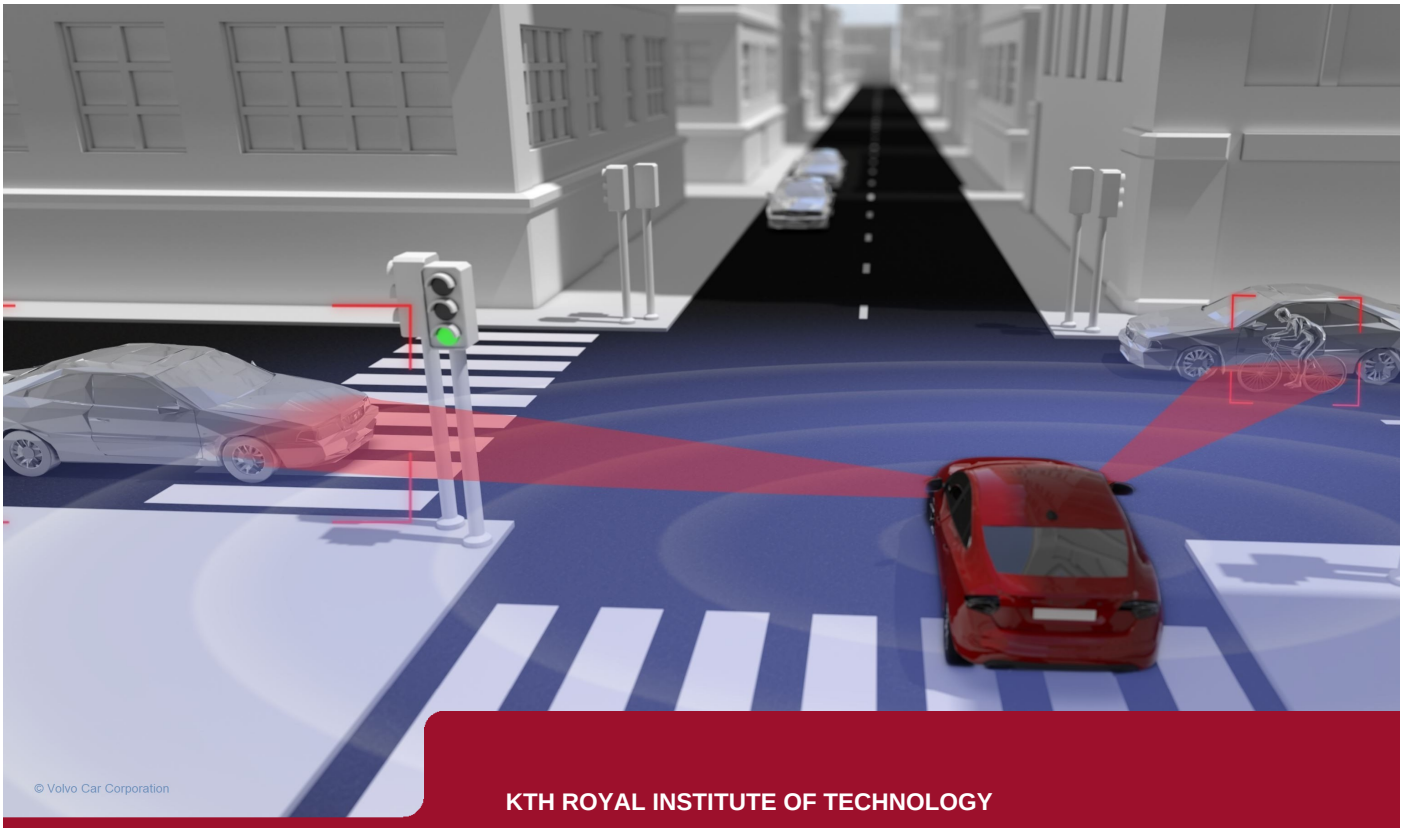
DEGREE PROJECT, IN COMPUTER SCIENCE , SECOND LEVEL
STOCKHOLM, *SWEDEN 2015*

Grid-Based Multi-Sensor Fusion for On-Road Obstacle Detection: Application to Autonomous Driving

CARLOS GÁLVEZ DEL POSTIGO FERNÁNDEZ



KTH ROYAL INSTITUTE OF TECHNOLOGY

SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION

Grid-Based Multi-Sensor Fusion for On-Road Obstacle Detection: Application to Autonomous Driving

—

Rutnätsbaserad multisensorfusion för detektering av hinder på vägen: tillämpning på självkörande bilar

CARLOS GÁLVEZ DEL POSTIGO FERNÁNDEZ
cgdpf@kth.se

Master's Thesis at KTH-CVAP and Volvo Car Corporation

Supervisor at KTH:	John Folkesson
Supervisor at Volvo:	Daniel Svensson
Examiner:	Patric Jensfelt

In partial fulfillment of the requirements for the degree of Master of Science in Systems, Control and Robotics

Stockholm, August 2015

School of Computer Science and Communication
KTH Royal Institute of Technology

To my parents, for their unconditional love and support

"Challenges make life interesting.
Overcoming them makes it meaningful."

Abstract

Self-driving cars have recently become a challenging research topic, with the aim of making transportation safer and more efficient. Current advanced driving assistance systems (ADAS) allow cars to drive autonomously by following lane markings, identifying road signs and detecting pedestrians and other vehicles. In this thesis work we improve the robustness of autonomous cars by designing an on-road obstacle detection system.

The proposed solution consists on the low-level fusion of radar and lidar through the occupancy grid framework. Two inference theories are implemented and evaluated: Bayesian probability theory and Dempster-Shafer theory of evidence. Obstacle detection is performed through image processing of the occupancy grid. Last, the Dempster-Shafer additional features are leveraged by proposing a sensor performance estimation module and performing advanced conflict management.

The work has been carried out at Volvo Car Corporation, where real experiments on a test vehicle have been performed under different environmental conditions and types of objects. The system has been evaluated according to the quality of the resulting occupancy grids, detection rate as well as information content in terms of entropy. The results show a significant improvement of the detection rate over single-sensor approaches. Furthermore, the Dempster-Shafer implementation may slightly outperform the Bayesian one when there is conflicting information, although the high computational cost limits its practical application. Last, we demonstrate that the proposed solution is easily scalable to include additional sensors.

Keywords — autonomous driving, occupancy grid mapping, sensor fusion, Dempster-Shafer, obstacle detection

Sammanfattning

Självkörande bilar är ett forskningsområde med växande intresse, vilket syftar till att göra transporter säkrare och effektivare. Nuvarande förarstödssystem tillåter bilar att köra självständigt genom att följa körfältsmarkeringar, identifiera vägmärken och upptäcka fotgängare och andra fordon. I detta examensarbete förbättra vi autonoma bilars robusthet genom att utforma ett detektionssystem för hinder på vägen.

Den föreslagna lösningen använder sig av lågnivåfusion av radar och lidar inom ett rutnätsbaserat ramverk kallat beläggningsnät, eller "occupancy grid". Två inferensmetoder för att uppdatera beläggningsnätet genomförs och utvärderas: Bayesiansk sannolikhetsteori och Dempster-Shafer-teori. Objektdetektering sker genom bildbehandling av rutnätet. Utöver detta föreslås inom ramen för Dempster-Shafer-teorin en ytterligare funktion vilken innefattar sensorprestandauppskattning samt avancerad konflikthantering av sensordata.

Arbetet har utförts vid Volvo Personvagnar, där verkliga experiment med ett provfordon har utförts under olika väderförhållanden och för olika typer av objekt. Systemet har utvärderats med avseende på kvaliteten på beläggningsnäten, detekteringssannolikhet samt informationsinnehåll i form av entropi. Resultaten visar en signifikant förbättring av detekteringssannolikheten jämfört med ensensormetoder. Vidare ges visst stöd för att Dempster-Shafer-metoden kan ge viss prestandaförbättring jämfört med den Bayesianska metoden vid motstridiga sensordata, även om den höga beräkningskostnaden begränsar dess praktiska tillämpning. Slutligen visar vi att den föreslagna lösningen är skalbar till att inkludera ytterligare sensorer.

Nyckelord — självkörande bilar, rutnätskartering, sensorfusion, Dempster-Shafer, hinderdetektering

Acknowledgments

I would like start this report with a few words expressing my most sincere gratitude to a number of people who in some way were part of this memorable period of my life as a student and contributed to the successful completion of my studies.

First, to Volvo Cars, for giving me the opportunity to carry out my Master's Thesis with them. This was possible thanks to Daniel Svensson, my supervisor, and Tove Hellgård, manager of the Sensor Fusion team, a group of really nice and comitted people I had the pleasure to work with.

Furthermore, to KTH, for the enriching international experience during these last two years. Especially to Patric Jensfelt, director of the Systems, Control and Robotics Master programme, for the valuable help throughout the whole master, starting with the intricate admission process and ending with my thesis examination. For giving me lots of opportunities to learn, make mistakes and challenge myself to find out what I am capable of. Furthermore, to John Folkesson, my thesis supervisor, for his advice and support during this project. Finally, to all the friends and classmates I met during these two years, who contributed to this wonderful international experience.

I am also grateful to my home university, ETSIT-UPM, where I started my student life within Telecommunication Engineering. Thanks for four years of excellent education and for making the international exchange possible. To all my friends at school, for all the good and bad moments we shared during these last years. To the people of the IEEE student branch, for being an infinite source of knowledge, as well as for all the amusing activities and crazy parties. Finally, I would also like to thank my friends from the Gómez Pardo student residence, for four years full of memorable experiences.

Last but not least, I owe a few words to my family, especially to my parents Antonio and Angelina and my brother Manuel. For inspiring me in life, for giving me everything in exchange of nothing, and for loving me no matter what. For letting me pursue my dreams despite leaving home to go far away. Finally, to those who sadly cannot be here today but would still be very proud of me.

Contents

List of Figures

List of Tables

Nomenclature

ACC	Adaptive Cruise Control
ADAS	Advance Driver Assistance Systems
BOF	Bayesian Occupancy Filter
BPA	Basic Probability Assignment
CAN	Controller Area Network
CUDA	Compute Unified Device Architecture
DoF	Degrees of Freedom
EM	Expectation-Maximization
FCTA	Fast Clustering-Tracking Algorithm
FMCW	Frequency Modulated Continuous Wave
FOD	Frame of Discernment
GPS	Global Positioning System
GPU	Graphics Processing Unit
HMI	Human-Machine Interaction
HMM	Hidden Markov Model
HSBOF	Hybrid Sampling Bayesian Occupancy Filter
IMM	Interacting Multiple Models
IOP	Independent Opinion Pool
JPDA	Joint Probabilistic Data Association
Lidar	Laser Imaging Detection and Ranging
LIOP	Logarithmic Independent Opinion Pool
LOP	Linear Opinion Pool

MAP	Maximum A Posteriori
MCMC	Markov Chain Monte Carlo
MHT	Multiple Hypothesis Tracking
MoG	Mixture of Gaussians
MURIEL	MUltiple Representation, Independence Evidence Log
Radar	Radio Detection and Ranging
RANSAC	RANdom SAmple Consensus
RCS	Radar Cross Section
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization And Mapping
SON	Self-Organizing Network
STAR-PD	Simultaneous Transmit-Receive Pulse Doppler
TBM	Transferable Belief Model

Chapter 1

Introduction

The field of autonomous driving has become an increasingly interesting area of research in both industry and academia during the last decades. Fully autonomous cars, which will drive completely on their own without any kind of human supervision, will have a great impact on society [1][2]. First, self-driving cars will be *safer*, since human mistakes (drowsiness, distractions or just a not fast enough response time) will not be a cause of traffic accidents anymore. This will potentially save thousands of lives per year. In addition, autonomous cars will be very beneficial in terms of *sustainability*. For instance, it will be possible for them to communicate with each other, plan optimal routes to avoid traffic jams or even platoon in highways in order to save energy. On the other hand, this emergent technology will also pose complex, non-technical challenges we might not even conceive yet. To begin with, new legal frameworks will need to be developed to regulate the use of autonomous vehicles. Even how people (drivers or not) will perceive and interact with them is still an unknown, which for sure will be a subject to reflect upon. In sum, autonomous driving will completely change our current vision on transportation systems.

Nowadays, many institutions are actively working on the development of autonomous cars. The DARPA Grand Challenge, in the United States, was an effective starting point of the research in this field, applied to the military area. The winner car of the 2005 DARPA Grand Challenge [3], named Stanley (see Figure 1.1a), was designed by a team from the Stanford Artificial Intelligence Lab, led by Sebastian Thrun. He later promoted

(a) Stanley car (2005)

(b) Volvo self-driving car (2014)

Figure 1.1: Self-driving cars: examples.

the development of the Google Self-Driving Car project within the Google X division. Recently, many automotive companies, such Audi, Mercedes, BMW, Tesla and Volvo Cars (see Figure 1.1b), have started to follow this promising line of research.

Current state-of-the-art autonomous cars are surprisingly robust. For instance, Google cars have traveled more than one million kilometers in the United States without any major failure [4]. Nonetheless, further development is still required in order to fulfill the high safety standards in the automotive industry. Self-driving cars are complex autonomous systems, since they must adapt to a wide variety of scenarios. One of their key competences they must have is *situation awareness*: to estimate what the environment surrounding the car is like. Nowadays, advanced driving assistance systems (ADAS) allow vehicles to drive autonomously under specific scenarios. For instance, it is possible to drive in highways combining radars and cameras to detect other vehicles, lane markings and road signs. In addition, state-of-the-art computer vision technologies allow vehicles to avoid collisions with pedestrians and cyclists in city environments. Nevertheless, fully autonomous cars require a more general solution to be robust. In particular they should also be able to detect any kind of small obstacle on the road, regardless of its properties.

The design of a robust on-road obstacle detection system is a challenging task due to the high variability of the environment, both in terms of obstacles' characteristics and especially weather conditions. Hence, it is likely that different types of sensors are required, complementing each other's weaknesses to adapt to different environmental conditions. The process of combining different sensor data to produce a unified output is commonly called *sensor fusion*. An important question then arises: how to handle information from a diverse set of sensors? A widely used approach consists on transforming the raw sensor data into a common framework: the *occupancy grid*, which is especially suited to work with heterogeneous and noisy sensor data, and eliminates the data association problem. Finally, the information from each sensor is combined by fusing the individual grids.

This thesis work presents an analysis and implementation of current occupancy grid and sensor fusion techniques. For this purpose, the two most common frameworks, Bayesian inference and Dempster-Shafer theory of evidence, are evaluated in terms of accuracy and performance. Furthermore, this report describes the design, implementation and experimental results on a real test vehicle, in collaboration with Volvo Car Corporation.

1.1 Thesis Objectives

The work presented in this thesis project aims to achieve the following goals:

- Occupancy grid building within the Bayesian and Dempster-Shafer frameworks.
- Development of sensor models for specific sensors: radar and lidar.
- Grid fusion within the aforementioned frameworks, paying special attention to the analysis of conflicting information situations.
- Obstacle detection on the fused occupancy grid through image processing.
- Efficient implementation using MATLAB.
- Experimental evaluation on a real test vehicle.

1.2 Major Contributions

This thesis work incorporates the following major contributions to the field:

- Theoretical and experimental **comparison between Bayesian and Dempster-Shafer approaches** for the occupancy grid mapping and obstacle detection problems.
- **Refined sensor models**. The proposed radar model is able to effectively mitigate the impact of multi-echo measurements by analyzing the received power. The lidar model is specifically designed for multi-layer lidars.
- **Sensor performance estimation**. A novel system based on the Dempster-Shafer conflict metric is proposed in order to automatically estimate the performance of every sensor, without the need of ground truth.

1.3 Thesis Outline

The work that has been carried out in this project is described in detail in the present thesis report, according to the following structure. First, a review of the current state of the art is presented in Chapter 2, starting with an overview of the latest advances in autonomous cars and sensor fusion techniques. Afterwards, we investigate low-level fusion techniques based on the occupancy grid mapping paradigm, as well as how to perform obstacle detection. Finally, we investigate how to adapt the sensor contribution in the fusion process according to its performance.

Next, the methodology applied in this project is presented in Chapters 3-7. First, Chapter 3 provides a graphical overview of the architecture of the proposed solution, both in terms of hardware and software. Next, the problem of occupancy grid mapping is described in Chapter 4, considering both the Bayesian and Dempster-Shafer theories. Afterwards, raw sensor data processing and modelling is presented in Chapter 5. In Chapter 6, we explain how to perform grid fusion, paying special attention to conflicting information situations. Lastly, a high-level representation of the obstacles is extracted through image processing of the grid, as described in Chapter 7.

Afterwards, the experimental evaluation is described in Chapter 8, and further analyzed in Chapter 9. The report concludes with Chapter 10, where some reflections about the work done and the obtained results are presented, as well as future work proposals.

Moreover, some research about how the Dempster-Shafer theory can be further leveraged is performed in Appendix A. First, a sensor performance estimator is designed to reduce the contribution of noisy sensors. Finally, we propose an advanced conflict management to increase the detection rate. Lastly, additional information can be found in Appendices B-E, including a description of the hardware and images of the experimental setup, as well as relevant implementation details and derivations.

