}_i + R_{i\to o}\ \ddot{X}_i + \ddot{T}\\ &= \ddot{R}_{i\to o}\ X_i + 2\ \dot{R}_{i\to o}\ \dot{X}_i + R_{i\to o}\ \ddot{X}_i + \ddot{T}\end{aligned}$$

$\ddot{R}_{i\to o}$ is the second order derivative of the rotation matrix and denotes the change in orientation velocity:

$$\begin{aligned}\ddot{R}_{i\to o} &= \frac{d}{dt}\dot{R}_{i\to o}\\ &= \frac{d}{dt}\begin{pmatrix}-\sin(\alpha)\ \dot{\alpha} & -\cos(\alpha)\ \dot{\alpha}\\ \cos(\alpha)\ \dot{\alpha} & -\sin(\alpha)\ \dot{\alpha}\end{pmatrix}\\ &= \begin{pmatrix}-\cos(\alpha)\ \dot{\alpha}^2 - \sin(\alpha)\ \ddot{\alpha} & \sin(\alpha)\ \dot{\alpha}^2 - \cos(\alpha)\ \ddot{\alpha}\\ -\sin(\alpha)\ \dot{\alpha}^2 + \cos(\alpha)\ \ddot{\alpha} & -\cos(\alpha)\ \dot{\alpha}^2 - \sin(\alpha)\ \ddot{\alpha}\end{pmatrix}\end{aligned}$$

The transformation of position, velocity and acceleration can be combined into one single matrix multiplication:

$$\begin{aligned}\begin{pmatrix}X_o\\ \dot{X}_o\\ \ddot{X}_o\\ 1\end{pmatrix} &= L_{i\to o}\begin{pmatrix}X_i\\ \dot{X}_i\\ \ddot{X}_i\\ 1\end{pmatrix}\\ &= \begin{pmatrix}R_{i\to o} & 0 & 0 & T\\ \dot{R}_{i\to o} & R_{i\to o} & 0 & \dot{T}\\ \ddot{R}_{i\to o} & 2\dot{R}_{i\to o} & R_{i\to o} & \ddot{T}\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}X_i\\ \dot{X}_i\\ \ddot{X}_i\\ 1\end{pmatrix}\end{aligned}$$

Outer to Inner System

The derivation for the reverse transformation is analogously:

$$\ddot{X}_i = \frac{d}{dt}\left(\dot{R}_{o\to i}\ X_o + R_{o\to i}\ \dot{X}_o - \dot{R}_{o\to i}\ T - R_{o\to i}\ \dot{T}\right) \tag{A.10}$$

$$= \dot{R}_{o\to i}\ \dot{X}_o + \ddot{R}_{o\to i}\ X_o + R_{o\to i}\ \ddot{X}_o + \dot{R}_{o\to i}\ \dot{X}_o - \dot{R}_{o\to i}\ \dot{T} - \ddot{R}_{o\to i}\ T - R_{o\to i}\ \ddot{T} - \dot{R}_{o\to i}\ \dot{T} \tag{A.11}$$

$$= 2\ \dot{R}_{o\to i}\ \dot{X}_o + \ddot{R}_{o\to i}\ X_o + R_{o\to i}\ \ddot{X}_o - 2\ \dot{R}_{o\to i}\ \dot{T} - \ddot{R}_{o\to i}\ T - R_{o\to i}\ \ddot{T} \tag{A.12}$$

with:

$$\begin{aligned}
\ddot{R}_{o\to i} &= \frac{d}{dt}\dot{R}_{o\to i} \\
&= \frac{d}{dt}\begin{pmatrix} -\sin(\alpha)\ \dot{\alpha} & \cos(\alpha)\ \dot{\alpha} \\ -\cos(\alpha)\ \dot{\alpha} & -\sin(\alpha)\ \dot{\alpha} \end{pmatrix} \\
&= \begin{pmatrix} -\cos(\alpha)\ \dot{\alpha}^2 - \sin(\alpha)\ \ddot{\alpha} & -\sin(\alpha)\ \dot{\alpha}^2 + \cos(\alpha)\ \ddot{\alpha} \\ \sin(\alpha)\ \dot{\alpha}^2 - \cos(\alpha)\ \ddot{\alpha} & -\cos(\alpha)\ \dot{\alpha}^2 - \sin(\alpha)\ \ddot{\alpha} \end{pmatrix}
\end{aligned}$$

Written as single matrix multiplication:

$$\begin{aligned}
\begin{pmatrix} X_i \\ \dot{X}_i \\ \ddot{X}_i \\ 1 \end{pmatrix} &= L_{o\to i}\begin{pmatrix} X_o \\ \dot{X}_o \\ \ddot{X}_o \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} R_{o\to i} & 0 & 0 & (-R_{o\to i}\ T) \\ \dot{R}_{o\to i} & R_{o\to i} & 0 & (-\dot{R}_{o\to i}\ T - R_{o\to i}\ \dot{T}) \\ \ddot{R}_{o\to i} & 2\ \dot{R}_{o\to i} & R_{o\to i} & (-2\ \dot{R}_{o\to i}\dot{T} - \ddot{R}_{o\to i}T - R_{o\to i}\ddot{T}) \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} X_o \\ \dot{X}_o \\ \ddot{X}_o \\ 1 \end{pmatrix}
\end{aligned}$$

A.4 Extend to Three Dimensions

Using the formulas for three dimensions extends the position (X), velocity ($\dot{X}$), offset (T) and offset velocity ($\dot{T}$) vector to 3x1 vectors. The rotation matrices ($R_{i\to o}$ and $R_{o\to i}$) will become 3x3 matrices and are a combination of three single rotations (around each of the three coordinate axis):

$$\begin{aligned}
R_{i\to o} &= R^z_{i\to o}\ R^y_{i\to o}\ R^x_{i\to o} \\
&= \begin{pmatrix} \cos(\alpha_z) & -\sin(\alpha_z) & 0 \\ \sin(\alpha_z) & \cos(\alpha_z) & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} \cos(\alpha_y) & 0 & \sin(\alpha_y) \\ 0 & 1 & 0 \\ -\sin(\alpha_y) & 0 & \cos(\alpha_y) \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_x) & -\sin(\alpha_x) \\ 0 & \sin(\alpha_x) & \cos(\alpha_x) \end{pmatrix}
\end{aligned}$$

$$\begin{aligned}
R_{o\to i} &= R^x_{o\to i}\ R^y_{o\to i}\ R^z_{o\to i} \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_x) & \sin(\alpha_x) \\ 0 & -\sin(\alpha_x) & \cos(\alpha_x) \end{pmatrix}\begin{pmatrix} \cos(\alpha_y) & 0 & -\sin(\alpha_y) \\ 0 & 1 & 0 \\ \sin(\alpha_y) & 0 & \cos(\alpha_y) \end{pmatrix}\begin{pmatrix} \cos(\alpha_z) & \sin(\alpha_z) & 0 \\ -\sin(\alpha_z) & \cos(\alpha_z) & 0 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}$$

Derivation of the Rotation Matrices

Inner to Outer

$$\dot{R}_{i\to o} = \dot{R}^z_{i\to o}\; R^y_{i\to o}\; R^x_{i\to o} + R^z_{i\to o}\; \dot{R}^y_{i\to o}\; R^x_{i\to o} + R^z_{i\to o}\; R^y_{i\to o}\; \dot{R}^x_{i\to o}$$

$$\dot{R}^z_{i\to o} = \begin{pmatrix} -\sin(\alpha_z)\;\dot{\alpha_z} & -\cos(\alpha_z)\;\dot{\alpha_z} & 0 \\ \cos(\alpha_z)\;\dot{\alpha_z} & -\sin(\alpha_z)\;\dot{\alpha_z} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\dot{R}^y_{i\to o} = \begin{pmatrix} -\sin(\alpha_y)\;\dot{\alpha_y} & 0 & \cos(\alpha_y)\;\dot{\alpha_y} \\ 0 & 0 & 0 \\ -\cos(\alpha_y)\;\dot{\alpha_y} & 0 & -\sin(\alpha_y)\;\dot{\alpha_y} \end{pmatrix}$$

$$\dot{R}^x_{i\to o} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\sin(\alpha_x)\;\dot{\alpha_x} & -\cos(\alpha_x)\;\dot{\alpha_x} \\ 0 & \cos(\alpha_x)\;\dot{\alpha_x} & -\sin(\alpha_x)\;\dot{\alpha_x} \end{pmatrix}$$

Outer to Inner

$$\dot{R}_{o\to i} = \dot{R}^x_{o\to i}\; R^y_{o\to i}\; R^z_{o\to i} + R^x_{o\to i}\; \dot{R}^y_{o\to i}\; R^z_{o\to i} + R^x_{o\to i}\; R^y_{o\to i}\; \dot{R}^z_{o\to i}$$

$$\dot{R}^x_{o\to i} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\sin(\alpha_x)\;\dot{\alpha_x} & \cos(\alpha_x)\;\dot{\alpha_x} \\ 0 & -\cos(\alpha_x)\;\dot{\alpha_x} & -\sin(\alpha_x)\;\dot{\alpha_x} \end{pmatrix}$$

$$\dot{R}^y_{o\to i} = \begin{pmatrix} -\sin(\alpha_y)\;\dot{\alpha_y} & 0 & -\cos(\alpha_y)\;\dot{\alpha_y} \\ 0 & 0 & 0 \\ \cos(\alpha_y)\;\dot{\alpha_y} & 0 & -\sin(\alpha_y)\;\dot{\alpha_y} \end{pmatrix}$$

$$\dot{R}^z_{o\to i} = \begin{pmatrix} -\sin(\alpha_z)\;\dot{\alpha_z} & \cos(\alpha_z)\;\dot{\alpha_z} & 0 \\ -\cos(\alpha_z)\;\dot{\alpha_z} & -\sin(\alpha_z)\;\dot{\alpha_z} & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

B Motion Model

B.1 General

We define a motion model with **C**onstant **T**urn-**R**ate & (longitudinal) **A**cceleration.

Definitions

- $x(t)$ X-position in meters
- $y(t)$ Y-position in meters
- $\Psi(t)$ Orientation in radiants
- $v(t)$ (Longitudinal) velocity in meters per seconds
- $\dot{\Psi}(t)$ Change of orientation in radiants per seconds
- $\dot{v}(t) = a(t)$ (Longitudinal) acceleration in meters per square-seconds

Start Conditions

$$\begin{aligned} x(0) &= x_0 \\ y(0) &= y_0 \\ \Psi(0) &= \Psi_0 \\ v(0) &= v_0 \end{aligned}$$

Constants

$$\begin{aligned} \dot{\Psi}(t) &= \dot{\Psi} \\ \dot{v}(t) &= a \end{aligned}$$

Differential Equations

$$\begin{aligned} \dot{x}(t) &= \cos\left(\Psi(t)\right) \cdot v(t) \\ \dot{y}(t) &= \sin\left(\Psi(t)\right) \cdot v(t) \\ \dot{\Psi}(t) &= \dot{\Psi} \\ \dot{v}(t) &= a \end{aligned}$$

B.2 Solution

Pose

$$\begin{aligned}
x(t) &= \begin{cases} x_0 + \frac{v(t)\cdot(\sin(\Psi(t))-\sin(\Psi_0))}{\dot\Psi} - \frac{a\left(\frac{\cos(\Psi_0)-\cos(\Psi(t))}{\dot\Psi}-\sin(\Psi_0)\cdot t\right)}{\dot\Psi} & \dot\Psi \neq 0 \\ x_0 + \cos(\Psi)\left(v_0 t + \frac{1}{2}at^2\right) & \dot\Psi = 0 \end{cases} \\
y(t) &= \begin{cases} y_0 + \frac{v(t)\cdot(\cos(\Psi_0)-\cos(\Psi(t)))}{\dot\Psi} - \frac{a\left(\cos(\Psi_0)\cdot t-\frac{\sin(\Psi(t))-\sin(\Psi_0)}{\dot\Psi}\right)}{\dot\Psi} & \dot\Psi \neq 0 \\ y_0 + \sin(\Psi)\left(v_0 t + \frac{1}{2}at^2\right) & \dot\Psi = 0 \end{cases} \\
\Psi(t) &= \Psi_0 + \dot\Psi \cdot t
\end{aligned}$$

Velocity

$$\begin{aligned}
v(t) &= v_0 + a \cdot t \\
\dot\Psi(t) &= \dot\Psi
\end{aligned}$$