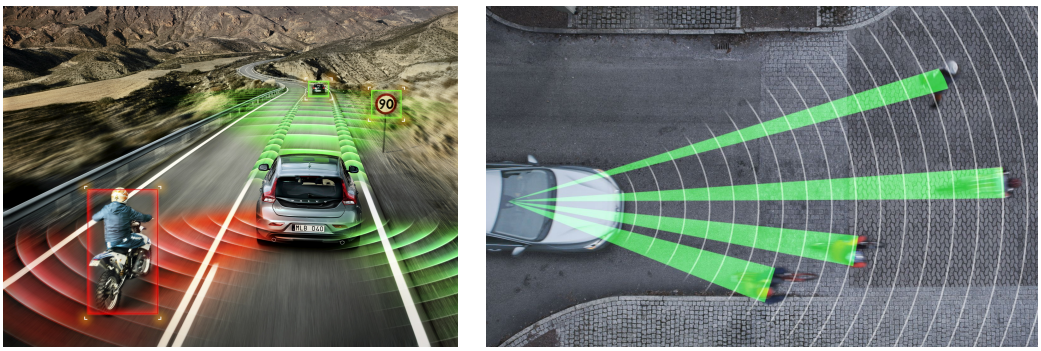
Chapter 2

Background

This chapter presents an overview of the most common approaches to sensor fusion and obstacle detection in the context of advance driver assistance systems (ADAS) and autonomous driving. First, a brief overview of the state of the art in autonomous driving is presented, to motivate the need for research in obstacle detection. Then, we analyze different topics that are directly related to the work presented in this project, such as typical passive and active sensors and common sensor fusion approaches, especially those based on the occupancy grid paradigm. Finally, object detection from occupancy grids will be reviewed. Additionally, more advanced topics such as dynamic environment handling and sensor performance estimation are investigated.

2.1 Autonomous Driving: State of The Art

Nowadays vehicles have multiple driver assistance systems that make driving easier, safer and more comfortable, warn the driver in case of potential hazards or even take full control of the vehicle to prevent traffic accidents. The key to succeed is to have an excellent *situational awareness*, for which a wide variety of sensors is required. A summary of the available systems is illustrated in Figure 2.1, which we describe in the following paragraphs.



(a) Road (b) City

Figure 2.1: Driver assistance systems under different scenarios. Source: Volvo Cars.

- The most popular system is the **adaptive cruise control** (ACC), which uses a radar placed on the front grille of the vehicle. It can detect vehicles in front of the car and adjust the own velocity accordingly in order to keep a safety distance. Many vehicles have this functionality for highway driving, but recently it is also available in city environments, where stop-and-go maneuvers and denser traffic are more challenging.
- In addition, some cars are equipped with cameras to extract basic information from the environment. For instance, it is possible to perform **lane marking detection**, which helps the vehicle to follow the road and keep on track. In addition, cameras can identify **road signs**, such as speed limit signs, and adjust the vehicle velocity accordingly, as presented in Figure 2.1a. This technology is however very sensitive to adverse weather conditions, such as rain and snow.
- A combination of radar and cameras can be used to **avoid collisions** with pedestrians, cyclists and other vehicles, as shown in Figure 2.1b. The system usually warns the driver about the potential collision, and performs an emergency brake if it is unavoidable, in order to minimize the injuries.
- In addition, **parking assistance systems** are very common nowadays, using ultrasonic sensors and cameras to detect obstacles in the proximities of the vehicle. Some companies even offer completely autonomous parking solutions.
- Lastly, the research community has recently become interested in performing **SLAM** (Simultaneous Localization And Mapping), given that the required localization accuracy for autonomous driving is in the order of centimeters, which cannot be achieved with GPS. Lidar sensors are especially suited for this purpose, given their excellent resolution and accuracy. However, due to the high production cost, they are not yet integrated into commercial vehicles, but they will eventually become a crucial part of the car's sensory architecture as manufacturing costs decrease.

These technologies allow cars to drive autonomously under a small set of controlled scenarios, such as highways or simple urban environments. To improve the field of application of autonomous vehicles, and to fulfill safety requirements, the situational awareness must be further improved. In this project we contribute to it by developing a general-purpose obstacle detection module, designed to detect small obstacles on the road.

2.2 Sensors for Obstacle Detection

The challenge of obstacle detection in automotive applications using only one type of sensor has been well studied in the literature. Even though we will focus on multi-sensor fusion, analyzing the performance of single-sensor approaches will be insightful to understand their strengths and weaknesses and motivate the choice of sensors for this project.

Discand *et al.* [5] present a brief study about the available sensors for obstacle detection. On the one hand, *passive* sensors only capture energy from the environment, and they are therefore cheap and produce no interference to other sensors. The most common

passive sensors for automotive applications are **cameras**, providing high resolution images from the environment. However, an energy source is required: visible light or infrared radiation for the case of normal and infrared cameras, respectively. Many authors have performed vision-based obstacle detection by means of normal color images [6][7] and infrared images [8], to cite a few. Even though they achieve a good performance in obstacle detection and even classification, this approach may not work well at night or under adverse weather conditions (rain, snow, dust). Furthermore, monocular cameras do not directly provide depth information. It is then common to use **stereo cameras**, as seen in the literature [9–11]. This approach relies on the same suitable weather conditions as in the case of monocular cameras to work properly.

On the other hand, it is also frequent to use *active* sensors, which emit a waveform and measure the received reflections from potential targets. In this category we include radar, lidar and sonar.

- **Radar** has recently been incorporated into the field of robotics and autonomous vehicles [12][13]. Radars designed for automotive applications usually work in the millimeter-wave band, around $76 - 77$ GHz. Their main advantage is an all-weather capability, therefore compensating for the main vision weakness [14]. They have a good range resolution and are even able to measure the radial speed of moving objects by means of the Doppler effect. Their drawback is often a poor angular resolution, which is directly related to antenna size. Automotive applications must therefore trade off resolution and sensor size. However, the use of phased-array receivers and superresolution algorithms allow for a much better resolution, up to $\pm 1^{\circ}$ [15]. We notice that radars and cameras are good in orthogonal directions: radial and cross-radial, respectively, so they complement each other very well.

- **Lidar** is based on the emission of laser beams at fixed angular steps, usually in the infrared band (900 nm). It provides an excellent angular resolution and range accuracy, and it is becoming very popular for SLAM and obstacle detection in autonomous vehicles [16][17]. Nonetheless, they usually suffer from light scattering under heavy rain, mist, snow or dust conditions [18]. They also have difficulties with specular surfaces as well as transparent objects, such as glass. Another drawback, compared to radar, is that they usually cannot measure the radial speed of the targets.

- **Sonar** is based on the same principle as radar, but emitting ultrasonic waveforms instead. They are cheaper but very sensitive to weather conditions: the speed of sound is highly dependent on the temperature. Therefore, they are not very common in the context of road vehicles (for autonomous driving), although there exist some applications in the literature [19]. They are more popular in the field of underwater autonomous vehicles.

As it can be observed, every type of sensor has its own strengths and weaknesses, and only one cannot handle every situation. In order to create a robust obstacle detection system it is therefore necessary to combine them so that they can compensate for each other's drawbacks. This approach is widely known as sensor fusion, which we investigate further in this project.

2.3 Sensor Fusion

To increase the robustness as well as to improve the total estimation capability of a sensory system, a lot of research effort has recently been dedicated towards combining complementary sensor data, also known as sensor fusion. Khaleghi *et al.* [20] present a comprehensive review of the current challenges and approaches to sensor fusion. The main difficulties arise from spatio-temporal registration of the sensor data, data association, the management of conflicting information and especially the heterogeneity of the sensors, which requires a common framework in which all sensor data can be represented. Luo *et al.* [21], show that sensor fusion can be performed at different levels, as presented in Figure 2.2.

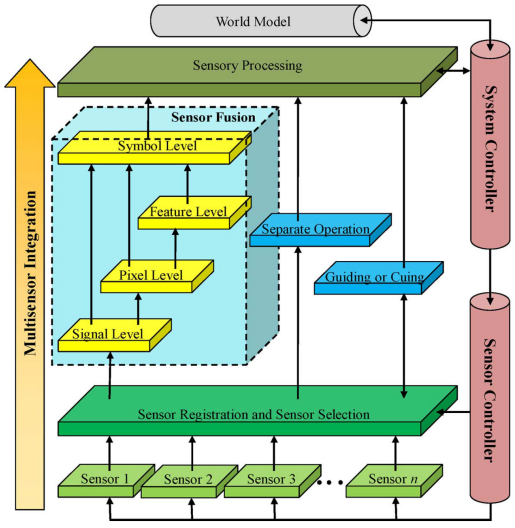


Figure 2.2: Different multi-sensor fusion levels, as shown by Luo *et al.* [21].

As can be observed, there exist many approaches to sensor fusion in the context of driver assistance systems. Below, we present a number of proposed methods and applications that can be found in the literature, according to the level of abstraction at which the sensor fusion is performed.

2.3.1 High-Level Sensor Fusion

Some authors choose to fuse the sensor data at a *high level*. In this case, the sensor information is represented as high-level structures, such as features or objects. Perrollaz *et al.* [22] propose a long-range obstacle detection system based on stereovision and a laser scanner. Obstacles are detected and tracked by the laser scanner, and the tracks are later validated by the vision system. This approach greatly reduces the false-alarm rate as compared to single-sensor methods. Floudas *et al.* [23] perform high-level fusion of long and short-range radars, a laser scanner and a lane camera system with redundant fields of view, mounted on a truck. Each sensor outputs a number of tracks corresponding to objects, and they are fused afterwards. Wang *et al.* [24] propose to fuse a millimeter-

wave radar and a monocular camera for on-road obstacle detection and tracking. Object detection and tracking is performed for both the radar and the camera, extracting a high-level track representation. Only matching tracks from both sensors are considered as valid. This reduces the false-alarm rate as well as the computational time of conventional vision-only approaches. Furthermore, Kubertschack *et al.* [25] propose a unified architecture for sensor fusion applied to static environments mapping. The sensor data is translated into high-level object lists, which share a common format across different sensor inputs.

The main difficulty when performing high-level sensor fusion is the **data association** problem: given a set of high-level representations of objects (*tracks*) for every sensor, one must determine which of those tracks belong to the same object. To this extent, several techniques can be found in the literature, such as Global Nearest Neighbour (GNN), Joint Probability Data Association (JPDA) [26] or Multiple Hypothesis Tracking (MHT) [27], to mention a few. It is also required to perform tracking and filtering, for which the common solution is to use separate Kalman filters (or any of its variants) or particle filters for each of the tracks. More complex techniques such as Interacting Multiple Models (IMM) provide more robustness when tracking multiple targets.

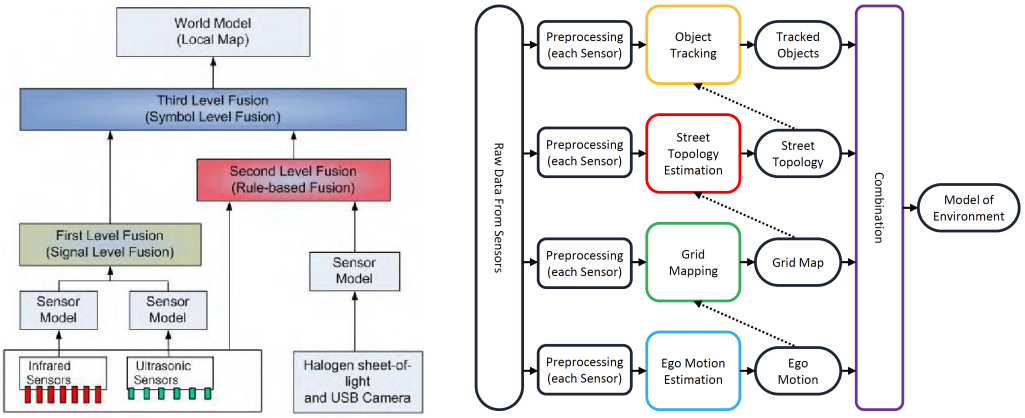
Another potential drawback of high-level sensor fusion is that the received signal from the individual sensors might be too weak to trigger the creation of an object track. In these cases, raw, low-level information must be used to effectively detect obstacles.

2.3.2 Hierarchical Sensor Fusion

It is also possible to perform sensor fusion in a hierarchical or *multi-level* way, although it is not very popular in the literature. Kim and Lee [28] propose a three-level fusion architecture for efficient and accurate map building, based on vision, ultrasonic and infrared sensors, as can be observed in Figure 2.3a. First, infrared and ultrasonic sensors are fused at a signal level through an occupancy grid. A second level enhances the camera information with infrared and ultrasonic sensors. Finally, the third level fuses the results from the previous ones in a probabilistic way. Their results show an improvement in efficiency and accuracy in the obtained map. Furthermore, Lindner and Wanielik [29] design a pre-crash safety system by fusing the information from a multi-layer laser scanner and radar for vehicle detection. At the lowest level, an occupancy grid is built from the raw laser data. Features such as edges and corners are extracted from it at a second level, and they are associated to higher-level tracks afterwards. Finally, at an object level, the radar information is taken into account to validate the laser information and estimate the velocity of the moving objects. More recently, Nuss *et al.* [30] combine occupancy grid maps, digital road maps and multi-object tracking for a rich and robust environmental perception system, as shown in Figure 2.3b.

2.3.3 Low-Level Sensor Fusion

Finally, many authors decide to perform sensor fusion at a *low level*. The main reason to prefer low-level fusion approaches is that the decision rule —determining the final state of the map, position of objects, and so on— is applied at very last stage. This way, no information is lost in intermediate stages, which occurs in high-level sensor fusion approaches.



(a) Kim and Lee [28] (b) Nuss *et al.* [30]

Figure 2.3: Hierarchical sensor fusion architectures.

The most commonly used approach for obstacle detection in robotics is based on the **occupancy grid** paradigm, introduced by Elfes and Moravec [31][32] in the context of robot mapping using sonar sensors. The environment is represented as a grid with finite resolution, where each cell is described by a random variable following a binomial distribution that represents the probability of occupancy. This is a solid framework upon which many current state-of-the-art sensor fusion approaches are built. Its main advantage is that it is easy to translate any sensor measurement into a probability distribution over the grid, from which the integration of heterogeneous information is straightforward.

There exist a large number of low-level sensor fusion approaches in the literature. Kumar *et al.* [33] fuse the information from a stereo camera set and an infrared sensor in order to obtain a 3D occupancy map of the environment. They show how the global error of the fused map is smaller than the maps generated by the individual sensors alone.

Many authors rely on the occupancy grid to fuse information from a monocular camera and a laser rangefinder. Baig and Aycard [34] apply the Mixture of Gaussians (MoG) technique to extract moving objects from the video stream. This information is later projected into an occupancy grid. The laser scanner maps the static environment. The final grid is the result of a linear combination between the previous two. The main drawback of this approach is the requirement of a static robot to successfully perform background subtraction. Further work fuses data from a multi-layer lidar [35], as shown in Figure 2.4. Homm *et al.* [36] use a laser scanner and a monocular camera for lane detection. They take advantage of the high reflectivity of the lane markings, easily detected by the laser, and popular edge detection systems for image processing. They show that these two systems perfectly complement each other in changing illumination conditions. Recently, Kang *et al.* [37] and Nuss *et al.* [38] have successfully performed mapping by fusing laser scanner and monocular camera information into a single occupancy grid.

Stereo cameras are more popular in the context of driver assistance systems, since they can provide with a measure of depth. It is possible to fuse stereo and lidar information by means of an occupancy grid. Paromtchick *et al.* [39] propose an object detection system based on the Bayesian Occupancy Filter, which is an extension to the classical occupancy

(a) Camera view (b) Occupancy grid

Figure 2.4: Occupancy grids in automotive applications. Measurements from a multi-layer lidar are integrated into an occupancy grid. Figure from Baig *et al.* [35].

grid framework to account for dynamic environments. The implementation is tested on a real vehicle in both urban and highway environments. The work is further improved by Adarve *et al.* [40]. Their results show great robustness in case of faulty or spurious measurements from any of the sensors and when they provide conflicting information.

A specially interesting combination of sensors is lidar and radar, since they emit waveforms in different bands of the electromagnetic spectrum and therefore they better complement each other's weaknesses. Although it is not usual, it is possible to perform low-level fusion of these sensors. Garcia *et al.* [41] use a laser scanner and a long range radar mounted on a truck. Two occupancy grids are built for each sensor and fused afterwards. Finally, they extract objects from the grid and perform a high-level tracking.

Stepan *et al.* [42] analyze the possibilities of robust data fusion using a monocular camera, a laser scanner and a sonar sensor. Individual occupancy grids are built for every sensor and fused together afterwards. They also take into account the different sensor precision when performing the fusion. The performance of the fused grid is evaluated in terms of its applicability to grid-based path planning. Even though they have three sensors, they only perform fusion between pairs of sensors.

Finally, it is also possible to perform low-level data fusion combining radar and monocular cameras, as shown by Groover *et al.* [43]. However, they do not rely on the general occupancy grid framework; instead, they cluster the data in both radar and vision inputs to extract blobs, which are afterwards matched in the fusion process. This is a more *ad-hoc* procedure that requires a complex preprocessing of the sensor data.

2.4 Occupancy Grid Mapping: Inference Frameworks

As seen in the previous section, the occupancy grid mapping is a versatile framework to perform low-level multi-sensor fusion. The problem is reduced to estimating the state of occupancy for every cell in the map. Nevertheless, the occupancy grid is just a framework, which requires an inference theory to estimate the probability of occupancy for each cell. For this purpose, there exist a wide variety of estimation techniques in the literature, as discussed by Khalegui *et al.* [20] (see Figure 2.5).

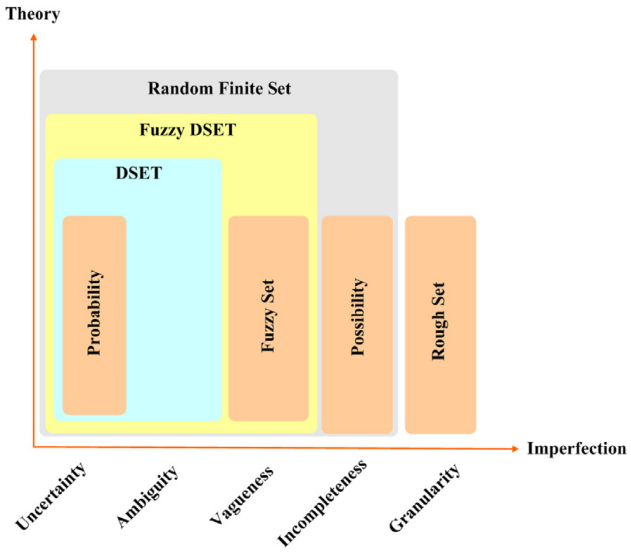


Figure 2.5: Inference frameworks for fusing noisy and incomplete data, presented by Khalegui *et al.* [20].