Acceleration

$$\dot v(t) = a(t) = a$$

B.3 Partial Derivatives

Pose

$$\begin{aligned}
\frac{\partial x}{\partial x_0} &= 1 \\
\frac{\partial x}{\partial y_0} &= 0 \\
\frac{\partial x}{\partial v_0} &= \begin{cases} \frac{\sin\left(\Psi_0+\dot\Psi_0\cdot t\right)-\sin(\Psi_0)}{\dot\Psi_0} & \dot\Psi_0 \neq 0 \\ \cos\left(\Psi_0\right)\cdot t & \dot\Psi_0 = 0 \end{cases} \\
\frac{\partial x}{\partial \Psi_0} &= \begin{cases} \frac{v_0\left(\cos\left(\Psi_0+\dot\Psi_0\cdot t\right)-\cos(\Psi_0)\right)+a_0\cdot t\cdot\cos\left(\Psi_0+\dot\Psi_0\cdot t\right)}{\dot\Psi_0} + \frac{a_0\left(\sin(\Psi_0)-\sin\left(\Psi_0+\dot\Psi_0\cdot t\right)\right)}{\dot\Psi_0^2} & \dot\Psi_0 \neq 0 \\ -\sin\left(\Psi_0\right)\left(v_0\cdot t + \frac{1}{2}\cdot a_0\cdot t^2\right) & \dot\Psi_0 = 0 \end{cases} \\
\frac{\partial x}{\partial a_0} &= \begin{cases} \frac{t\sin\left(\Psi_0+\dot\Psi_0\cdot t\right)}{\dot\Psi_0} + \frac{\cos\left(\Psi_0+\dot\Psi_0\cdot t\right)-\cos(\Psi_0)}{\dot\Psi_0^2} & \dot\Psi_0 \neq 0 \\ \frac{1}{2}\cos\left(\Psi_0\right)t^2 & \dot\Psi_0 = 0 \end{cases} \\
\frac{\partial x}{\partial \dot\Psi_0} &= \begin{cases} \frac{t(v_0+a_0t)\cos\left(\Psi_0+\dot\Psi_0\cdot t\right)}{\dot\Psi_0} + \frac{v_0\left(\sin(\Psi_0)-\sin\left(\Psi_0+\dot\Psi_0\cdot t\right)\right)-2a_0\cdot t\cdot\sin\left(\Psi_0+\dot\Psi_0\cdot t\right)}{\dot\Psi_0^2} \\ \quad + \frac{2a_0\left(\cos(\Psi_0)-\cos\left(\Psi_0+\dot\Psi_0\cdot t\right)\right)}{\dot\Psi_0^3} & \dot\Psi_0 \neq 0 \\ 0 & \dot\Psi_0 = 0 \end{cases}
\end{aligned}$$

$$\frac{\partial y}{\partial x_0} = 0$$
$$\frac{\partial y}{\partial y_0} = 1$$
$$\frac{\partial y}{\partial v_0} = \begin{cases} \frac{\cos(\Psi_0)-\cos\left(\Psi_0+\dot{\Psi}_0\cdot t\right)}{\dot{\Psi}_0} & \dot{\Psi}_0 \neq 0 \\ \sin(\Psi_0)\cdot t & \dot{\Psi}_0 = 0 \end{cases}$$
$$\frac{\partial y}{\partial \Psi_0} = \begin{cases} \frac{v_0\left(\sin\left(\Psi_0+\dot{\Psi}_0\cdot t\right)-\sin(\Psi_0)\right)+a_0\cdot t\cdot\sin\left(\Psi_0+\dot{\Psi}_0\cdot t\right)}{\dot{\Psi}_0} + \frac{a_0\left(\cos\left(\Psi_0+\dot{\Psi}_0\cdot t\right)-\cos(\Psi_0)\right)}{\dot{\Psi}_0^2} & \dot{\Psi}_0 \neq 0 \\ \cos\left(\Psi_0\right)\left(v_0\cdot t+\frac{1}{2}\cdot a_0\cdot t^2\right) & \dot{\Psi}_0 = 0 \end{cases}$$
$$\frac{\partial y}{\partial a_0} = \begin{cases} \frac{\sin\left(\Psi_0+\dot{\Psi}_0\cdot t\right)-\sin(\Psi_0)}{\dot{\Psi}_0^2} - \frac{t\cdot\cos\left(\Psi_0+\dot{\Psi}_0\cdot t\right)}{\dot{\Psi}_0} & \dot{\Psi}_0 \neq 0 \\ \frac{1}{2}\sin\left(\Psi_0\right)t^2 & \dot{\Psi}_0 = 0 \end{cases}$$
$$\frac{\partial y}{\partial \dot{\Psi}_0} = \begin{cases} \frac{t(v_0+a_0t)\sin\left(\Psi_0+\dot{\Psi}_0\cdot t\right)}{\dot{\Psi}_0} + \frac{v_0\left(\cos\left(\Psi_0+\dot{\Psi}_0\cdot t\right)-\cos(\Psi_0)\right)+2a_0\cdot t\cdot\cos\left(\Psi_0+\dot{\Psi}_0\cdot t\right)}{\dot{\Psi}_0^2} \\ \quad + \frac{2a_0\left(\sin(\Psi_0)-\sin\left(\Psi_0+\dot{\Psi}_0\cdot t\right)\right)}{\dot{\Psi}_0^3} & \dot{\Psi}_0 \neq 0 \\ 0 & \dot{\Psi}_0 = 0 \end{cases}$$

$$\frac{\partial \Psi}{\partial x_0} = 0$$
$$\frac{\partial \Psi}{\partial y_0} = 0$$
$$\frac{\partial \Psi}{\partial v_0} = 0$$
$$\frac{\partial \Psi}{\partial \Psi_0} = 1$$
$$\frac{\partial \Psi}{\partial a_0} = 0$$
$$\frac{\partial \Psi}{\partial \dot{\Psi}_0} = t$$

Velocity

$$\frac{\partial v}{\partial x_0} = 0$$
$$\frac{\partial v}{\partial y_0} = 0$$
$$\frac{\partial v}{\partial v_0} = 1$$
$$\frac{\partial v}{\partial \Psi_0} = 0$$
$$\frac{\partial v}{\partial a_0} = t$$
$$\frac{\partial v}{\partial \dot{\Psi}_0} = 0$$

$$\frac{\partial \dot{\Psi}}{\partial x_0} = 0$$
$$\frac{\partial \dot{\Psi}}{\partial y_0} = 0$$
$$\frac{\partial \dot{\Psi}}{\partial v_0} = 0$$
$$\frac{\partial \dot{\Psi}}{\partial \Psi_0} = 0$$
$$\frac{\partial \dot{\Psi}}{\partial a_0} = 0$$
$$\frac{\partial \dot{\Psi}}{\partial \dot{\Psi}_0} = 1$$

Acceleration

$$
\begin{aligned}
\frac{\partial a}{\partial x_0} &= 0 \\
\frac{\partial a}{\partial y_0} &= 0 \\
\frac{\partial a}{\partial v_0} &= 0 \\
\frac{\partial a}{\partial \Psi_0} &= 0 \\
\frac{\partial a}{\partial a_0} &= 1 \\
\frac{\partial a}{\partial \dot{\Psi}_0} &= 0
\end{aligned}
$$

B.4 Mathematical Derivation

$$
\begin{aligned}
v(t) &= v_0 + \int_0^t \dot{v}(\tau)\ d\tau \\
&= v_0 + \int_0^t a\ d\tau \\
&= v_0 + [a \cdot \tau]_0^t \\
v(t) &= v_0 + a \cdot t
\end{aligned}
$$

$$
\begin{aligned}
\Psi(t) &= \Psi_0 + \int_0^t \dot{\Psi}(\tau)\ d\tau \\
&= \Psi_0 + \left[\dot{\Psi} \cdot \tau\right]_0^t \\
\Psi(t) &= \Psi_0 + \dot{\Psi} \cdot t
\end{aligned}
$$

$$
\begin{aligned}
x(t) &= x_0 + \int_0^t \cos\left(\Psi(\tau)\right) \cdot v(\tau)\ d\tau \\
&= x_0 + \left[\left(\int_0^\tau \cos\left(\Psi(\alpha)\right)\ d\alpha\right) v(\tau)\right]_0^t - \int_0^t \left(\left(\int_0^\tau \cos\left(\Psi(\alpha)\right)\ d\alpha\right) \dot{v}(\tau)\right) d\tau \\
&= x_0 + \left[\frac{v(\tau)}{\dot{\Psi}}\left(\sin\left(\Psi_0 + \dot{\Psi}\tau\right) - \sin\left(\Psi_0\right)\right)\right]_0^t \\
&\quad - \int_0^t \left(\frac{\dot{v}(\tau)}{\dot{\Psi}}\left(\sin\left(\Psi_0 + \dot{\Psi}\tau\right) - \sin\left(\Psi_0\right)\right)\right) d\tau && \text{(Eq. B.3)} \\
&= x_0 + \frac{v_0 + a \cdot t}{\dot{\Psi}}\left(\sin\left(\Psi_0 + \dot{\Psi} \cdot t\right) - \sin\left(\Psi_0\right)\right) \\
&\quad - \int_0^t \left(\frac{\dot{v}(\tau)}{\dot{\Psi}}\left(\sin\left(\Psi_0 + \dot{\Psi}\tau\right) - \sin\left(\Psi_0\right)\right)\right) d\tau \\
&= x_0 + \frac{v_0 + a \cdot t}{\dot{\Psi}}\left(\sin\left(\Psi_0 + \dot{\Psi} \cdot t\right) - \sin\left(\Psi_0\right)\right) \\
&\quad - \frac{a}{\dot{\Psi}}\left(\frac{1}{\dot{\Psi}}\left(\cos\left(\Psi_0\right) - \cos\left(\Psi_0 + \dot{\Psi}t\right)\right) - \sin\left(\Psi_0\right) \cdot t\right) && \text{(Eq. B.5)} \\
&= x_0 + \frac{v(t) \cdot \left(\sin\left(\Psi(t)\right) - \sin\left(\Psi_0\right)\right)}{\dot{\Psi}} - \frac{a\left(\frac{\cos(\Psi_0) - \cos(\Psi(t))}{\dot{\Psi}} - \sin(\Psi_0) \cdot t\right)}{\dot{\Psi}} && \text{(B.1)}
\end{aligned}
$$

$$
\begin{aligned}
y(t) &= y_0 + \int_0^t \sin\left(\Psi(\tau)\right) \cdot v(\tau)\ d\tau \\
&= y_0 + \left[\left(\int_0^\tau \sin\left(\Psi(\alpha)\right)\ d\alpha\right) v(\tau)\right]_0^t - \int_0^t \left(\left(\int_0^\tau \sin\left(\Psi(\alpha)\right)\ d\alpha\right) \dot{v}(\tau)\right) d\tau \\
&= y_0 + \left[\frac{v(\tau)}{\dot{\Psi}}\left(\cos\left(\Psi_0\right) - \cos\left(\Psi_0 + \dot{\Psi}\tau\right)\right)\right]_0^t \\
&\quad - \int_0^t \left(\frac{\dot{v}(\tau)}{\dot{\Psi}}\left(\cos\left(\Psi_0\right) - \cos\left(\Psi_0 + \dot{\Psi}t\right)\right)\right) d\tau && \text{(Eq. B.4)} \\
&= y_0 + \frac{v(t)}{\dot{\Psi}}\left(\cos\left(\Psi_0\right) - \cos\left(\Psi_0 + \dot{\Psi}t\right)\right) \\
&\quad - \int_0^t \left(\frac{\dot{v}(\tau)}{\dot{\Psi}}\left(\cos\left(\Psi_0\right) - \cos\left(\Psi_0 + \dot{\Psi}t\right)\right)\right) d\tau \\
&= y_0 + \frac{v(t)}{\dot{\Psi}}\left(\cos\left(\Psi_0\right) - \cos\left(\Psi_0 + \dot{\Psi}t\right)\right) \\
&\quad - \frac{a}{\dot{\Psi}}\left(\cos\left(\Psi_0\right) \cdot t - \frac{1}{\dot{\Psi}}\left(\sin\left(\Psi_0 + \dot{\Psi}t\right) - \sin\left(\Psi_0\right)\right)\right) && \text{(Eq. B.6)} \\
&= y_0 + \frac{v(t) \cdot \left(\cos\left(\Psi_0\right) - \cos\left(\Psi(t)\right)\right)}{\dot{\Psi}} - \frac{a\left(\cos\left(\Psi_0\right) \cdot t - \frac{\sin(\Psi(t)) - \sin(\Psi_0)}{\dot{\Psi}}\right)}{\dot{\Psi}} && \text{(B.2)}
\end{aligned}
$$

B.5 Auxiliary Calculations

$$\begin{aligned}
\int_0^t \cos\left(\Psi(\tau)\right)\ d\tau &= \int_0^t \cos\left(\Psi_0 + \dot{\Psi}\cdot\tau\right)\ d\tau \\
&= \frac{1}{\dot{\Psi}}\int_0^t \cos\left(\Psi_0 + \dot{\Psi}\cdot\tau\right)\dot{\Psi}\ d\tau \\
&= \frac{1}{\dot{\Psi}}\int_{\Psi_0}^{\Psi_0+\dot{\Psi}\tau} \cos x\ dx && \text{(Integration by substitution)} \\
&= \frac{1}{\dot{\Psi}}\left[\sin x\right]_{\Psi_0}^{\Psi_0+\dot{\Psi}\tau} \\
\int_0^t \cos\left(\Psi(\tau)\right)\ d\tau &= \frac{1}{\dot{\Psi}}\left(\sin\left(\Psi_0+\dot{\Psi}t\right) - \sin\left(\Psi_0\right)\right) && \text{(B.3)}
\end{aligned}$$