Ivanjko *et al.* [44] present a comprehensive experimental comparison of the main approaches towards occupancy grid mapping, based on sonar sensor data. First, the classical **Bayesian probability theory** is presented. It is the most common approach since it builds upon solid probabilistic grounds: the Bayes' theorem. This framework was originally used by Elfes [31] when he introduced the occupancy grid paradigm for map building. Many authors have later followed this approach in the context of automotive applications.

Second, the **Dempster-Shafer Evidence Theory** (DSET) is another interesting approach for integrating information. Some researchers consider it a generalization of the classical probability theory. It has been proven to be especially good in data fusion problems, since it is able to explicitly model ignorance and handle conflicting data. Many authors make use of the Dempster-Shafer theory for multi-sensor fusion in the context of driver assistance systems. Wu *et al.* [45] work on human-machine interaction (HMI) and use the Dempster-Shafer approach to fuse video and audio information. They later extend the work [46] with a modified combination formula that allows for the weighting of the different sensors. Xiang *et al.* [47] design a fusion system for autonomous off-road vehicles combining radars, lidars, cameras and ultrasounds at different levels. Similarly, Soleimanpour *et al.* [48] also use this framework to perform localization. They leverage the conflict detection capabilities of the Dempster-Shafer theory to detect irrelevant sensor outputs. Furthermore, Delafosse *et al.* [49] perform SLAM fusing information obtained from cameras with the Transferable Belief Model, a modification of the Dempster-Shafer theory. Cao *et al.* [50] show how mapping can easily be done by integrating lidar measurements into the occupancy grid framework and the Dempster-Shafer theory. Finally, an interesting work by Plascencia and Bendtsen [51] fuses SIFT features extracted from camera images and sonar measurements on an occupancy grid.

The Bayesian and Dempster-Shafer theories will be studied in detail in this project.

Nonetheless, it is worth mentioning that there exist a number of other techniques for integrating sensor information into a grid map. Oriolo *et al.* [52] introduced the concept of **fuzzy maps**, where the fuzzy sets theory is combined with an occupancy grid to solve the mapping problem. **Borenstein maps** [53] were introduced with the purpose of fast map building where accuracy is not as important as computation time, so it should serve as a real-time solution for obstacle avoidance. Another representation is the **MURIEL map**, introduced by Konolige [54]. It was especially designed to handle outliers in sonar data, mostly due to specular reflections and spurious measurements.

2.4.1 Dynamic Environment Handling

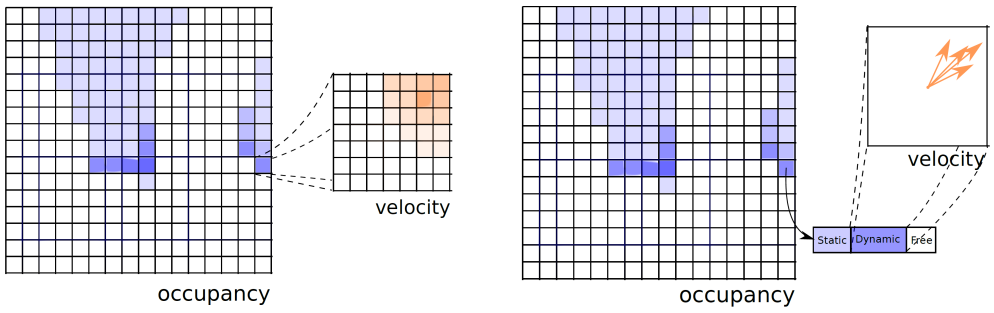
The estimation techniques mentioned so far rely on a strong assumption: the map to be estimated is static. Therefore, dynamic objects cannot be properly modelled in this framework. In this project we limit ourselves to the detection of static on-road obstacles, given that dynamic objects, mostly vehicles, pedestrians, and so on are already detected by current driving assistance systems. Nevertheless, it is interesting to know the available approaches to handle dynamic environments within the occupancy grid framework, which allows for low-level sensor fusion.

A popular solution is the **Bayesian Occupancy Filter** (BOF), originally presented by Coue *et al.* [55]. In this case, a four-dimensional grid is maintained; two dimensions represent the x, y coordinates of each cell, and the remaining two dimensions are used to estimate the probability distribution of the velocities of each cell in x and y direction, as depicted in Figure 2.6a. The grid is updated following a predict-correct scheme, where a constant velocity model is assumed to account for the cell motion. The main disadvantage of this approach is the computational cost and memory footprint, considering that it increases exponentially with the number of dimensions. Therefore, it is not suitable for real-time applications.

To greatly reduce the computational complexity, Chen *et al.* [56] proposed a modification to the algorithm. Instead of having a velocity grid for each cell, they consider a two-dimensional grid where each cell contains a probability distribution of its velocity, apart from the classical occupancy probability. In addition, they propose an efficient implementation which is highly parallelizable, therefore suitable for running on GPUs to achieve real-time performance.

Negre *et al.* [57] introduce the Hybrid Sampling Bayesian Occupancy Filter (HSBOF), which combines the classical occupancy grid to estimate the probability of occupancy with a particle filter for each cell to estimate its velocity, as illustrated in Figure 2.6b. They implement the algorithm in GPU and achieve real-time performance while accurately estimating the occupancy and velocity of objects on the map.

A different approach, presented by Meyer-Delius *et al.* [58], consists on implementing the predict-update algorithm using a Hidden Markov Model (HMM). It is based on an initial distribution, transition probability matrix and observation probability matrix. Only the transition probability matrix, which models the dynamics of the environment, needs to be estimated, for which the use of the classical Expectation-Maximization (EM) algorithm is suggested. Nonetheless, the complexity of the algorithm likely prevents this approach from running on real time. This framework allows for both sensor fusion and object



(a) BOF (b) HSBOF

Figure 2.6: Dynamic occupancy grid: BOF and HSBOF. The latter uses particles to estimate the velocity for each cell, being more accurate and efficient. Images from Negre *et al.* [57].

tracking, and has been used widely in the literature [59][35].

Recent approaches leverage the advantages of the Dempster-Shafer theory to detect dynamic objects. For instance, Moras *et al.* [60] as well as Jungnickel and Korf [61] show how the conflicting information can be used to detect moving objects. A grid containing the conflicts resulting from temporal fusion is later processed to extract and track clusters belonging to the moving objects.

2.5 Sensor Weighting and Confidence Estimation

Most of the grid-based sensor fusion approaches are considered to be *democratic*, i.e. every sensor contributes equally to the final result. This might not be desirable in certain contexts. For instance, under adverse weather conditions it is likely that a lidar or a camera return incorrect and noisy measurements, whereas a radar sensor would be unaffected by this. It is thus especially interesting to design a system that can analyze the performance of each sensor in real time and assign some quality measure to it, which will have an impact on the sensor contribution in the fusion process. The main difficulty is that usually a ground truth is not available.

Some authors have tackled this issue. Zhou *et al.* [62] propose a sensor fusion framework based on a linear combination of the sensor inputs, each of them having an associated weight. The weights are computed as a solution to an optimization problem with the aim of minimizing the entropy of the fused distribution, taking into account only empirical data from the sensors.

Wu *et al.* [46] introduce an extension to the classic Dempster-Shafer sensor fusion framework by including a weight for every sensor. They propose two ways of weighting: static (fixed weight) and dynamic. The dynamic approach computes a weight by analyzing the performance over the last set of measurements. They assume that the sensors provide a reliable measure of self-confidence, either as a result of comparing their measurements to a ground truth or through additional information channels. Unfortunately, current sensors can only tell whether they are working properly (i.e. not faulty), and a ground truth is usually not available in real-world applications. Kumar *et al.* [63] propose a similar approach within the Bayesian domain by introducing an additional term to the classical

formulation, to represent the probability that a measurement is spurious. Whenever the measurements are inconsistent with each other the variance of the sensor distribution is increased, thus having a smaller contribution in the fusion process.

More recently, some authors explore the advantages of the Dempster-Shafer framework to assess the reliability of the sensor readings. Carlson and Murphy [64] exploit the measure of *conflict* that is automatically provided by the Dempster-Shafer combination formula, in the context of unknown, dynamic environments where no ground truth is available. They only make a reasonable assumption: the environment is *consistent*, i.e. a cell cannot be occupied and empty at the same time. They perform a series of experiments using sonar and laser data from a robot, and they evaluate different inconsistency indicators based on the Dempster-Shafer conflict measure. Their experiments show successful results, being able to detect, estimate and isolate sensing problems in unknown environments. A similar study, also based on the Dempster-Shafer theory, is presented by Gage [65]. Several conflict measures are tested with satisfactory results for sensor accuracy estimation and data isolation, improving the overall quality of the fused map.

2.6 Object Detection on Occupancy Grids

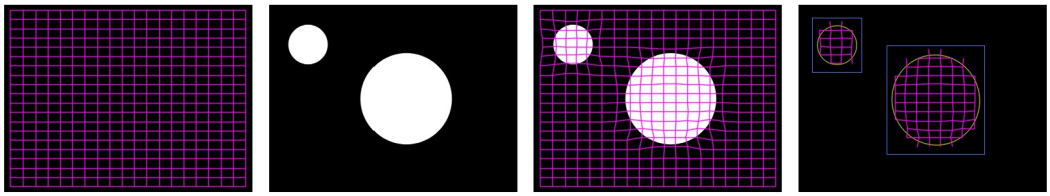
The ultimate goal of the project is to detect obstacles on the road. This information is to be extracted from the fused occupancy grid, after having integrated all the sensor measurements. Several approaches can be found in the literature.

First, the most intuitive approach is to leverage the segmentation techniques from the Computer Vision domain, since the occupancy grid can be considered as a single-channel image. Common segmentation approaches first perform a binary classification for every pixel in the image as *background* or *foreground*. Next, it is possible to cluster large groups of foreground pixels to extract a high-level representation of the objects. This is easily performed using the standard **connected components** technique [66], where a neighbourhood of 4 or 8 pixels is analyzed to detect pixels that are connected to each other.

A similar approach is used by Nguyen *et al.* [67], who implement a **hierarchical clustering** to extract objects from an occupancy grid updated with a stereo camera set. First, initial segments are created by grouping pixels which are sufficiently close to each other, according to some threshold over the Euclidean distance. Then, segments are merged together if the closest distance among them is smaller than a given threshold. The operation is repeated until no more segments can be merged.

A more advanced fast clustering technique based on Self-Organizing Networks (SON) is proposed by Vasquez *et al.* [68]. Every time the occupancy grid is updated, a graph of nodes and edges is learned based on the occupancy probability of each cell on the grid. Next, graph theory is applied in order to perform cuts to those edges with low weights, which indicate that two nodes are not likely part of the same object. The nodes that remain connected represent the extracted obstacles. In addition, the weights and positions of the nodes allow for the computation of an abstract representation of the object through a bounding box, a Gaussian model, and so on. Figure 2.7 shows the process of building the network and extracting objects.

In the context of the Bayesian Occupancy Filter, Mekhnacha *et al.* [69] introduce the Fast Clustering-Tracking Algorithm (FCTA) with the aim of detecting moving objects.



(a) Initial SON configuration (b) Detection bitmap (c) SON after learning (d) Extracted objects

Figure 2.7: Object extraction from occupancy grids using a Self-Organizing Network. Figure from Vasquez *et al.*[68].

Objects are clustered by first connecting cells with high probability of occupancy in an eight-neighbour fashion. Since the BOF also provides a velocity estimate for each cell, they improve the clustering by additionally taking into account the Mahalanobis distance between the velocity distributions of each cell. This greatly reduces the number of noisy small obstacles detected with classical approaches.

Finally, there exist many other popular clustering techniques, such as k-Nearest Neighbours, Expectation Maximization and so on. However, they are not applicable to this problem, since they require the number of clusters to be known in advance. This information is not available since our goal is to detect obstacles in unknown environments. In addition, they tend to be quite computationally heavy and the result is dependent on the initialization, so several runs need to be performed to robustly determine the final solution.

Chapter 3

System Overview

In this chapter we briefly present a general overview of the project, both in terms of the hardware used and also the software architecture of the developed system.

3.1 Hardware Architecture

For this project we make use of a Volvo test vehicle equipped with a variety of sensors, as illustrated in Figure 3.1. In particular, the sensor architecture consists of two short-range radars and a front lidar. The complete sensor specifications can be found in Appendix D. In addition, GPS localization and ego-vehicle motion from an inertial measurement unit and wheel odometry are also available.

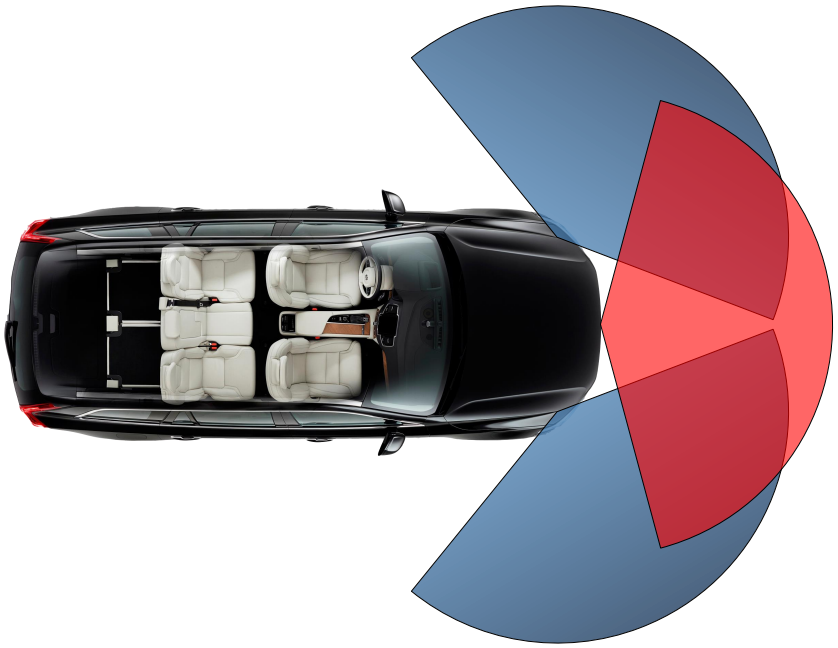


Figure 3.1: Sensor configuration on the vehicle: radar (blue) and lidar (red). The ranges are not to scale. Approximate, only for illustration purposes.

3.2 Coordinate Frames Convention

Throughout the project we will handle data from the detections from a variety of sensors, and it is essential to know in which coordinate frame they are expressed. We will refer to some of these coordinate frames when performing coordinate frames transformations of the sensor data. The following coordinate frames, as depicted in Figure 3.2, are defined in this project:

- `World` (W). A fixed reference coordinate frame, with respect to which the vehicle is moving.
- `Vehicle` (V). Placed in the middle of the rear wheel axle, at the ground level.
- `Velodyne` (Vel). When available, we will use a Velodyne lidar to generate ground truths of the environment. It will be mounted on top of the vehicle, approximately in the center.
- `Left Radar` (LR) and `Right Radar` (RR), mounted on the front corners of the vehicle, concealed by the car body.
- `Lidar` (L). Mounted in the center of the front bumper, approximately at a 40 cm height above the ground.

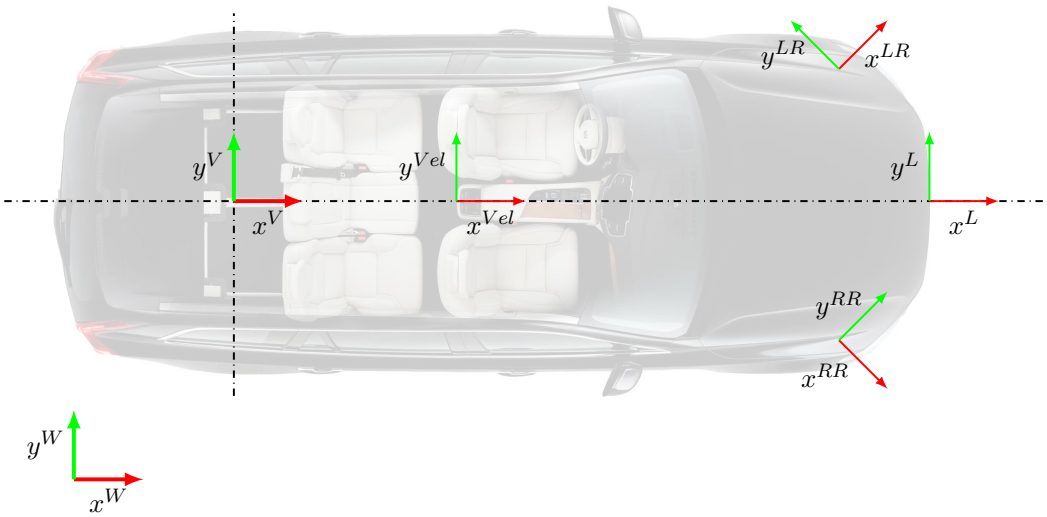


Figure 3.2: Coordinate frame definitions (ISO 8855:2011/DIN 70000).

3.3 Software Architecture

Finally, a general overview of the software implementation is presented in Figure 3.3.

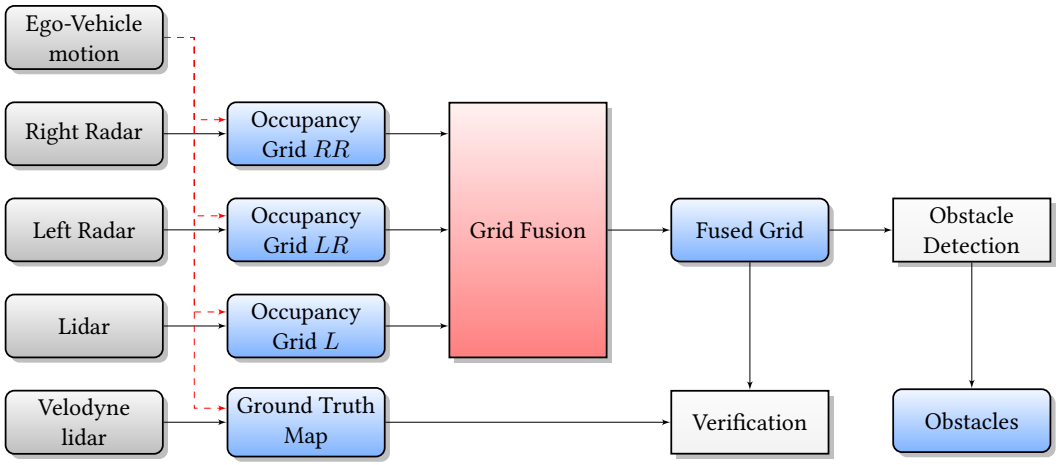


Figure 3.3: Proposed software architecture.

The workflow of the proposed solution is summarized as follows, from left to right:

1. First, data is gathered from all the sensory inputs, as well as the ego-vehicle motion based on wheel odometry, an inertial measurement unit (IMU) and GPS.

2. Next, an occupancy grid is maintained and updated for each of the sensors. Therefore, the data from every sensor is filtered individually.

3. After the grids have been updated with the latest sensor data, they are fused together into another occupancy grid: the fused grid.

4. The fused grid contains all the information available from the sensors. It is then possible to analyze it through image processing techniques and detect potential obstacles, which are then stored in a list to create the final output of the system.

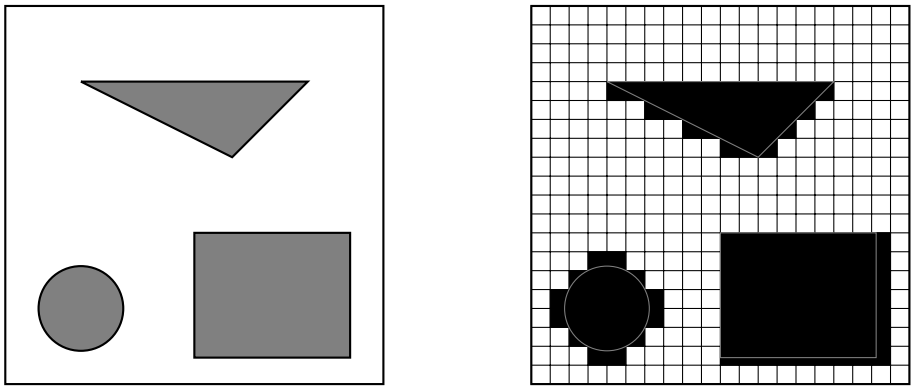
5. In addition, we will make use of a Velodyne lidar to generate a ground truth of the environment. Another occupancy grid will be created for this sensor and the others will be compared to this one as a verification step.

In the following chapters we will describe in detail the methodology applied at each step of the algorithm outlined above.

Chapter 4

Occupancy Grid Mapping

The occupancy grid framework, introduced by Elfes [31], is one of the most popular approaches towards the fusion of diverse sensory information in the context of environment mapping. The main idea behind the occupancy grid is to represent the map as a two-dimensional grid with fixed dimensions and spatial resolution. Each of the cells contained in the grid is described by a random variable corresponding to the state of occupancy of the cell. An example of an occupancy grid is presented in Figure 4.1.



(a) Real map

(b) Occupancy grid representation

Figure 4.1: Map representation in the occupancy grid framework. White: empty space; black: occupied space.

Due to the finite resolution of the grid, a loss of accuracy is inevitable in this kind of representation, as can be observed in the figure. It is worth noting that Figure 4.1b represents a *ground truth* of the map. On the contrary, when updating the occupancy grid using sensor information, each cell will have a probability associated with its state of occupancy, instead of a binary value (occupied/empty).

Problem formulation

Let us introduce the formulation that we will use throughout this project:

- The **vehicle pose** at a given time t is described as a K-dimensional vector:

$$\mathbf{x}_t = \begin{pmatrix} x_t^1 & x_t^2 & \dots & x_t^K \end{pmatrix}^\mathsf{T} \tag{4.1}$$

- The **sensor measurement** at a given time t from a sensor S is represented as a M-dimensional vector:

$$\mathbf{z}_t^S = \begin{pmatrix} z_t^{S,1} & z_t^{S,2} & \dots & z_t^{S,M} \end{pmatrix}^\mathsf{T} \tag{4.2}$$

- Finally, the **map** is described in terms of a two-dimensional grid:

$$\mathbf{m} = \{m_{ij} : 1 \leq i \leq N_H, 1 \leq j \leq N_W\} \tag{4.3}$$

  where N_H and N_W are the number of cells in height and width, thus making a total of $N_H \times N_W$ cells. The probability distribution is updated taking into account the sensor measurements and the vehicle pose.

Furthermore, the occupancy grid is just a framework over which the map can be efficiently represented. There exist different inference theories to actually update the probability distribution by integrating sensor information. In this project we analyze the performance of the two most common probabilistic approaches for updating the occupancy grid given sensory information: the classical Bayesian inference theory and the Dempster-Shafer theory of evidence, described in detail in the following sections.

4.1 Bayesian Inference Theory

Under the Bayesian inference theory, the problem formulation for the occupancy grid mapping typically consists on computing the posterior probability of the map, given all the measurements $\mathbf{z}_{1:t}$ and vehicle poses $\mathbf{x}_{1:t}$ so far. The most general case assumes that the map, $\mathbf{m}$, is dynamic and evolves over time. The posterior is then:

$$p(\mathbf{m}_{1:t}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}) \tag{4.4}$$

In this case we are interested in estimating only the current state of the map, $\mathbf{m}_t$. This can be estimated iteratively by a classical predict-update procedure:

$$\textbf{Predict: } p(\mathbf{m}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1}) = \int p(\mathbf{m}_t|\mathbf{m}_{t-1}) \cdot p(\mathbf{m}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1}) \mathrm{d}\mathbf{m}_{t-1} \tag{4.5}$$

$$\textbf{Update: } p(\mathbf{m}_t|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{m}_t, \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}) \cdot p(\mathbf{m}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t})} \tag{4.6}$$

where it is assumed that the evolution of the map only depends on its previous state, not on the vehicle pose or the measurements. The previous expression is usually hard to compute, since the dynamics of the map, $p(\mathbf{m}_t|\mathbf{m}_{t-1})$, are normally unknown or unpredictable. For these reasons the literature mostly focuses on the assumption that the map is **static**. In this case, the goal is to compute the following posterior:

$$p(\mathbf{m}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}) \tag{4.7}$$

The dependence relations between the previous variables are depicted in terms of a graphical model based on Thrun *et al.* [70], as shown in Figure 4.2.

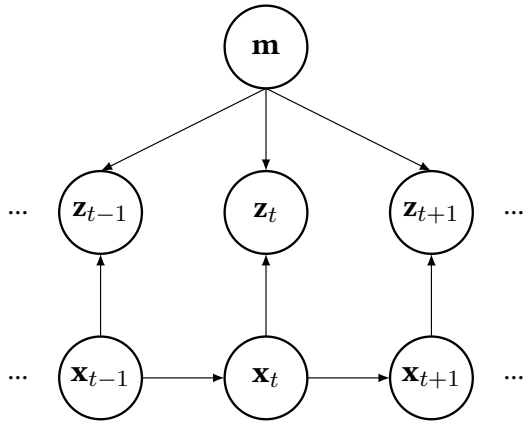


Figure 4.2: Graphical model describing the dependence relations between the static map, the sensor measurements and the vehicle states over time.

Discussion

We remark the implications of the inference model presented in Figure 4.2:

- The map is assumed to be **static**, i.e. it does not change over time. This is a common assumption applied in the literature that allows for a great simplification in the mathematical derivations. The main drawback is that dynamic maps cannot, in theory, be modelled by this approach. As discussed in Chapter 2, there exist other alternatives that can handle dynamic maps, although they are out of the scope of this project. However, a solution based on a *forgetting factor*, as presented in Section 4.5, is proposed in order to more easily support dynamic environments (e.g. containing moving objects).

- The sensor measurements depend on the vehicle pose and the static map. In addition, at each time step, the current vehicle pose $\mathbf{x}_t$ depends only on the previous one, $\mathbf{x}_{t-1}$. This is known as a first-order Markov chain.

- Sensor measurements at different time steps are **conditionally independent** given the map. That is:

$$p(\mathbf{z}_t, \mathbf{z}_{t-1}|\mathbf{m}) = p(\mathbf{z}_t|\mathbf{m}) \cdot p(\mathbf{z}_{t-1}|\mathbf{m}) \tag{4.8}$$

  where $p(\mathbf{z}_t|\mathbf{m}, \mathbf{x}_t)$ is called the *forward sensor model*. It represents the probability of obtaining a sensor measurement given the current status of the map. It should be noted that the forward sensor model also depends on the vehicle pose, $\mathbf{x}_t$. The conditional independence assumption is only valid if the sensors **are not biased**, i.e. if they do not have a common source of error. Otherwise, the measurements would not depend exclusively on the map and there would be additional dependence relations. This property will be exploited later on in the derivations.

4.1.1 Conventions

Under the Bayesian inference framework, each cell m_{ij} is assumed to be a binary random variable. The following conventions are assumed in this work:

- $p(m_{ij}) = 0.0 \implies$ the cell is `empty` with total certainty.
- $p(m_{ij}) = 1.0 \implies$ the cell is `occupied` with total certainty.
- $p(m_{ij}) = 0.5 \implies$ the state of the cell is `unknown`.

The notation $p(m_{ij}) = p(m_{ij} = \texttt{occupied})$ and $p(\overline{m_{ij}}) = p(m_{ij} = \texttt{empty})$ will be used in the rest of the report. Since m_{ij} can only have two states, the following constraint applies:

$$p(m_{ij}) + p(\overline{m_{ij}}) = 1 \tag{4.9}$$

Therefore, the complete map $\mathbf{m}$ can have $2^{N_H \times N_W}$ states. Computing the posterior presented in Equation 4.7 thus becomes intractable from a computational point of view.

4.1.2 Practical Approximation

A popular solution is to make an additional assumption: all the cells in the grid are **independent** of each other. This is not true at all; for example, occupied cells belonging to the same object are related to each other. However, it has been shown that it is a fair enough approximation of the problem and allows for a computationally affordable way of estimating the posterior. Taking this assumption into account, the posterior can be decomposed as follows:

$$p(\mathbf{m}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}) = \prod_{i=1}^{N_H} \prod_{j=1}^{N_W} p(m_{ij}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}) \tag{4.10}$$

The problem is now reduced to estimating the posterior of single cells independently. The algorithm to solve it is called the Binary Bayes Filter [70], and its derivation is presented in Appendix B. The result is a recursive formula in terms of the *log-odds ratio*, $l(m_{ij})$, as presented in Equation 4.11:

$$l_t(m_{ij}) = \log \frac{p(m_{ij}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t})}{1 - p(m_{ij}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t})} = l_{t-1}(m_{ij}) + \log \frac{p(m_{ij}|\mathbf{z}_t, \mathbf{x}_t)}{1 - p(m_{ij}|\mathbf{z}_t, \mathbf{x}_t)} - \log \frac{p(m_{ij})}{1 - p(m_{ij})} \tag{4.11}$$

Three terms are involved in this equation. First, the previous state of the map, $l_{t-1}(m_{ij})$. The second term involves the log-odds of the probability distribution $p(m_{ij}|\mathbf{z}_t, \mathbf{x}_t)$, which is called *inverse sensor model* in the literature. It represents the probability of each cell in the map given the current measurement $\mathbf{z}_t$ and vehicle pose $\mathbf{x}_t$. This is the step where the current estimate of the map is updated with new sensor information. Finally, the third term represents the *prior* probability of the map, which will normally be $p(m_{ij}) = 0.5$ since the map is unknown a priori. If a map is available, this information can be integrated through this last term.

4.1.3 Grid Initialization

The grid is initialized assuming that the state of every cell is `unknown`, since no prior information is available in this project (we only rely on sensor data). Therefore:

$$p(m_{ij}) = 0.5 \quad \forall i, j \tag{4.12}$$

4.1.4 Probability Limitation

A practical consideration to be taken into account is not to let the probability of occupancy reach the value of 0 or 1. This would make the log-likelihood tend to $\pm\infty$ and therefore new sensor measurements would not contribute to keep updating the probability of occupancy. Therefore, whenever performing an update, we set the following conditions:

$$p(m_{ij}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}) \in [P_{\min}, P_{\max}] \tag{4.13}$$

A value of $P_{\min} = 10^{-5}$, $P_{\max} = 1 - P_{\min}$ is chosen for this project. This is a fair enough approximation of the complete certainty about occupied or empty state, but without having the numerical instabilities that 0/1 probabilities cause.

4.1.5 Decision Rule

The ultimate goal of the occupancy grid is to determine the presence of obstacles on the road. Therefore, a decision must be made to translate a probabilistic formulation into a logical state. The state of each cell, s_{ij}, can be one of the following: `empty`, `occupied` or `unknown`. The following decision rule is applied:

$$s_{ij} = \begin{cases} \texttt{empty} & ; \quad p(m_{ij}) < 0.5 - \epsilon \\ \texttt{unknown} & ; \quad |p(m_{ij}) - 0.5| \leq \epsilon \\ \texttt{occupied} & ; \quad p(m_{ij}) > 0.5 + \epsilon \end{cases} \tag{4.14}$$

A small margin around 0.5 of size 2ϵ is chosen for two reasons. First, because probabilities will rarely be exactly 0.5, which is the definition of `unknown`. Second, a cell is classified as `empty` or `occupied` only if the probability is high enough, so there is a reasonably high confidence. In this case, we choose $\epsilon = 0.2$. This will help removing false positives, since normally several sensor measurements will have to be integrated for a cell to have a high enough probability of occupancy.

4.2 Dempster-Shafer Theory of Evidence

A potential drawback of the Bayesian inference theory is that it is not capable of explicitly representing ignorance (absence of information), and it does not handle conflicting information well. To solve this, a generalization was proposed by Dempster [71] and later extended by Shafer [72] , who gave rise to the Dempster-Shafer theory of evidence. First, some basic definitions are presented in order to summarize the Dempster-Shafer theory. Afterwards, its application to this particular problem is developed.

4.2.1 Definitions

The following concepts comprise the core of the Dempster-Shafer theory [45]:

- **Frame of discernment** (FOD), Θ. It is a finite set of mutually exclusive and exhaustive propositions X_i. It can be viewed as the set of all possible K outcomes or states of a random variable.

$$\Theta = \{X_1, X_2, \ldots, X_K\} \tag{4.15}$$

- **Basic probability assignment (BPA) function**. It is the equivalent of the probability density function in the classic probability theory, and it can be assigned to any subset of the frame of discernment. In our case it is directly related to the probability of occupancy of the map as well as the sensor model. It is defined as:

$$m : 2^{\Theta} \rightarrow [0, 1] \tag{4.16}$$

  where 2^{Θ} is the power set of Θ, i.e. the set of all possible subsets of Θ.

  For every subset $A \in 2^{\Theta}$, the BPA function $m(A)$, also called *A's basic belief number* or the *mass of A* [73], must fulfill the following conditions:

$$\sum_{A \in 2^{\Theta}} m(A) = 1 \quad ; \quad m(\emptyset) = 0 \tag{4.17}$$

- **Belief** of a subset $X \in 2^{\Theta}$. It accounts for all the evidence that supports the proposition X. It represents the degree to which X is believed to be true [74].

$$Bel(X) = \sum_{A \subseteq X} m(A) \tag{4.18}$$

- **Plausability** of a subset $X \in 2^{\Theta}$. It considers the evidence that does not provide knowledge about the preposition X. It therefore represents the degree to which X is believed not to be false [74].

$$Pl(X) = 1 - \sum_{A \cap X = \emptyset; A \in 2^{\Theta}} m(A) \tag{4.19}$$

  It can be shown that these two functions $-Bel(X)$ and $Pl(X)-$ define the lower and upper bounds for the probability of X (in the classical sense) [75]:

$$Bel(X) \leq P(X) \leq Pl(X) \tag{4.20}$$

- **Dempster-Shafer combination formula**. Two sources of information $m_1(X)$ and $m_2(X)$ about the proposition $X \in 2^{\Theta}$, are combined into a fused belief function $m_{12}(X)$ through the $\oplus$ operator, according to Equation 4.21:

$$m_{12}(X) = (m_1 \oplus m_2)(X) = \begin{cases} \frac{(m_1 \textcircled{\cap} m_2)(X)}{1 - (m_1 \textcircled{\cap} m_2)(\emptyset)} & X \neq \emptyset \\ 0 & X = \emptyset \end{cases} \tag{4.21}$$

where $(m_1 \textcircled{\cap} m_2)(X)$ is the *conjunctive rule*:

$$(m_1 \textcircled{\cap} m_2)(X) = \sum_{A,B \in 2^\Theta | A \cap B = X} m_1(A) \cdot m_2(B) \tag{4.22}$$

The denominator in Equation 4.21 is a normalization factor. The element $K = (m_1 \textcircled{\cap} m_2)(\emptyset)$ is a measure of the *conflict* between the sources of information. The *conflict factor*, Con, is defined as follows:

$$Con = \log\left(\frac{1}{1-K}\right) \tag{4.23}$$

Given that $0 \leq K \leq 1$ (since K is a mass function), it is clear that $0 < Con < \infty$. The larger the value, the more conflicting the two sources of information are.

It is important to understand the effect of the normalization factor. Since $m(\emptyset)$ must be zero, its value must be **redistributed** among the other three events (`empty`, `occupied` and `unknown`), which is done by dividing by $1 - K$. Interestingly, some of the mass will be assigned to the `unknown` state, represented by $m(\Theta)$.