$$\begin{aligned}
\int_0^t \sin\left(\Psi(\tau)\right)\ d\tau &= \int_0^t \sin\left(\Psi_0 + \dot{\Psi}\cdot\tau\right)\ d\tau \\
&= \frac{1}{\dot{\Psi}}\int_0^t \sin\left(\Psi_0 + \dot{\Psi}\cdot\tau\right)\dot{\Psi}\ d\tau \\
&= \frac{1}{\dot{\Psi}}\int_{\Psi_0}^{\Psi_0+\dot{\Psi}\tau} \sin x\ dx && \text{(Integration by substitution)} \\
&= \frac{1}{\dot{\Psi}}\left[-\cos x\right]_{\Psi_0}^{\Psi_0+\dot{\Psi}\tau} \\
\int_0^t \sin\left(\Psi(\tau)\right)\ d\tau &= \frac{1}{\dot{\Psi}}\left(\cos\left(\Psi_0\right) - \cos\left(\Psi_0+\dot{\Psi}t\right)\right) && \text{(B.4)}
\end{aligned}$$

$$\begin{aligned}
&\int_0^t \left(\frac{\dot{v}(\tau)}{\dot{\Psi}}\left(\sin\left(\Psi_0+\dot{\Psi}\tau\right) - \sin\left(\Psi_0\right)\right)\right)d\tau = \\
&= \frac{a}{\dot{\Psi}}\int_0^t \left(\sin\left(\Psi_0+\dot{\Psi}\tau\right) - \sin\left(\Psi_0\right)\right)d\tau \\
&= \frac{a}{\dot{\Psi}}\left(\int_0^t \sin\left(\Psi_0+\dot{\Psi}\tau\right)d\tau - \int_0^t \sin\left(\Psi_0\right)d\tau\right) \\
&= \frac{a}{\dot{\Psi}}\left(\int_0^t \sin\left(\Psi_0+\dot{\Psi}\tau\right)d\tau - \sin\left(\Psi_0\right)\cdot t\right) \\
&= \frac{a}{\dot{\Psi}}\left(\frac{1}{\dot{\Psi}}\left(\cos\left(\Psi_0\right) - \cos\left(\Psi_0+\dot{\Psi}t\right)\right) - \sin\left(\Psi_0\right)\cdot t\right) && \text{(Insert eq. B.4)} \quad \text{(B.5)}
\end{aligned}$$

$$\begin{aligned}
&\int_0^t \left(\frac{\dot{v}(\tau)}{\dot{\Psi}} \left(\cos\left(\Psi_0\right) - \cos\left(\Psi_0 + \dot{\Psi}\tau\right)\right)\right) d\tau = \\
&= \frac{a}{\dot{\Psi}} \left(\int_0^t \cos\left(\Psi_0\right) d\tau - \int_0^t \cos\left(\Psi_0 + \dot{\Psi}\tau\right) d\tau \right) \\
&= \frac{a}{\dot{\Psi}} \left(\cos\left(\Psi_0\right) \cdot t - \int_0^t \cos\left(\Psi_0 + \dot{\Psi}\tau\right) d\tau \right) \\
&= \frac{a}{\dot{\Psi}} \left(\cos\left(\Psi_0\right) \cdot t - \frac{1}{\dot{\Psi}} \left(\sin\left(\Psi_0 + \dot{\Psi}t\right) - \sin\left(\Psi_0\right)\right)\right) \qquad \text{(Insert eq. B.3)} \qquad \text{(B.6)}
\end{aligned}$$

C Evaluation Results

C.1 Grid Evaluation

Range	*F*	*S*	*D*
5m	1 579 225	37 696	8 421
10m	5 231 105	195 351	32 791
15m	10 642 021	402 950	91 455
20m	17 049 042	662 217	182 239
30m	31 512 866	1 262 973	390 591
40m	47 410 914	1 670 790	571 799
60m	83 021 330	2 258 569	781 428
90m	100 716 903	2 362 658	861 507

Table C.1: Maximum score values over different ranges. All 10 scenes summed up. See figure 6.6.

Range	Lidar		Radar			Combined		
	F	*SD*	*F*	*D*	*SD*	*F*	*D*	*SD*
5m	12.7	0.2	0.0	0.0	0.0	12.7	0.0	0.2
10m	24.2	0.3	0.2	0.0	0.0	24.4	0.0	0.3
15m	20.5	0.4	1.9	0.0	0.0	22.1	0.0	0.4
20m	15.9	0.3	3.0	0.0	0.0	18.6	0.0	0.4
30m	9.7	0.3	3.4	0.0	0.1	12.9	0.0	0.3
40m	6.6	0.2	3.2	0.1	0.1	9.6	0.1	0.3
60m	3.8	0.1	2.3	0.1	0.1	6.0	0.1	0.2
90m	3.1	0.1	2.0	0.1	0.1	5.0	0.1	0.2

Table C.2: Detection scores in percentage for freespace of lidar, radar and combined sensor grid. All 10 scenes summed up. See figure 6.7.

Range	Lidar		Radar			Combined		
	F	SD	F	D	SD	F	D	SD
5m	2.1	12.9	0.0	0.0	0.0	2.1	0.0	12.9
10m	0.8	10.9	0.0	0.0	0.0	0.8	0.0	10.9
15m	0.5	8.3	0.9	0.0	0.4	1.4	0.0	8.6
20m	0.4	6.2	1.0	0.1	0.5	1.3	0.1	6.7
30m	0.2	4.0	0.8	0.1	0.6	1.0	0.1	4.5
40m	0.2	3.1	0.8	0.1	0.6	0.9	0.1	3.7
60m	0.1	2.4	0.7	0.1	0.7	0.8	0.1	3.1
90m	0.1	2.3	0.6	0.1	0.7	0.7	0.1	2.9

Table C.3: Detection scores in percentage for static occupancy of lidar, radar and combined sensor grid. All 10 scenes summed up. See figure 6.10.

Range	Lidar		Radar			Combined		
	F	SD	F	D	SD	F	D	SD
5m	1.7	12.9	0.0	0.0	0.0	1.7	0.0	12.9
10m	0.7	14.6	0.1	0.0	0.0	0.8	0.0	14.6
15m	0.5	9.6	1.4	2.3	0.0	1.7	2.3	9.4
20m	0.4	6.8	1.7	3.2	0.1	2.0	3.2	6.6
30m	0.3	4.3	1.3	3.9	0.2	1.5	3.9	4.3
40m	0.2	3.1	1.1	4.1	0.1	1.2	4.1	3.1
60m	0.1	2.4	0.9	4.1	0.1	1.0	4.1	2.3
90m	0.1	2.1	0.8	4.1	0.1	0.9	4.1	2.1

Table C.4: Detection scores in percentage for dynamic occupancy of lidar, radar and combined sensor grid. All 10 scenes summed up. See figure 6.13.