- **Probability**. It is possible to compute a classical probability measure from the mass functions using the *pignistic transformation*, $BetP$, as shown by Smets [76]:

$$BetP(A) = \sum_{X \in 2^\Theta} m(X) \cdot \frac{|A \cap X|}{|X|} \tag{4.24}$$

where $|X|$ is the cardinal of the subset X.

4.2.2 Formulation for the Occupancy Grid Mapping Problem

In the occupancy grid mapping problem, every cell is assumed to have a binary state: `empty` (E) or `occupied` (O), as was done in the Bayesian framework. For every cell, the frame of discernment, Θ, and its power set, 2^Θ, are defined as follows:

$$\Theta = \{E, O\} \quad ; \quad 2^\Theta = \{\emptyset, E, O, \{E, O\}\} \tag{4.25}$$

The sets $\emptyset$ and $\{E, O\}$ represent the *null* set and the *ignorance* set, respectively. The former will always have a mass function equal to zero ($m(\emptyset) = 0$), since a cell must be in any of the states defined in the FOD. The latter set is especially interesting, since it represents that the status is either E *or* O —in other words, `unknown`. As can be seen, it allows for an **explicit representation of ignorance** about the state of a cell. In the Bayesian framework, one can model this situation by assigning $p(m_{ij}) = 0.5$, but there is no measure of how confident we are about that statement. In the Dempster-Shafer framework, on the other hand, it is possible to express that the cell *might* have an `unknown` state, with a certain probability.

The implementation of the Dempster-Shafer framework now requires to maintain two grids, in order to represent the mass functions $m_{ij}(E)$, $m_{ij}(O)$, associated to the `empty`

(E) and `occupied` (O) states of a cell, respectively. Even though the power set 2^{Θ} is composed of four propositions, it can be seen that only two BPAs, $m(E)$ and $m(O)$, are sufficient to fully describe it, taking into account Equation 4.17.

$$\left.\begin{array}{l}\sum_{A\in 2^{\Theta}} m(A) = 1 \\ m(\emptyset) = 0\end{array}\right\} \implies m(\{E,O\}) = 1 - m(E) - m(O) \tag{4.26}$$

4.2.3 Grid Initialization

During the initialization, the fact that no knowledge has been yet incorporated by sensor measurements is represented by assigning a zero mass distribution to both the empty and occupied states.

$$m_{ij}(E) = m_{ij}(O) = 0 \qquad \forall i,j \tag{4.27}$$

It is worth noting that this implies that $m_{ij}(\{E,O\}) = 1$: it is explicitly stated that there is complete ignorance about the state of the cells —*"I only know that I know nothing"*.

4.2.4 Grid Update

When updating an occupancy grid using information from a single sensor, the current map m^t, and the inverse sensor model, m^s, will be used to update the BPA for the `empty` (E) and `occupied` (O) states. The Dempster-Shafer combination formula (see Equation 4.21) is applied to this particular problem:[1]

$$m^{t+1}(O) = \frac{m^t(O)m^s(O) + m^t(O)m^s(\{O,E\}) + m^t(\{O,E\})m^s(O)}{1 - m^t(E)m^s(O) - m^t(O)m^s(E)} \tag{4.28}$$

$$m^{t+1}(E) = \frac{m^t(E)m^s(E) + m^t(E)m^s(\{O,E\}) + m^t(\{O,E\})m^s(E)}{1 - m^t(E)m^s(O) - m^t(O)m^s(E)} \tag{4.29}$$

Similar to the Bayesian theory, an inverse sensor model is required in order to update the grid. In this case, the probabilities of a cell being empty or occupied, $m^s(E)$ and $m^s(O)$ can be modelled independently unlike the Bayesian theory, in which a single probability density function is specified. The relation to the Bayesian sensor model will be explained in Section 5.5. Note that $m^s(\{O,E\})$ can be obtained through Equation 4.17, which yields:

$$m^s(\{O,E\}) = 1 - m^s(O) - m^s(E) \tag{4.30}$$

4.2.5 Criticism and Extensions to the Dempster-Shafer Theory

Despite the advantages of the Dempster-Shafer framework for explicitly modelling uncertainty and conflicting information, many authors have criticized the theory claiming that it might sometimes give totally counterintuitive results. This issue was first pointed out by Zadeh [77], which showed that the normalization factor $1 - K$ in Equation 4.21, which is a measure of how much the two sources of information are in conflict, may give

[1] This equation is applied to every cell in the grid. The subindexes ij have been omitted for clarity.

rise to a large probability for events that do not actually have evidential support. He presents an example that has become very popular in the literature. Consider the case of disease diagnosis, where there exist three possible diseases: $\Theta = \{A, B, C\}$. Two doctors give an opinion about what the most likely disease might be, as presented in Table 4.1.

	$m(A)$	$m(B)$	$m(C)$	$m(X) : X \subseteq \Theta, X \neq A, B, C$
Doctor 1	0.99	0	0.01	0
Doctor 2	0	0.99	0.01	0

Table 4.1: Zadeh's example: doctors' opinion.

This is a clear case of highly **conflicting information**. Nevertheless, the doctors at least agree that it is most likely not disease C. However, if the classical Dempster-Shafer rule of combination (Equation 4.21) is applied to fuse these two sources of information, the result is the following:

$$m(A) = 0,\ m(B) = 0,\ m(C) = 1 \tag{4.31}$$

which is clearly in contradiction with the doctors' information.

After this, there has been a great interest in the research community to develop alternative combination rules that try to avoid this issue [78–82]. With the exception of Yager's rule [78], which will be further analyzed in Chapter 6, alternative combination rules tend to be an *ad-hoc* solution to the particular problem at hand. Instead of analyzing each of them, we will study the current problem and reason about whether the current classical Dempster-Shafer rule is valid or not.

A popular extension of the Dempster-Shafer theory is the Transferable Belief Model (TBM), introduced by Smets [83]. In this case, the combination rule does not include the normalization term; instead, the conflicting mass is associated to the null set: $m(\emptyset) = K$. This totally contradicts the classical Dempster-Shafer theory which assumes $m(\emptyset) = 0$. The motivation for this new framework is the so-called *open-world* assumption: the events in the frame of discernment are *not exhaustive*. For instance, when tossing a coin, the possible results are not only heads or tails, but it is also (remotely) possible that the coin ends up standing on its edge.

Arguments supporting the classical Dempster-Shafer theory can also be found. An interesting article by Haenni [84] argues that the problem does not rely on the Dempster-Shafer framework, but on Zadeh's example. Indeed, there is an underlying assumption in the Dempster-Shafer theory: the sources of information are *reliable*. In other words, if Doctor 1 claims that $m(B) = 0$, it means that with total certainty that disease B is not possible. The same reasoning can be applied to Doctor 2 about disease A. Therefore, the only alternative is of course disease C, which logically ends up with probability 1 given that the two other options have been discarded.

After this brief discussion, we reach some conclusions applicable to this project. First, we realize that our specific application —occupancy grid mapping— has two clear exclusive and exhaustive events in the frame of discernment: `empty` and `occupied`. Therefore, there is no need to take into account the *open-world* assumption and the issues associated

Figure 4.3: Dempster-Shafer theory: different decision rules, from Moras *et al.* [60]. Left: $Pl(O) > 0.5$. Middle: $Bel(O) > 0.5$. Right: $P(O) > 0.5$.

with it. To account for possible high-conflict fusion situations, our approach will be based on the **conflict analysis**.

4.2.6 Decision

Similarly to the Bayesian case, a decision rule must be defined in order to determine the state of a cell. It is easier to do in this case since the empty and occupied BPAs are decoupled. Some authors in the literature make use of the belief (Bel) and plausability (Pl) definitions (see Equations 4.18 and 4.19) to make the decision rule. For instance, Daniel and Lauffenburger [85] propose these definitions to determine the state of a cell, s:

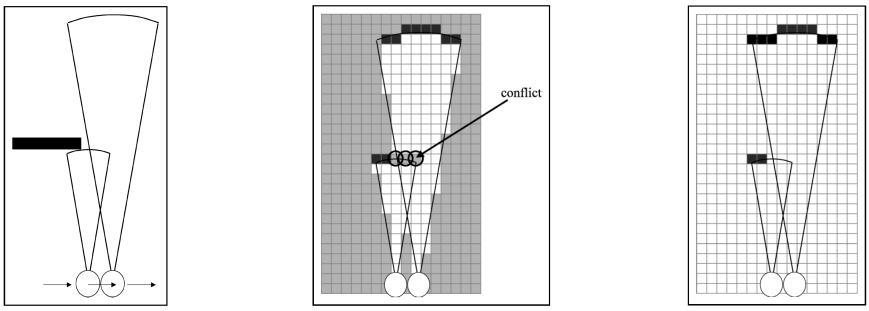
$$s = \arg\max_k Bel(X_k) \tag{4.32}$$

$$s = \arg\max_k Pl(X_k) \tag{4.33}$$

for any $X_k \in 2^\Theta$. It is interesting to briefly analyze these alternatives. Moras *et al.* [60] presented an analysis of the accuracy of the resulting occupancy grid, as shown in Figure 4.3. Remembering that $Bel(X) \leq P(X) \leq Pl(X)$, these results can be explained. The plausability Pl is the upper limit of the probability, and will include the `unknown` regions since it is possible that they are actually occupied; we just do not know. The belief Bel and probability P grids are pretty similar. However, Bel is more refined since it does not include the `unknown` regions in the limit of `empty` and `occupied` space, which usually happens due to the sensor noise.

Furthermore, it is also possible to use the *pignistic transformation* to convert from mass functions to classical probabilities. In this case, we are interested in the probability of occupancy:

$$P(O) = \sum_{X \in 2^\Theta} m(X) \cdot \frac{|O \cap X|}{|X|} = m(O) + \frac{m(\{E, O\})}{2} \tag{4.34}$$



(a) Environment (b) Inverse sensor model (c) Forward sensor model

Figure 4.4: Forward and inverse sensor models: comparison. In this example, an occupancy grid is built using sonar data. In 4.4a, the real environment. In 4.4b, the inverse sensor model is used, whereas in 4.4c a forward sensor model is applied. In the first case, a conflict is generated between the two measurements. However, in the latter case an optimal, joint solution for the map can be found. Images from Thrun *et al.* [70].

Once we have a probability value, the same criteria as the one used in the Bayesian framework can be applied.

In order to have an even comparison between the Bayesian and Dempster-Shafer approaches, we will use the probability function P, obtained through the pignistic transformation, to perform the decision rule. Bel could be used to improve the accuracy of the detections.

4.3 Forward and Inverse Sensor Models

When performing map building based on the occupancy grid framework it is possible to work with forward or inverse sensor models. In this section a brief comparison of both approaches is presented. Afterwards, we motivate our choice of the **inverse sensor model** approach for this project.

The main advantage of forward sensor models, as discussed by Thrun *et al.* [70], is that it allows for a more accurate sensor modelling. Indeed, the model represents a direct causal relationship from the map to the corresponding measurement generated on the sensor. In addition, forward sensor models do not rely on the assumption that the cells in the occupancy grid are independent of each other. This allows solving situations with conflicting sensor information, as depicted in Figure 4.4.

On the other hand, forward sensor models require the map $\mathbf{m}$ to be known or estimated. Therefore the estimation problem is translated into a Maximum A Posteriori (MAP) problem, where the most likely map is to be found. As mentioned in the introduction, it is computationally intractable to analyze all the possible combinations. There exist some proposals in the literature that try to reduce the computational load. Thrun *et al.* [70] propose an Expectation Maximization algorithm to find a locally optimal solution, although it is not able to run on real-time and requires several runs to find a satisfactory solution. More recently, Merali and Barfoot [86] propose a more efficient method known as Gibbs sampling, a form of Markov Chain Monte Carlo (MCMC). They draw a large number of

samples from the distribution in order to capture it accurately. Even though it is not able to run in real time, it is a computationally feasible approach. However, as discussed by Dhiman *et al.* [87], this solution is still slow and prone to getting stuck in local minima. They propose instead to use modern MAP inference methods over graphical models. Their experiments show a more accurate and efficient map building.

The choice of using inverse sensor models for this project is based on their simplicity and low computational load. The goal is not to create an extremely accurate map; rather, a framework over which to fuse sensor information and detect roughly the position of obstacles on the road is the priority. In addition, the Dempster-Shafer theory works with inverse sensor models, so we must use the same models on the Bayesian framework if we want to perform an even comparison between these two techniques. Furthermore, given the lack of in-depth technical information about the sensors and the complexity to derive a good model (especially in the case of the radar), we would not be able to leverage the advantages of forward sensor models by creating a more complex model. For these reasons, it is more appropriate to use inverse sensor models for this project.

4.4 Grid Update Algorithm

An occupancy grid is built and maintained for every sensor. At every iteration, with cycle time $\Delta t = 25$ ms, each grid will be updated following a **predict-update** loop using sensor information according to the block diagram presented in Figure 4.5:

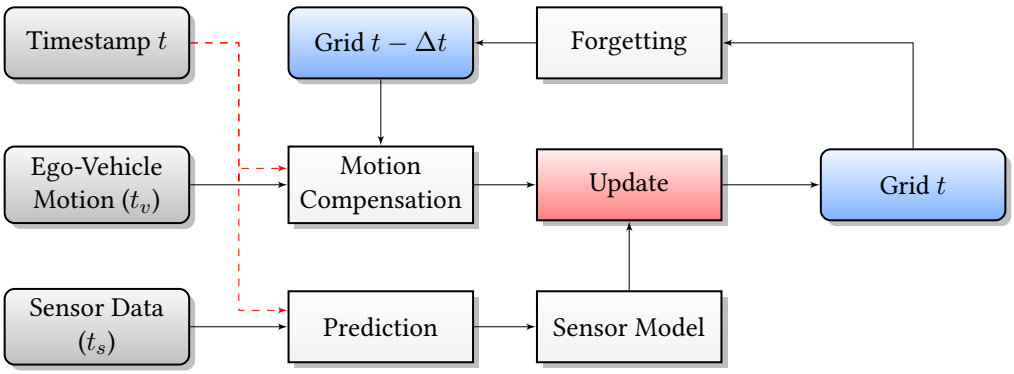


Figure 4.5: Overview of the occupancy grid update algorithm.

1. First, it is assumed that the grid is initialized at time $t = 0$ following the criteria that was described in Section 4.1.3 and 4.2.3 for the Bayesian and Dempster-Shafer cases, respectively. The following steps are performed for $t > 0$.

2. The **predict** phase is performed in two steps. First, a *forgetting factor* is applied to account for information obsolescence and to handle dynamic environments more easily, as described in Section 4.5. Next, the grid is rotated and translated to compensate for the ego-vehicle motion, with the aim of keeping the vehicle in the same position (see Section 4.6). In addition, we account for sensor time synchronization by predicting its output at a time t, as presented in Section 6.5. The ego-vehicle and sensor data are received at times $t_v \leq t$ and $t_s \leq t$, respectively.

3. Next, the grid is **updated** with new sensor information. This is done by applying the update formula from either the Bayesian (see Section 4.1) or Dempster-Shafer (see Section 4.2.4) framework, fusing the old grid with the information given by the **inverse sensor model**. We explain how to compute this model for every sensor in Chapter 5.

4. The output is an updated grid at the current time t, which will be fed back to the predict-update loop in the next iteration.

4.5 Forgetting Factor

As discussed in the Chapter 2, the occupancy grid framework assumes a static environment. Extensions such as the Bayesian Occupancy Filter, which increases the dimensionality of the grid to four, allow to model the velocity of each cell, at the cost of a much higher computational cost.

For this project, we propose an alternative to relax the static world assumption while maintaining a reasonable computational load. We introduce an exponential **forgetting or decay factor** for every cell in the grid whose occupancy probability is different than 0.5. The probability of occupancy will tend towards the `unknown` state as time passes by unless new sensor measurements update the state. Another advantage of the decay factor is that it will progressively remove outliers from the sensor data. The solution is implemented as follows:

- **Bayes grid**. In this case, the `unknown` state is represented with $p(\mathbf{m}) = 0.5$, so the cell state should exponentially tend towards that value. Therefore:

$$p(m_{ij}) \leftarrow (p(m_{ij}) - 0.5) \cdot \mathrm{e}^{-\frac{\Delta t}{\tau}} + 0.5 \tag{4.35}$$

  where Δt is the fixed update interval in the sensor fusion algorithm, and τ [s^{-1}] is the time constant that specifies how fast the probability decays to 0.5. Note that this equation is valid for any value of $p(m_{ij}) \in [0, 1]$.

- **Dempster-Shafer grid**. For the Dempster-Shafer grid, the goal is to make the BPA corresponding to the `empty` and `occupied` state decay exponentially towards 0, which will imply that the `unknown` state will grow towards one. Therefore:

$$m^e_{ij} \leftarrow m^e_{ij} \cdot \mathrm{e}^{-\frac{\Delta t}{\tau}} \tag{4.36}$$

$$m^o_{ij} \leftarrow m^o_{ij} \cdot \mathrm{e}^{-\frac{\Delta t}{\tau}} \tag{4.37}$$

In this project, the cycle time of the sensor fusion module is $\Delta_t = 25$ ms, small enough so that the vehicle has enough time to apply changes to its trajectory. In addition, we choose a value $\tau = 1$ s, similar to what it is done in the literature such as Moras *et al.* [60] for example, who take $\tau = 1.3$ s. It is a trade-off between the grid's ability to quickly adapt to the environment and to preserve old information.

4.6 Ego-Vehicle Motion Compensation

To create a relatively high-resolution occupancy grid containing a complete map of all the places the vehicle has moved into is computationally unfeasible, especially regarding memory consumption. There exist some memory-efficient implementations in the literature based on Quad-Trees [88][89], that allow for larger, multi-resolution maps. However, it is not the goal of this project to generate a complete map of the environment. Instead, a local, detailed map is enough for the purpose of local obstacle avoidance.

We propose to have an **ego-centered occupancy grid**, i.e. the vehicle position is fixed and the contents of the grid move relative to the vehicle. This is performed in two steps:

1. Ego-motion estimation.
2. Grid transformation.

4.6.1 Conventions

First, let us introduce some coordinate frame conventions. We define a local vehicle pose $\mathbf{x}_v = \begin{pmatrix} x_v & y_v & \theta_v \end{pmatrix}^\mathsf{T}$ which represents the vehicle pose with respect to a `world` frame, defined at the bottom-left corner of the grid, as shown in Figure 4.6. The position of the `world` frame is completely arbitrary, for an easier visualization. It is important to note that this pose does not have to lie on a given cell of the grid. In other words, the values of x_v, y_v and θ_v are continuous variables, not discrete.

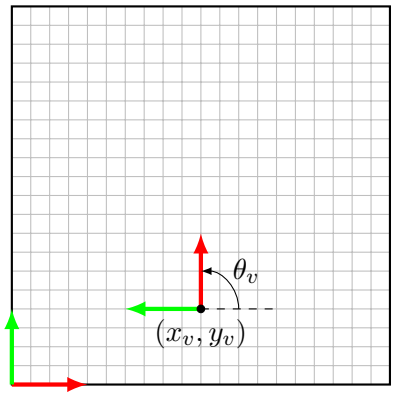


Figure 4.6: Local vehicle pose definition: $\mathbf{x}_v = \begin{pmatrix} x_v & y_v & \theta_v \end{pmatrix}^\mathsf{T}$. The `world` reference frame is located at the bottom-left corner of the grid.

4.6.2 Ego-Vehicle Motion Compensation

The first step in the process is to estimate the ego-vehicle motion. This operation is performed using the internal proprioceptive sensors of the car: wheel odometry and an inertial measurement unit (IMU). In this project we are provided with a software module that takes these measurements and estimates the vehicle motion $\delta\mathbf{x}_v^V = \begin{pmatrix} \delta x_v & \delta y_v & \delta\theta_v \end{pmatrix}^\mathsf{T}$, in local `vehicle` coordinates, between two time instants, t_1 and t_2, with $t_2 \geq t_1$. At every time step, we call this module to obtain the displacement of the vehicle with respect to

the previous iteration, with which the pose is updated. Then, to update the local vehicle pose $\mathbf{x}_v$, we transform the displacement $\delta\mathbf{x}_v^V$ from local to global coordinate frames with respect to the `world` frame to obtain $\delta\mathbf{x}_v^W$. Finally, the updated pose is:

$$\mathbf{x}_v \leftarrow \mathbf{x}_v + \delta\mathbf{x}_v^W \tag{4.38}$$

If the vehicle pose has changed too much from its origin, a grid transformation is required in order to keep the vehicle in the same position. This is implemented as a translation and rotation of the grid in opposite direction as the vehicle motion. This process is described in more detail in Appendix C, Section C.1.

Chapter 5

Sensor Modelling

In this chapter we present the sensors used in this project as well as the developed inverse sensor models that are used to incorporate the sensor measurements into the occupancy grids.

5.1 Velodyne Lidar

The Velodyne lidar is a 3D laser scanner containing 64 laser transmitters and receivers, separated vertically, giving 64 elevation angles or layers. A receiver registers the reflected signal of the laser beam and computes the distance to the target by time-of-flight method. The sensor is mounted on top of a spinning platform that allows it to cover a 360º view of the surroundings of the car. It has a very high angular resolution and accuracy, as can be observed in the specifications (see Table D.1). Such a high quality implies a higher cost, which is normally above the cost limit of a normal production car. Therefore, we only use this sensor for verification purposes, by means of creating a ground truth occupancy grid to which the others can be compared.

Input data

The raw data received from the sensor is a **point cloud**: a set of N_V points in 3D space, referred to the local sensor coordinate frame:

$$\mathcal{Z}^V = \{\mathbf{z}_1^V, \mathbf{z}_2^V, \ldots, \mathbf{z}_{N_V}^V\} \quad ; \quad \mathbf{z}_i^V = \begin{pmatrix} x_i & y_i & z_i \end{pmatrix}^\mathsf{T} \tag{5.1}$$

The goal is to transfer this information into an occupancy grid. However, we notice that the raw data contains detections from the ground, which is not considered as an obstacle. Therefore, we first preprocess the input cloud by removing the ground points. Finally, we build an occupancy grid with the remaining points, belonging only to potential obstacles.

5.1.1 Ground Removal

The topic of ground removal in point clouds is still an open problem under active research. A naïve approach would be to remove those points whose z component (height) is smaller than a given threshold. This technique is however highly dependent on two

assumptions. First, that the sensor extrinsic calibration is very accurate. Second, and most importantly, that the road is perfectly flat in the region covered by the sensor. This might be accurate for indoor environments, but not for roads.

A more robust technique is plane fitting using RANSAC, originally proposed by Fischler and Bolles [90]. It is a general framework used to fit a model to a set of data points containing outliers. The method loops over a large number of iterations, which depends on the expected proportion of outliers in the data. At every iteration, a minimum set of points that allow to fit a model are chosen at random, and a model is fitted to those points. Then, the distance from the rest of the points to the model is computed; the closest ones are considered to belong to the model (inliers). The process is repeated and the best model (i.e. the one with largest number of inliers) is kept. The main advantage of this technique is its robustness against outliers, so it is appropriate for the use with real-world data.

In this particular application, the input is a point cloud and the model to be fit is a plane, since the ground can be assumed to be approximately planar in small regions of space. Hence, we divide the coverage region of the sensor (around 100×100 m) into small regions of 1×1 m and apply the RANSAC algorithm to each of them. The model to be fitted is therefore a plane, following the plane equation:

$$ax + by + cz + d = 0 \tag{5.2}$$

A plane can be uniquely represented by a minimum set of three points. Given three points $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$, the coefficients are computed as follows:

1. The normal vector of the plane is computed as the cross product of two vectors lying on the plane. For example, let $\mathbf{v}_1 = \mathbf{P}_3 - \mathbf{P}_1$, $\mathbf{v}_2 = \mathbf{P}_3 - \mathbf{P}_2$. Then:

$$\mathbf{n} = \mathbf{v}_1 \times \mathbf{v}_2 \quad ; \quad \hat{\mathbf{n}} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \frac{\mathbf{n}}{||\mathbf{n}||} \tag{5.3}$$

2. Finally, the coefficient d can be easily obtained by replacing in Equation 5.2 the obtained coefficients and whichever of the three selected points.

The signed Euclidean distance from any point $\mathbf{P}$ to a plane $\mathcal{P} = \begin{pmatrix} a & b & c & d \end{pmatrix}^\mathsf{T}$ is:

$$d(\mathbf{P}, \mathcal{P}) = \frac{\begin{pmatrix} \mathbf{P}^\mathsf{T} & 1 \end{pmatrix} \cdot \mathcal{P}}{||\mathcal{P}||} \tag{5.4}$$

Having formulated the mathematical model, the RANSAC algorithm can be applied for plane extraction. A pseudocode is presented in Algorithm 1. The result of applying the algorithm to experimental data is presented in Figure 5.1. It can be seen that it removes most of the road, while the main obstacles such as cars, guardrails and road signs remain. The main drawback is a high computational time, but this is not an issue since this operation is performed offline and the map is static.

Algorithm 1 Ground extraction using RANSAC

Input: `cloud` ▷ 3D point cloud
Output: `mask` ▷ Binary mask: ground points set to `true`

$n_{\max} \leftarrow 0$ ▷ Maximum number of inliers so far
for $i = 1 : N_{\text{iters}}$ **do**
  $P_{\text{test}} \leftarrow$ SampleThreePoints(`cloud`)
  $\mathcal{P} \leftarrow$ FitPlane(P_{test})
  $d \leftarrow$ ComputeDistancePointPlane(`cloud`, $\mathcal{P}$)
  $m \leftarrow |d| < \epsilon_d$ ▷ Binary mask: points closer than ϵ_d to the plane
  $n \leftarrow$ sum(m) ▷ Number of inliers (ground points)
  if n $> n_{\max}$ **then**
    $n_{\max} \leftarrow n$
    `mask` $\leftarrow m$
  end if
end for

Implementation remarks

There are several aspects that should be taken into account for the implementation:

- Only planes that are approximately parallel to the XY-plane are considered as ground. We select a maximum slope of $\alpha_{\max} = 20°$. This avoids removing vertical planes such as road signs or parts of vehicles. The angle α between the plane the XY-plane, that defines the slope, is computed as follows:

$$\alpha = \cos^{-1}\left(\frac{\hat{\mathbf{n}}_z}{\sqrt{\hat{\mathbf{n}}_x^2 + \hat{\mathbf{n}}_y^2 + \hat{\mathbf{n}}_z^2}}\right) = \cos^{-1}\left(\hat{\mathbf{n}}_z\right) \tag{5.5}$$

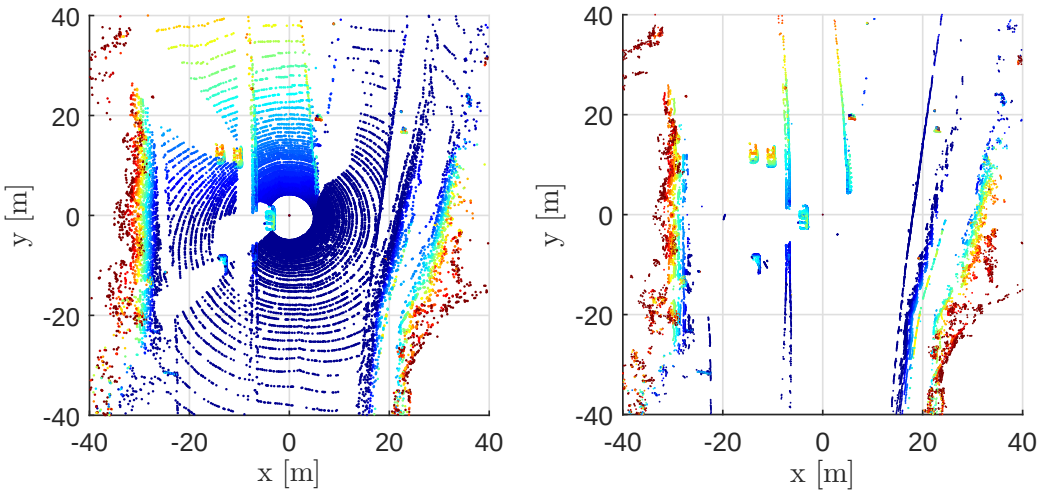
- The three random points that define the test plane are chosen so that one of them is always the **lowest point** (minimum z-component). This ensures that planar surfaces from objects (e.g. the top of a car) are not wrongly classified as road.

5.1.2 Inverse Sensor Model

After removing the points belonging to the road, the result is a point cloud whose points correspond to potential obstacles only. We propose the use of an **ideal** inverse sensor model, given the high accuracy of the sensor:

$$p(m_{ij}|\mathbf{z}_k) = \begin{cases} 1 & \text{if } \mathbf{z}_k \subset C_{ij} \\ 0 & \text{otherwise} \end{cases} \quad ; \quad 1 \leq i \leq N_H, 1 \leq j \leq N_W, 1 \leq k \leq N_f \tag{5.6}$$

where the operation $\mathbf{z}_k \subset C_{ij}$ returns 1 whenever the x and y component of $\mathbf{z}_k$ are contained inside the grid cell C_{ij}. N_f is the number of points of the filtered point cloud (after ground removal). The point cloud is assumed to be already transformed into the



(a) Raw input

(b) Output

Figure 5.1: Ground removal from Velodyne raw data. In the example, the ego-vehicle, at $(x, y) = (0, 0)$ is driving on a highway. One car is detected to its left, and three other vehicles appear on the opposite lane. $N_{\text{iters}} = 5000$, $\epsilon_d = 0.1$.

`world` coordinate frame. The height information is not taken into account: the points are simply projected onto the occupancy grid. The result is a binary occupancy grid, as shown in Figure 5.2, against which we can compare the other grids later on.

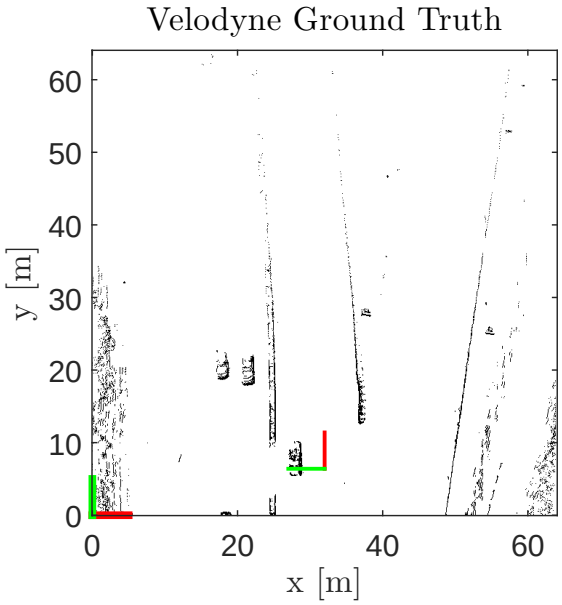


Figure 5.2: Ground truth grid from the Velodyne sensor.

5.2 Lidar

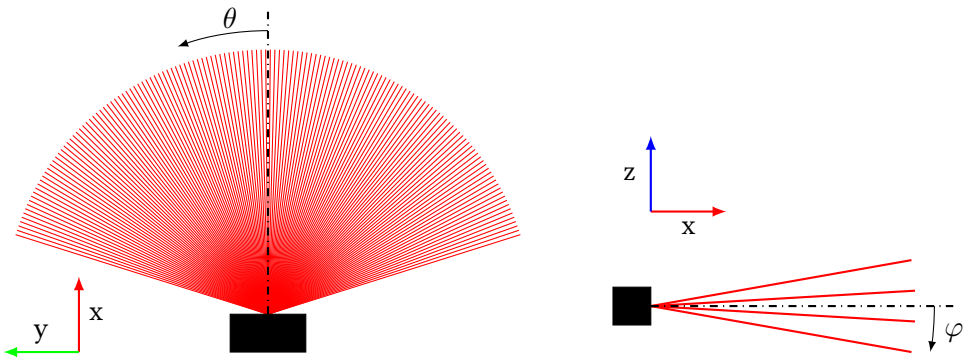
In addition to the Velodyne lidar, we use a common lidar for automotive applications in this project, much smaller, easier to handle and cheaper. The main difference with respect to the Velodyne is that there is no moving platform; instead, an internal mirror rotates to point a single laser beam to a given direction, sweeping within the designed field of view. This sensor has worse specifications than the Velodyne, as can be observed in Table D.2, but it is still reasonably accurate. It is also a 3D lidar, containing four layers per scan, and it has multi-echo technology. In particular, a maximum of three measurements per angle and layer can be obtained, making the sensor much more robust especially under adverse weather conditions, such as rain, snow or dust.

Input data

At every time step, the following raw data measurements are received:

$$\mathcal{Z}^L = \{\mathbf{z}_1^L, \mathbf{z}_2^L, \dots, \mathbf{z}_{N_L}^L\} \quad ; \quad \mathbf{z}_i^L = \begin{pmatrix} r_i & \theta_i & \varphi_i \end{pmatrix}^\mathsf{T} \tag{5.7}$$

where N_L is the number of laser detections, r_i is the distance at which the ith laser beam detects an object, θ_i is the corresponding azimuth angle, and φ_i is the elevation angle, directly related to the layer number, as can be seen in Figure 5.3. To mimic the physical principle as much as possible, we will develop sensor models in **polar coordinates**.



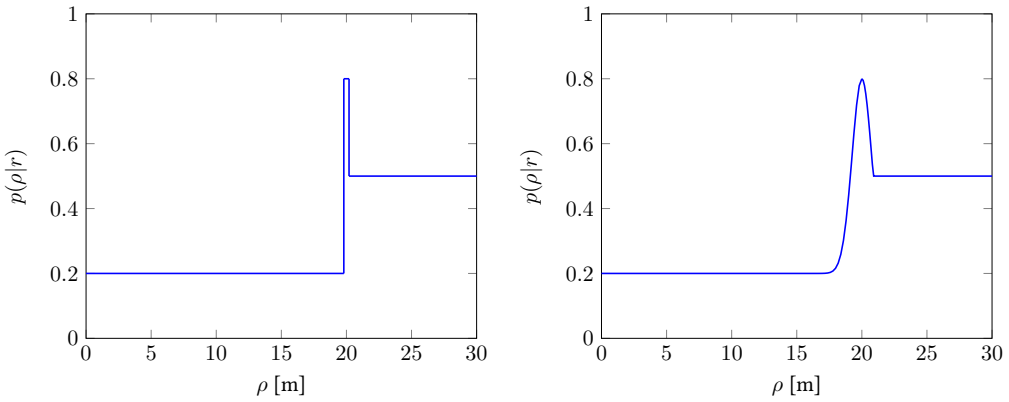
(a) Top view (b) Lateral view

Figure 5.3: Front lidar: schematic representation.

5.2.1 2D Lidar Sensor Model

It is interesting to first study the 2D lidar model, which is the most common in the literature. We will then propose a new 3D lidar model based on the former.

Given the high angular resolution of the laser scanner, the uncertainty in the angular dimension is negligible compared to the one in the range dimension. Therefore, a simple one-dimensional inverse sensor model in the range dimension is proposed for every beam.



(a) Ideal sensor model

(b) Noisy sensor model

Figure 5.4: 2D lidar sensor model: example for a single laser beam. The simulated target is detected at $r = 20$ m, with a sensor noise of $\sigma = 0.75$ m.