Sensor	Front			Rear-right			Rear-left			Combined		
Error $[\frac{m}{s}]$	1	2	4	1	2	4	1	2	4	1	2	4
5m	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10m	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15m	0.1	0.1	0.1	0.4	0.6	0.8	0.4	0.5	0.9	0.8	1.1	1.5
20m	0.2	0.3	0.4	0.5	0.7	1.0	0.5	0.8	1.2	1.2	1.7	2.2
30m	0.6	0.8	0.9	0.7	0.9	1.0	0.7	0.9	1.2	1.8	2.3	2.8
40m	0.7	0.9	1.1	0.8	1.1	1.3	0.9	1.1	1.4	2.2	2.7	3.4
60m	0.8	1.1	1.4	0.8	1.0	1.3	0.8	1.1	1.4	2.2	2.9	3.5
90m	0.9	1.2	1.5	0.8	1.0	1.2	0.8	1.0	1.4	2.3	2.9	3.6

Table C.5: Percentage of cells with an estimated velocity within a given error compared to the reference value. Only cells with a reference velocity are considered. This table shows the result of the three single radar sensors, as well as the combined sensor grid. See figure 6.15.

Range	Sensor grid			Dynamic grid				
	F	D	SD	F	S	D	FD	SD
5m	12.7	0.0	0.2	12.7	0.3	1.2	50.1	0.2
10m	24.4	0.0	0.3	24.4	0.3	1.5	53.3	0.3
15m	22.1	0.0	0.4	22.1	0.5	1.6	54.7	0.4
20m	18.6	0.0	0.4	18.6	0.4	1.5	54.0	0.5
30m	12.9	0.0	0.3	12.9	0.4	1.3	47.4	0.7
40m	9.6	0.1	0.3	9.6	0.4	1.2	40.5	0.8
60m	6.0	0.1	0.2	6.0	0.3	0.9	29.5	1.0
90m	5.0	0.1	0.2	5.0	0.3	0.8	25.6	1.0

Table C.6: Detection scores in percentage for freespace of the input sensor grid and the dynamic grid. All 10 scenes summed up. See figure 6.16.

Range	Sensor grid			Dynamic grid				
	F	*D*	*SD*	*F*	*S*	*D*	*FD*	*SD*
5m	2.1	0.0	12.9	2.1	19.5	7.6	13.2	10.9
10m	0.8	0.0	10.9	0.7	14.5	11.3	5.8	11.8
15m	1.4	0.0	8.6	1.3	11.7	10.6	10.2	11.4
20m	1.3	0.1	6.7	1.3	9.2	9.3	10.5	10.4
30m	1.0	0.1	4.5	1.0	6.2	7.0	9.2	8.5
40m	0.9	0.1	3.7	0.9	5.2	6.1	8.9	8.0
60m	0.8	0.1	3.1	0.8	4.3	5.0	8.1	7.6
90m	0.7	0.1	2.9	0.7	4.1	4.8	7.8	7.4

Table C.7: Detection scores in percentage for static occupancy of the input sensor grid and the dynamic grid. All 10 scenes summed up. See figure 6.17.

Range	Sensor grid			Dynamic grid				
	F	*D*	*SD*	*F*	*S*	*D*	*FD*	*SD*
5m	1.7	0.0	12.9	1.7	1.8	32.7	21.8	6.6
10m	0.8	0.0	14.6	0.8	2.4	34.9	20.9	8.4
15m	1.7	2.3	9.4	1.7	1.7	29.5	28.9	6.3
20m	2.0	3.2	6.6	1.9	1.1	25.6	31.8	4.8
30m	1.5	3.9	4.3	1.5	0.7	19.5	29.2	3.6
40m	1.2	4.1	3.1	1.2	0.6	17.0	26.2	3.1
60m	1.0	4.1	2.3	1.0	0.4	14.6	21.9	2.6
90m	0.9	4.1	2.1	0.9	0.4	14.0	20.4	2.4

Table C.8: Detection scores in percentage for dynamic occupancy of the input sensor grid and the dynamic grid. All 10 scenes summed up. See figure 6.18.

	Sensor grid			Dynamic grid		
Error $[\frac{m}{s}]$	1	2	4	1	2	4
5m	0.0	0.0	0.0	29.2	36.3	46.5
10m	0.0	0.0	0.0	19.1	27.2	39.6
15m	0.8	1.1	1.5	15.2	23.1	35.4
20m	1.2	1.7	2.2	12.8	19.8	31.0
30m	1.8	2.3	2.8	9.6	15.0	23.9
40m	2.2	2.7	3.4	8.5	13.2	21.0
60m	2.2	2.9	3.5	7.2	11.1	17.7
90m	2.3	2.9	3.6	6.9	10.6	16.9

Table C.9: Percentage of cells with an estimated velocity within a given error compared to the reference value. Only cells with a reference velocity are considered. This table shows the result of the input sensor grid and the dynamic grid. See figure 6.19.

Range	With orientation prior					Without orientation prior				
	F	*S*	*D*	*FD*	*SD*	*F*	*S*	*D*	*FD*	*SD*
5m	12.7	0.3	1.2	50.1	0.2	12.7	0.3	1.3	50.1	0.2
10m	24.4	0.3	1.5	53.3	0.3	24.4	0.4	1.6	53.2	0.3
15m	22.1	0.5	1.6	54.7	0.4	22.1	0.5	1.6	54.7	0.4
20m	18.6	0.4	1.5	54.0	0.5	18.6	0.5	1.5	53.9	0.5
30m	12.9	0.4	1.3	47.4	0.7	12.9	0.4	1.4	47.4	0.7
40m	9.6	0.4	1.2	40.5	0.8	9.6	0.4	1.2	40.5	0.8
60m	6.0	0.3	0.9	29.5	1.0	6.0	0.3	0.9	29.5	1.0
90m	5.0	0.3	0.8	25.6	1.0	5.0	0.3	0.8	25.6	1.0

Table C.10: Detection scores in percentage for freespace of the dynamic grid, with and without using the online orientation prior. All 10 scenes summed up. See figure 6.20.

Range	With orientation prior					Without orientation prior				
	F	*S*	*D*	*FD*	*SD*	*F*	*S*	*D*	*FD*	*SD*
5m	2.1	19.5	7.6	13.2	10.9	2.1	18.9	8.4	13.2	9.9
10m	0.7	14.5	11.3	5.8	11.8	0.7	13.3	12.9	5.7	10.4
15m	1.3	11.7	10.6	10.2	11.4	1.3	10.6	12.1	10.2	10.0
20m	1.3	9.2	9.3	10.5	10.4	1.3	8.3	10.6	10.4	9.0
30m	1.0	6.2	7.0	9.2	8.5	1.0	5.6	8.0	9.1	7.4
40m	0.9	5.2	6.1	8.9	8.0	0.9	4.7	6.9	8.8	7.0
60m	0.8	4.3	5.0	8.1	7.6	0.8	3.8	5.7	8.0	6.7
90m	0.7	4.1	4.8	7.8	7.4	0.7	3.7	5.5	7.8	6.6

Table C.11: Detection scores in percentage for static occupancy of the dynamic grid, with and without using the online orientation prior. All 10 scenes summed up. See figure 6.21.