For the case of an ideal, noise-free sensor, the sensor model is defined as follows:

$$p(\rho|r) = \begin{cases} P_{\text{min}} & ; \quad 0 \leq \rho < r - \Delta\rho/2 \\ P_{\text{max}} & ; \quad r - \Delta\rho/2 \leq \rho \leq r + \Delta\rho/2 \\ 0.5 & ; \quad \rho > r + \Delta\rho/2 \end{cases} \tag{5.8}$$

where ρ is the set of possible ranges for a given azimuth angle, discretized with $\Delta\rho$ resolution, and r is the actual measurement returned by the laser beam. A plot of this function is presented in Figure 5.4a. P_{min} and P_{max} limit the minimum and maximum probability, as described in Section 5.4. This model mimics the working principle of the lidar:

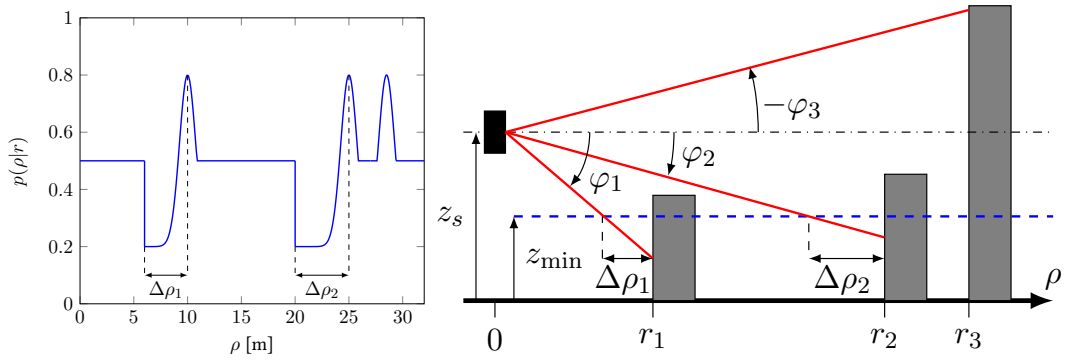
1. The region $0 \leq \rho < r$ is assumed to be `empty` space; otherwise, the lidar would have hit the object at a closer distance.
2. It is most likely `occupied` exactly at $\rho = r$, where the target has been detected.
3. Finally, since the laser beam cannot travel passed the object, additional information about the region $\rho > r$ cannot be obtained. Therefore, the probability is 0.5, representing the `unknown` state.

Nevertheless, real sensors are noisy. In our case, there is a non-negligible standard deviation of around 0.1 m in the range detected by the sensor. This can be modelled by convolving the ideal sensor model with a zero-mean Gaussian kernel with a variance equals to the sensor noise [91]. We approximate it as follows (see Figure 5.4b):

$$\begin{cases} f_1(\rho) = \eta\mathcal{N}(\rho; r, \sigma_r) & \rho \in [0, r] \\ f_2(\rho) = 0.5 & \rho > r \end{cases} \implies p(\rho|r) = \max\{f_1(\rho), f_2(\rho)\} \tag{5.9}$$

where $\mathcal{N}(x; \mu, \sigma)$ is the one-dimensional Gaussian distribution and η is a normalization factor so that we have an actual probability distribution:

$$\mathcal{N}(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{5.10}$$



(a) Inverse sensor model

(b) Physical placement of objects and sensor

Figure 5.5: Multi-layer lidar inverse sensor model: example. Three targets are placed at distances $r_1 = 10$ m, $r_2 = 20$ m and $r_3 = 25$ m, respectively.

5.2.2 3D Sensor Model

An extension to the previous model must be developed for our 3D lidar. Very few authors in the literature tackle this issue. Baig *et al.* [92] use a four-layer lidar and create one grid for each layer. The grids are merged afterwards through the Linear Opinion Pool technique. The main drawback of this method is that the upper layers override with empty space occupied areas that were detected with the lower layers. Yu *et al.* [93] propose a more careful modelling applied to a Velodyne (64 layers). They base their model on the *minimum commitment principle*: they do not set to empty the cells that go from the sensor origin to the target. They refine the model by setting empty space from the target, going backwards to the origin. We propose a more refined model following that approach.

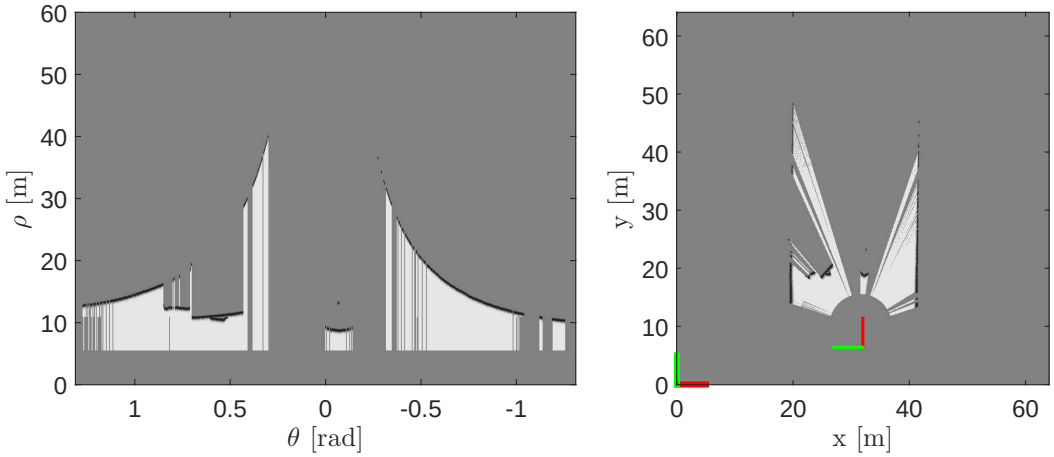
This model is motivated with an example, shown in Figure 5.5. In Figure 5.5b, the lidar detects three objects, each with a laser beam in a different layer. If traditional approaches were followed, the first and second obstacles, at $\rho = r_1, r_2$, would not be detected in the final occupancy grid, since the detections would be overridden by the empty space corresponding to the third obstacle. The most conservative solution would then be to set to unknown all the space around the target. This solution can be improved considering that there exist two regions, $\Delta\rho_1$ and $\Delta\rho_2$, that *must* be empty, assuming that the objects that we aim to detect have a minimum height $z_{\min}$ and the sensor is located at a height z_s above the ground. The resulting sensor model is depicted in Figure 5.5a. The three targets are properly detected and only a small region before them is set to empty space.

The free-space length $\Delta\rho_i$ is computed as follows, for each layer i:

$$\Delta\rho_i = r_i - \frac{z_s - z_{\min}}{\tan\varphi_i} \quad ; \quad i = 1, 2 \tag{5.11}$$

This only applies to the lower layers, whose laser beams point to the ground. This situation does not occur for the upper layers. In addition, it is not possible to make any assumption about the free space for these layers, since any object below the sensor height would not be detected. Therefore, the sensor model for the **two upper layers** is:

$$p(\rho|r) = \max\left\{p(\rho|r), \eta\mathcal{N}(\rho; r, \sigma_r)\right\} \tag{5.12}$$



(a) Polar grid

(b) Cartesian grid

Figure 5.6: Efficient, vectorized implementation of the lidar sensor model in polar coordinates, and transformation into Cartesian coordinates.

It is important to remark that the sensor model $p(\rho|r)$ is updated **sequentially**, starting with the lowest layer (Layer 1) and ending with the top layer (Layer 4). That is the reason why the term $p(\rho|r)$ is inside the max function in Equation 5.12: in order not to override the contributions of the previous layers.

5.2.3 Multi-Echo Selection

Common automotive lidars, such as the one used in this project, have multi-echo detection capabilities. This is especially useful under adverse weather conditions, such as rain, snow or dust. In these cases, the first laser reflection might be completely noisy due to light back-scattering, but further reflections may finally reach the real target.

For this reason, we make use of the **first available echo**. A few experiments under rainy conditions showed that the first echo was normally filtered out internally by the sensor, whereas the second and third echoes reached the target. We do not use all the three echoes since they are usually redundant and the computational time is greatly increased.

Implementation remarks

Given the large amount of data returned by the lidar and the fact that we create a sensor model for each and every laser beam, an efficient implementation is required in order to have a reasonable processing time. We compute the sensor models in parallel through **vectorization** in MATLAB. The result is a polar grid that contains the information from all the beams simultaneously, as shown in Figure 5.6a. This is later transformed into Cartesian coordinates, as described in Appendix C, Section C.2. An example of the result is presented in Figure 5.6b.

5.3 Radar

The radar sensor provides complementary measurements to the laser scanner. We use a FMCW millimeter-wave radar, which computes the range to the targets from the frequency shift between the emitted and received signals. It also estimates the radial velocity of the targets based on the Doppler effect. To cover a wide field of view, the radar performs an electronic sweep using a phased array of receptive antennas [94]. The interested reader is referred to [95] for a more detailed description of the radar working principle.

Input data

The radar sensor provides a set $\mathcal{Z}^R$ of measurements at every time step:

$$\mathcal{Z}^R = \{\mathbf{z}_1^R, \mathbf{z}_2^R, \dots, \mathbf{z}_{N_R}^R\} \quad ; \quad \mathbf{z}_i^R = \begin{pmatrix} r_i & \theta_i & v_i & \sigma_i \end{pmatrix}^\mathsf{T} \tag{5.13}$$

where N_R is the number of radar detections (at most 64, according to the manufacturer). Each measurement $\mathbf{z}_i^R$ contains the following information about a detected target:

- r: range to the detected target [m].
- θ: bearing to the detected target [rad].
- v: radial velocity of the target (in the direction of the beam) [m/s].
- σ: radar cross-section (RCS) of the target [dBsm].

In this project, only the values of range r and bearing θ will be used to define the probabilistic sensor model. The radial velocity, v, could be used to filter out measurements of dynamic targets, but we do not apply it in this work. The RCS is a measure of how detectable a target is, and it depends on a variety of factors: material, shape, etc. As will be discussed later, the RCS is be used to perform a weighted radar beam fusion, giving more confidence to electromagnetically large targets.

5.3.1 Inverse Sensor Model

Similar to the lidar model, it is more natural to develop the radar model in **polar coordinates**, in order to mimic as accurately as possible the physical process. However, a one-dimensional ray model of the radar is not accurate enough in this case, due to the large angular uncertainty, as can be seen in Table D.3. Therefore, every radar beam is modelled with two-dimensional Gaussian distributions in polar coordinates $\mathbf{x} = (\rho, \theta)$, as follows:

$$\left.\begin{aligned} f_e(\mathbf{x}) &= \eta_e \cdot \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_e, \Sigma_e) \\ f_o(\mathbf{x}) &= \eta_o \cdot \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_o, \Sigma_o) \end{aligned}\right\} \implies p(\mathbf{x}|\mathbf{z}) = 0.5 \cdot (1 + f_o - f_e) \tag{5.14}$$

where $\mathcal{N}(\mathbf{x}, \boldsymbol{\mu}, \Sigma)$ is the n-dimensional Gaussian distribution:

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{|2\pi\Sigma|^{n/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\mathsf{T}\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right) \tag{5.15}$$

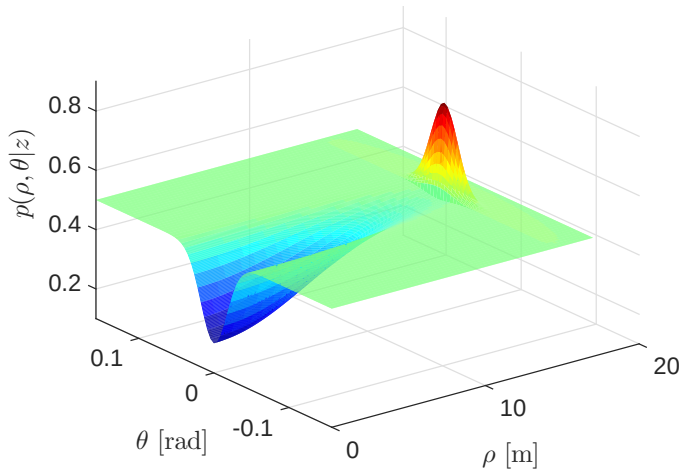


Figure 5.7: Inverse sensor model for a single radar beam, in polar coordinates. In this example, $r = 15$ m, $\theta = 0°$, Δr = 0.2 m, $\Delta\theta = 1°$.

The normalization factors η_e and η_o are required for the functions to be actual probability distributions. The function f_e and f_o represents the `empty` and `occupied` regions, respectively, with the following parameters:

$$\boldsymbol{\mu}_e = \begin{pmatrix} 0 \\ \theta_i \end{pmatrix} \quad ; \quad \Sigma_e = \begin{pmatrix} (r_i/2)^2 & 0 \\ 0 & \Delta_\theta^2 \end{pmatrix} \quad ; \quad \boldsymbol{\mu}_o = \begin{pmatrix} r_i \\ \theta_i \end{pmatrix} \quad ; \quad \Sigma_e = \begin{pmatrix} \Delta_r^2 & 0 \\ 0 & \Delta_\theta^2 \end{pmatrix} \tag{5.16}$$

In the angular dimension, the uncertainty is modelled by Δ_θ, equals to the angular accuracy specified by the supplier. The range uncertainty varies for f_e and f_o. To model empty space, an uncertainty $r/2$ is chosen so that the empty likelihood decreases with the distance as the beam approaches the obstacle. On the other hand, f_o models the range uncertainty directly using Δ_r, also known. Finally, the probability distribution for the Bayesian framework is computed from the combination of the previous two. The result is shown in Figure 5.7. Although more detailed models can be found in the literature [96], a more exhaustive characterization of the radar sensor is required. This information is not available or measurable in the context of this project, so a simpler model is developed.

Implementation remarks

The computation of the radar sensor model is performed **individually** for each of the 64 measurements (beams) returned by the radar, and then fused together using the classical Bayesian or Dempster-Shafer combination formulas. This is necessary since the Gaussian distributions might overlap in the angular dimension. In addition, the process is quite computationally expensive unless implemented efficiently. In particular, we only apply Equations 5.14 and 5.16 for the following values of ρ and θ:

$$0 \leq \rho \leq z_r + 3\sigma_r \quad ; \quad -(z_\theta + 3\sigma_\theta) \leq \theta \leq (z_\theta + 3\sigma_\theta) \tag{5.17}$$

These values are later transferred into a global polar grid in the fusion process. By considering the 3σ limit for the Gaussians we keep 99.7% of the information, which is a pretty close approximation and saves lots of computations.

5.3.2 Beam Weighting

In traditional sensor fusion approaches, the data from every sensor contributes equally. We propose a different approach, in which every radar beam has a *weight* associated to it. The main purpose is to mitigate the effect of **multi-echo** signals. Taking into account the radar equation [95], we analyze the received power for each target, P_r:

$$P_r = \frac{P_t G_t}{4\pi r^2} \cdot \sigma \cdot \frac{A_{eff}}{4\pi r^2} \propto \frac{\sigma}{r^4} \tag{5.18}$$

where P_t is the transmitted power, G_t is the gain of the emitting antenna, σ is the radar cross-section (RCS) of the target, A_{eff} is the effective area of the receiving antenna, and r is the range to the target. Since the transmitter sends a single waveform, P_t is constant for every measurement. The rest of the parameters are design constants. Therefore, we conclude that the received power is proportional to the RCS and the range.

When analyzing the received sensor data, we notice that multi-echo measurements have small RCS (σ) and large range. The relation σ/r^4 could therefore be considered as the weight of the radar beam. Nevertheless, this would also downweight measurements where there is actually an object behind another one. Therefore, we finally choose:

$$\tilde{w} = \sigma \propto P_r \cdot r^4 \tag{5.19}$$

In sum, the methodology to apply the beam weighting is described in Algorithm 2.

Algorithm 2 Radar beam weighting

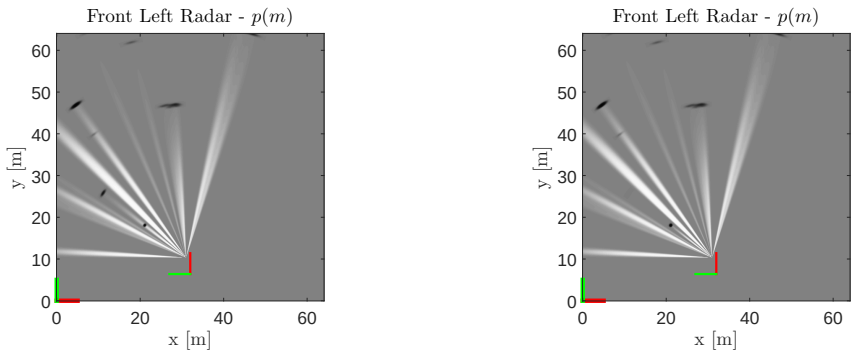
Input: $\mathtt{z}$ ▷ Set of radar measurements
Output: $\mathtt{w}$ ▷ Set of weights
$\mathtt{w} \leftarrow 1$ ▷ Weights initially set to a vector of 1
for $\left\{z^{(i)}, z^{(j)}\right\} \in \mathtt{z}, \quad j > i$ **do**
  if isMultiEcho$\left(z^{(i)}, z^{(j)}\right)$ **then**
    $p_i \leftarrow 10^{z_\sigma^{(i)}/10}$
    $p_j \leftarrow 10^{z_\sigma^{(j)}/10}$
    $\mathtt{w}(i) \leftarrow \mathtt{w}(i) \cdot p_i/(p_i + p_j)$ ▷ Weight according to z_σ (RCS)
    $\mathtt{w}(j) \leftarrow \mathtt{w}(j) \cdot p_j/(p_i + p_j)$
  end if
end for

As can be observed, we only apply beam weighting if two pairs of measurements are suspected to be multiple echoes. This is verified in the function `isMultiEcho`, which checks whether the azimuth angles are similar and the ranges are multiples of each other. The weights are initialized to one and updated when multiple reflections are found. The ones with larger power will end up with a larger weight. By using the RCS in the weight, we take into account the target's material. For instance, metallic objects, more dangerous and critical to be avoided, will have a larger weight, making their detection easier.