Range	With orientation prior					Without orientation prior				
	F	*S*	*D*	*FD*	*SD*	*F*	*S*	*D*	*FD*	*SD*
5m	1.7	1.8	32.7	21.8	6.6	1.7	3.2	25.0	27.2	7.6
10m	0.8	2.4	34.9	20.9	8.4	0.8	3.7	26.3	26.0	10.3
15m	1.7	1.7	29.5	28.9	6.3	1.7	2.5	23.5	32.7	7.3
20m	1.9	1.1	25.6	31.8	4.8	1.9	1.6	21.2	34.5	5.6
30m	1.5	0.7	19.5	29.2	3.6	1.5	1.0	16.7	30.8	4.1
40m	1.2	0.6	17.0	26.2	3.1	1.2	0.8	14.9	27.6	3.4
60m	1.0	0.4	14.6	21.9	2.6	1.0	0.6	12.9	23.0	2.8
90m	0.9	0.4	14.0	20.4	2.4	0.9	0.5	12.4	21.4	2.6

Table C.12: Detection scores in percentage for dynamic occupancy of the dynamic grid, with and without using the online orientation prior. All 10 scenes summed up. See figure 6.22.

Range	With prior			Without prior		
Error $[\frac{m}{s}]$	1	2	4	1	2	4
5m	29.2	36.3	46.5	28.3	37.3	49.2
10m	19.1	27.2	39.6	18.9	29.5	46.0
15m	15.2	23.1	35.4	14.3	24.1	40.0
20m	12.8	19.8	31.0	11.5	19.8	33.9
30m	9.6	15.0	23.9	8.3	14.4	25.1
40m	8.5	13.2	21.0	7.2	12.4	21.7
60m	7.2	11.1	17.7	6.1	10.4	18.1
90m	6.9	10.6	16.9	5.8	9.9	17.3

Table C.13: Percentage of cells with an estimated velocity within a given error compared to the reference value. Only cells with a reference velocity are considered. This table shows the result of the dynamic grid, with and without using the online orientation prior. See figure 6.23.

C.2 Objects Evaluation

C.2.1 Object Extractions

	Without orientation prior						With orientation prior					
Range	TP	FN	FP	Sen.	Pre.	F_1	TP	FN	FP	Sen.	Pre.	F_1
5m	10	42	1	19 %	91 %	32 %	16	36	16	31 %	50 %	38 %
10m	35	85	12	29 %	74 %	42 %	60	60	56	50 %	52 %	51 %
15m	80	172	20	32 %	80 %	45 %	160	92	103	63 %	61 %	62 %
20m	145	278	44	34 %	77 %	47 %	288	135	172	68 %	63 %	65 %
30m	323	543	141	37 %	70 %	49 %	537	329	413	62 %	57 %	59 %
40m	490	737	251	40 %	66 %	50 %	718	509	674	59 %	52 %	55 %
50m	560	871	368	39 %	60 %	47 %	794	637	868	55 %	48 %	51 %

Table C.14: True positives, false negatives, false positives, sensivity, precision and F_1 score over different ranges. All 10 scenes summed up. See figures 6.24 and 6.25.

	Without orientation prior				With orientation prior				
Range	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	P_q
5m	0.13	0.14	0.18	0.24	0.25	0.54	0.75	0.64	0.38
10m	0.12	0.18	0.31	0.26	0.18	0.26	0.55	0.42	0.20
15m	0.13	0.27	0.42	0.33	0.17	0.25	0.41	0.36	0.17
20m	0.16	0.29	0.45	0.35	0.17	0.27	0.43	0.37	0.17
30m	0.22	0.33	0.55	0.42	0.20	0.32	0.47	0.42	0.23
40m	0.22	0.35	0.58	0.43	0.21	0.34	0.50	0.43	0.25
50m	0.23	0.37	0.64	0.45	0.22	0.35	0.53	0.45	0.27

Table C.15: Position error [m] of extracted objects with and without using the online generated orientation prior over different ranges. All 10 scenes summed up. See figure 6.26.

	Without orientation prior				With orientation prior				
Range	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	P_q
5m	6.37	14.18	18.73	12.82	1.33	2.74	4.00	3.64	1.54
10m	4.63	10.25	15.57	10.84	1.65	2.93	8.63	6.80	2.02
15m	4.53	9.37	16.42	12.03	1.59	2.95	7.78	6.51	1.59
20m	2.51	6.37	16.89	12.07	1.63	3.39	8.17	7.01	1.64
30m	2.79	8.30	18.59	12.83	1.50	3.11	7.61	6.79	1.78
40m	2.95	7.85	18.06	12.61	1.46	3.12	7.89	6.97	1.95
50m	3.21	8.68	18.44	12.98	1.51	3.34	8.23	7.29	2.14

Table C.16: Orientation error [deg] of extracted objects with and without using the online generated orientation prior over different ranges. All 10 scenes summed up. See figure 6.27.

	Without orientation prior				With orientation prior				
Range	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	P_q
5m	5.02	7.64	8.27	6.50	1.84	2.04	4.78	3.01	1.94
10m	1.63	4.28	8.00	4.70	1.84	2.73	4.80	3.19	1.92
15m	0.93	2.91	7.21	3.94	2.02	3.40	4.79	3.51	2.02
20m	0.77	1.92	4.32	2.92	1.43	2.78	4.41	3.00	1.43
30m	0.69	1.46	3.22	2.34	0.96	2.01	3.69	2.46	1.17
40m	0.65	1.53	3.20	2.25	0.76	1.76	3.28	2.23	1.11
50m	0.67	1.68	3.24	2.32	0.79	1.74	3.21	2.22	1.18

Table C.17: Speed error [m/s] of extracted objects with and without using the online generated orientation prior over different ranges. All 10 scenes summed up. See figure 6.28.

	Without orientation prior				With orientation prior				
Range	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	P_q
5m	0.32	0.56	0.92	0.66	0.16	0.20	0.81	1.30	0.17
10m	0.37	0.70	1.12	0.99	0.16	0.26	0.60	0.75	0.18
15m	0.51	1.26	2.99	1.80	0.22	0.54	1.45	1.12	0.22
20m	0.62	1.66	3.31	2.13	0.27	0.74	2.34	1.45	0.27
30m	0.95	2.83	3.70	2.55	0.45	1.47	3.12	1.91	0.56
40m	1.88	3.09	3.74	2.81	0.62	2.37	3.22	2.19	1.04
50m	2.08	3.18	3.81	2.91	0.68	2.47	3.34	2.27	1.38

Table C.18: Length error [m] of extracted objects with and without using the online generated orientation prior over different ranges. All 10 scenes summed up. See figure 6.29.

	Without orientation prior				With orientation prior				
Range	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	P_q
5m	0.15	0.23	0.31	0.29	0.11	0.21	0.48	0.38	0.13
10m	0.09	0.20	0.37	0.33	0.09	0.20	0.47	0.30	0.10
15m	0.16	0.33	0.94	0.56	0.12	0.28	0.81	0.48	0.12
20m	0.22	0.59	1.08	0.68	0.15	0.46	1.05	0.62	0.15
30m	0.34	0.86	1.25	0.85	0.22	0.76	1.17	0.75	0.28
40m	0.34	0.82	1.19	0.82	0.23	0.70	1.15	0.73	0.36
50m	0.34	0.83	1.22	0.82	0.23	0.68	1.15	0.73	0.38

Table C.19: Width error [m] of extracted objects with and without using the online generated orientation prior over different ranges. All 10 scenes summed up. See figure 6.30.

C.2.2 Tracked Objects

	Extracted objects						Tracked objects					
Range	TP	FN	FP	Sen.	Pre.	F_1	TP	FN	FP	Sen.	Pre.	F_1
5m	16	36	16	31 %	50 %	38 %	50	2	10	96 %	83 %	89 %
10m	60	60	56	50 %	52 %	51 %	98	22	41	82 %	71 %	76 %
15m	160	92	103	63 %	61 %	62 %	206	46	62	82 %	77 %	79 %
20m	288	135	172	68 %	63 %	65 %	330	93	99	78 %	77 %	77 %
30m	537	329	413	62 %	57 %	59 %	599	267	231	69 %	72 %	71 %
40m	718	509	674	59 %	52 %	55 %	792	435	363	65 %	69 %	66 %
50m	794	637	868	55 %	48 %	51 %	868	563	481	61 %	64 %	62 %

Table C.20: True positives, false negatives and false positives over different ranges. All 10 scenes summed up. See figures 6.31 and 6.32.

	Extracted objects				Tracked objects				
Range	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	P_q
5m	0.25	0.54	0.75	0.64	0.22	0.39	1.22	1.31	0.17
10m	0.18	0.26	0.55	0.42	0.24	0.42	0.75	0.58	0.27
15m	0.17	0.25	0.41	0.36	0.24	0.41	0.75	0.61	0.32
20m	0.17	0.27	0.43	0.37	0.25	0.43	0.78	0.60	0.37
30m	0.20	0.32	0.47	0.42	0.30	0.49	0.84	0.66	0.45
40m	0.21	0.34	0.50	0.43	0.28	0.48	0.86	0.67	0.44
50m	0.22	0.35	0.53	0.45	0.29	0.50	0.88	0.69	0.46

Table C.21: Position error [m] of tracked objects over different ranges. All 10 scenes summed up. See figure 6.33.

	Extracted objects				Tracked objects				
Range	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	P_q
5m	1.33	2.74	4.00	3.64	2.98	11.67	16.61	10.86	1.78
10m	1.65	2.93	8.63	6.80	2.06	6.09	16.36	9.42	2.51
15m	1.59	2.95	7.78	6.51	1.69	4.11	13.87	7.82	2.70
20m	1.63	3.39	8.17	7.01	1.66	4.27	10.93	7.82	3.20
30m	1.50	3.11	7.61	6.79	1.36	3.24	9.00	6.86	2.76
40m	1.46	3.12	7.89	6.97	1.33	3.20	8.47	6.75	2.75
50m	1.51	3.34	8.23	7.29	1.31	3.21	8.16	6.67	2.82

Table C.22: Orientation error [deg] of tracked objects over different ranges. All 10 scenes summed up. See figure 6.34.

	Extracted objects				Tracked objects				
Range	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	P_q
5m	1.84	2.04	4.78	3.01	0.05	0.31	0.80	0.65	0.04
10m	1.84	2.73	4.80	3.19	0.24	1.20	2.12	1.28	0.41
15m	2.02	3.40	4.79	3.51	0.75	1.67	2.32	1.61	1.37
20m	1.43	2.78	4.41	3.00	0.62	1.44	2.22	1.50	1.24
30m	0.96	2.01	3.69	2.46	0.46	1.13	1.99	1.33	0.97
40m	0.76	1.76	3.28	2.23	0.40	0.93	1.84	1.22	0.83
50m	0.79	1.74	3.21	2.22	0.39	0.93	1.83	1.21	0.85

Table C.23: Speed error [m/s] of tracked objects over different ranges. All 10 scenes summed up. See figure 6.35.

	Extracted objects				Tracked objects				
Range	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	P_q
5m	0.16	0.20	0.81	1.30	0.36	0.56	4.13	1.82	0.32
10m	0.16	0.26	0.60	0.75	0.35	0.66	1.80	1.41	0.39
15m	0.22	0.54	1.45	1.12	0.35	0.79	1.43	1.24	0.55
20m	0.27	0.74	2.34	1.45	0.35	0.80	1.61	1.30	0.66
30m	0.45	1.47	3.12	1.91	0.38	0.86	2.11	1.39	0.79
40m	0.62	2.37	3.22	2.19	0.42	1.04	2.35	1.51	0.89
50m	0.68	2.47	3.34	2.27	0.42	1.02	2.38	1.51	0.86

Table C.24: Length error [m] of tracked objects over different ranges. All 10 scenes summed up. See figure 6.36.

	Extracted objects				Tracked objects				
Range	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	$P_{25\%}$	$P_{50\%}$	$P_{75\%}$	Mean	P_q
5m	0.11	0.21	0.48	0.38	0.16	0.58	0.62	0.41	0.09
10m	0.09	0.20	0.47	0.30	0.21	0.57	0.62	0.47	0.34
15m	0.12	0.28	0.81	0.48	0.25	0.51	0.63	0.47	0.34
20m	0.15	0.46	1.05	0.62	0.21	0.42	0.64	0.48	0.34
30m	0.22	0.76	1.17	0.75	0.24	0.53	0.68	0.52	0.43
40m	0.23	0.70	1.15	0.73	0.21	0.47	0.67	0.50	0.42
50m	0.23	0.68	1.15	0.73	0.20	0.46	0.67	0.50	0.42

Table C.25: Width error [m] of tracked objects over different ranges. All 10 scenes summed up. See figure 6.37.