It is also important to perform this weighting only for multi-echo measurements. If this was performed among the whole set of measurements, only the measurements corresponding to the guardrails or vehicles, which are metallic and very close to the car, would obtain a relatively high weight and would be preserved.

Experiments

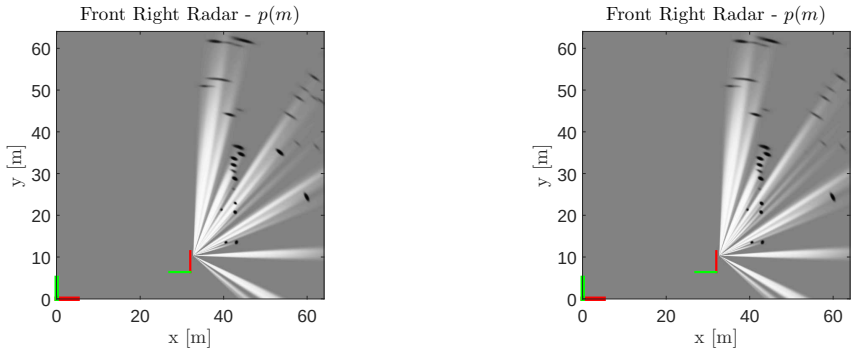
We have performed two experiments to validate this procedure. The first one contains just two corner reflectors, as shown in Figure E.1. The resulting occupancy grids for the left radar are presented in Figure 5.8. There is a corner reflector at approximately $(x, y) =$ 20 m, but a double reflection at a double radial distance from the sensor appears. After applying the weighting, this reflection is successfully removed.



(a) Before weighting (b) After Weighting

Figure 5.8: Radar weighting: experiment 1.

In the second experiment, we analyzed the influence of the weighting when multiple objects were placed at the same azimuth angle. The environment is shown in Figure E.5. We evaluate here the system's ability to detect the guardrail behind the objects while removing the double echoes. The results of running this experiment are shown in Figure 5.9.



(a) Before weighting (b) After weighting

Figure 5.9: Radar weighting: experiment 2.

The two large corner reflectors, with RCS = 20 dBsm, are clearly detected in both cases, at roughly $(x, y) = (40, 12)$ and $(40, 22)$ in the map. There are also detections for the guardrail behind them. In Figure 5.9a, we find a double reflection at $(x, y) = (50, 35)$ in the map, which overlaps with the detection of the guardrail at $(x, y) = (42, 23)$. If we perform weighting (see Figure 5.9b), the double reflection contribution is greatly reduced, so the detection of the guardrail, right behind the upper corner reflector, becomes clearer.

5.4 Probability Limitation

Similarly to what was done in Section 4.1.4, we establish a maximum and minimum level for the inverse sensor model, $p(\mathbf{m}|\mathbf{z}_t, \mathbf{x}_t)$, so that:

$$P^S_{\min} \leq p(\mathbf{m}|\mathbf{z}_t, \mathbf{x}_t) \leq P^S_{\max} \tag{5.20}$$

where we choose $P^S_{\min} = 0.2$, $P^S_{\max} = 1 - P^S_{\min}$. This can be implemented either by linearly rescaling the sensor models, or by replacing the Gaussian distributions with modified versions that enforce the minimum and maximum values.

The motivation for this is that sensors are not fully reliable, and therefore they should not be able to state that a cell is occupied with too high probability just after one iteration. Instead, by limiting the maximum and minimum probability on the sensor models we require an object to be detected along a number of frames to gain enough confidence about the state of occupancy of the corresponding cell in the grid.

5.5 Dempster-Shafer Sensor Models

All the sensor models that we have described so far provide a probabilistic inverse sensor model $p(\mathbf{m}|\mathbf{z}_t, \mathbf{x}_t)$, i.e. expressed in the classical Bayesian probability theory. The process must also be done for the Dempster-Shafer framework, for which we have to provide the belief masses for both the `empty` and `occupied` states.

Since we want to compare the Bayesian and Dempster-Shafer approaches, their sensor models should be as equivalent as possible. To do that, we build the Dempster-Shafer models upon the Bayesian models, as follows:

$$\begin{cases} \mathbf{m}(O) = p(\mathbf{m}|\mathbf{z}_t, \mathbf{x}_t) \\ \mathbf{m}(E) = 0 \end{cases} \quad ; \quad p(\mathbf{m}|\mathbf{z}_t, \mathbf{x}_t) > 0.5$$

$$\begin{cases} \mathbf{m}(O) = 0 \\ \mathbf{m}(E) = 1 - p(\mathbf{m}|\mathbf{z}_t, \mathbf{x}_t) \end{cases} \quad ; \quad p(\mathbf{m}|\mathbf{z}_t, \mathbf{x}_t) < 0.5 \tag{5.21}$$

$$\begin{cases} \mathbf{m}(O) = 0 \\ \mathbf{m}(E) = 0 \end{cases} \quad ; \quad p(\mathbf{m}|\mathbf{z}_t, \mathbf{x}_t) = 0.5$$

Here we can observe one of the benefits of the Dempster-Shafer theory: we can assign support to the `occupied` and `empty` events separately. In the first two cases, one of those states is assigned a belief mass of 0; otherwise we would have a sensor model with contradicting information. In addition, it can be noted that the value $p(\mathbf{m}|\mathbf{z}) = 0.5$ is not assigned to either $\mathbf{m}(O)$ or $\mathbf{m}(E)$. It is therefore implicitly assigned to the frame of discernment $\Theta = \{O, E\}$ so that $\mathbf{m}(\Theta) = 1$, which represents the `unknown` state.

Another thing to note is that these models are not *exactly* the same as the Bayesian ones, since the probability and belief mass definitions are not the same. Later, we will evaluate how this affects the final results.

Chapter 6

Grid Fusion

After updating the individual occupancy grids from every sensor data, grid fusion is performed. The result is a single occupancy grid combining the information gathered from all the sensors. We analyze the most common approaches within both the Bayesian and Dempster-Shafer domains.

6.1 Bayesian Fusion

There exist multiple approaches in the literature to perform sensor fusion using the Bayesian probability theory [97]. The most popular ones have been implemented and analyzed for this project. After a brief discussion, the most suitable is selected in the rest of the project. Let K be the number of sensors used for the fusion. For simplicity in the notation, let us define:

$$p(\mathbf{m}) = p(\mathbf{m}|\mathbf{z}^1_{1:t}, \dots, \mathbf{z}^K_{1:t}) \tag{6.1}$$

$$p_k(\mathbf{m}) = p(\mathbf{m}|\mathbf{z}^k_{1:t}) \quad ; \quad 1 \leq k \leq K \tag{6.2}$$

where $p(\mathbf{m})$ is the posterior of the map considering (fusing) all the sensors, and $p_k(\mathbf{m})$ is the posterior considering all the measurements $\mathbf{z}^k_{1:t}$ from only the kth sensor. The latter is the result of the occupancy grid mapping module implemented so far, for each sensor.

- Thrun *et al.* [70] suggest several approaches. The simplest and most conservative one is to just take the **maximum probability of occupancy**, for every cell:

  $$p(\mathbf{m}) = \max_k p_k(\mathbf{m}) \tag{6.3}$$

  This approach will generate maps biased towards a high number of occupied cells, so it is likely to increase the false positive rate. In addition, it will not take advantage of the complementarity of the sensors: `unknown` areas will be prioritized over `empty` areas, with lower probability.

  They also mention the possibility to apply the **De Morgan Law** to this problem:

  $$p(\mathbf{m}) = 1 - \prod_k (1 - p_k(\mathbf{m})) \quad ; \quad 1 \leq k \leq K \tag{6.4}$$

However, a quick analysis shows that it is also highly biased towards high probabilities. For example, if 10 sensors report $p_k(\mathbf{m}) = 0.5$ (`unknown`), the result is $p(\mathbf{m}) = 0.999$ (`occupied`), without any actual evidential support.

- A popular combination rule is the **Linear Opinion Pool** (LOP), used for instance by Adarve *et al.* [40] to fuse a multi-layer lidar and a stereo-vision system.

$$p(\mathbf{m}) = \alpha \sum_k w_k p_k(\mathbf{m}) \quad ; \quad \alpha = \left[\sum_k w_k \right]^{-1} \tag{6.5}$$

A set of weights $0 \leq w_k \leq 1$ allow to establish how much each sensor contributes to the fusion.

- A similar approach is the **Independent Opinion Pool** (IOP), applied to sources considered to be independent. It represents the geometric mean [98]:

$$p(\mathbf{m}) = \alpha \prod_k p_k(\mathbf{m}) \quad ; \quad \alpha = \left[\prod_k p_k(\mathbf{m}) + \prod_k (1 - p_k(\mathbf{m})) \right]^{-1} \tag{6.6}$$

A generalization of this method is the **Logarithmic Independent Opinion Pool** (LIOP), which introduces weights for each probability distribution:

$$p(\mathbf{m}) = \alpha \prod_k p_k(\mathbf{m})^{w_k} \quad ; \quad \alpha = \left[\prod_k p_k(\mathbf{m})^{w_k} + \prod_k (1 - p_k(\mathbf{m}))^{w_k} \right]^{-1} \tag{6.7}$$

with $0 \leq w_k \leq 1, \sum_k w_k = 1$. It is therefore the dual of the linear opinion pools technique: the first performs a weighted average and the second a geometric weighted average.

- Another method that is referred to in the literature is the **Superbayesian Independent Opinion Pool**, which is a direct application of Bayes' Rule, but making a more careful probabilistic formulation. However, it is shown in the literature [99][42] that it reduces to the Independent Opinion Pool in the particular case of the occupancy grid problem, where cells have a binary state (`empty`, `occupied`).

- Finally, it is also possible to apply the **Bayes Filter** formulation (see Appendix B) in a similar fashion as it was done previously, where *temporal* fusion was performed considering that all the measurements were independent. The only caveat is to assume **maximum entropy priors** for all the grids, so that the term $\log \frac{p(m_{ij})}{1-p(m_{ij})}$ in Equation B.7 becomes zero and therefore is not considered K times in the fusion process. Here we work with the *log-odds* instead of probabilities:

$$l(\mathbf{m}) = \log \frac{p(\mathbf{m})}{1 - p(\mathbf{m})} \quad ; \quad l_k(\mathbf{m}) = \log \frac{p_k(\mathbf{m})}{1 - p_k(\mathbf{m})} \tag{6.8}$$

The fused log-likelihood then simply becomes:

$$l(\mathbf{m}) = \sum_k l_k(\mathbf{m}) \tag{6.9}$$

6.1.1 Analytical Evaluation

We first evaluate the performance of these methods with respect to some representative cases. We simulate some scenarios over a single cell.

Single-sensor contribution

We first analyze the result when only one sensor provides information about the state of occupancy of a cell, i.e. $p_k(m_{ij}) \neq 0.5$ only for the k^*th sensor.

- **Bayes' rule**. The log-likelihood for all the sensors is zero except for the one that provides information, so $p(\mathbf{m}) = p_{k^*}(\mathbf{m})$.
- **Independent Opinion Pool**. As before, the result ends up depending only on the map generated by one sensor:

$$p(\mathbf{m}) = \frac{p_{k^*}(\mathbf{m}) \cdot 0.5^{K-1}}{p_{k^*}(\mathbf{m}) \cdot 0.5^{K-1} + (1 - p_{k^*}(\mathbf{m})) \cdot (1 - 0.5)^{K-1}} = p_{k^*}(\mathbf{m}) \tag{6.10}$$

- **Linear Opinion Pool**. Assuming $w_k = 1/K, \forall k$ (same sensor contribution):

$$p(\mathbf{m}) = \frac{p_{k^*}(\mathbf{m})}{K} + \frac{(K-1) \cdot 0.5}{K} \tag{6.11}$$

- **Logarithmic Independent Opinion Pool**.

$$p(\mathbf{m}) = \frac{p_{k^*}(\mathbf{m})^{1/K} \cdot 0.5^{(K-1)/K}}{p_{k^*}(\mathbf{m})^{1/K} \cdot 0.5^{(K-1)/K} + (1 - p_{k^*}(\mathbf{m}))^{1/K} \cdot (1 - 0.5)^{(K-1)/K}} = \tag{6.12}$$

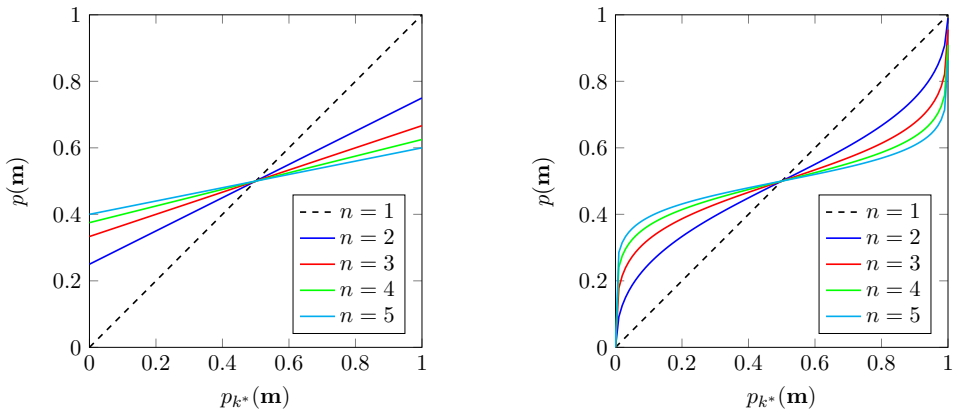
$$= \frac{p_{k^*}(\mathbf{m})^{1/K}}{p_{k^*}(\mathbf{m})^{1/K} + (1 - p_{k^*}(\mathbf{m}))^{1/K}} \tag{6.13}$$

The last two techniques are plotted in Figure 6.1. We observe that for these two approaches the final probability $p(\mathbf{m})$ **decreases** whenever other sensors provide no new information about the state of a cell. The LIOP method does not suffer that much from this issue, especially if the sensor is very confident ($p_{k^*}(\mathbf{m}) > 0.9$). In any case, both methods approach the unknown state when the number of non-informative sensors increases. This is not useful in cases when sensors are **complementary**, both in terms of the area they cover and also their physical working principle. For this project, we work with sensors based on different physical principles and with complementary fields of view, so these fusion techniques are not adequate.

General fusion

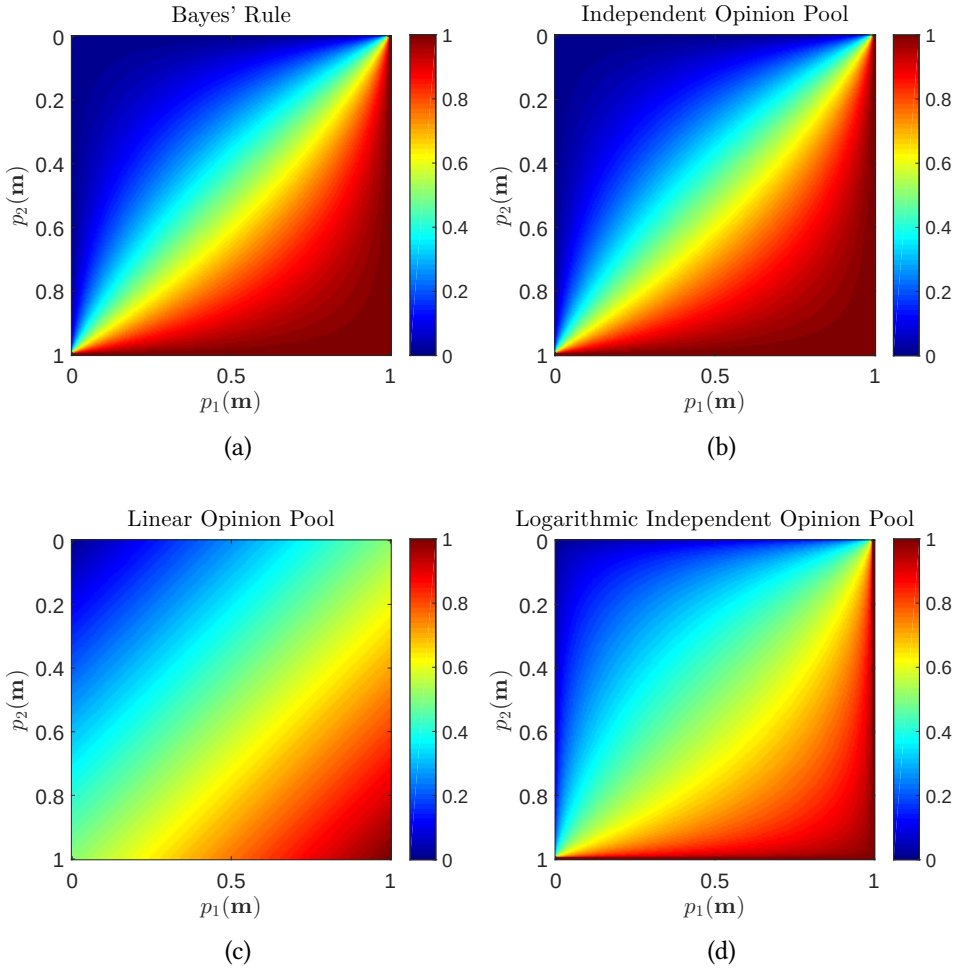
We analyze now how the different approaches work when two sensors report a different occupancy probability. For simplicity in the analysis and reasoning, only two sensors are considered. We apply the previous formulas for every combination of p_1 and p_2, ranging from 0 to 1 at fixed steps of 0.01. The result is a matrix where the cell ij corresponds to the fusion of $p_1(i)$ and $p_2(j)$, as can be seen in Figure 6.2.

We analyze these figures according to two scenarios:



(a) Linear Opinion Pool (b) Logarithmic Independent Opinion Pool

Figure 6.1: Bayesian fusion of n sensors, of which only one provides information.



(a) (b) (c) (d)

Figure 6.2: Bayesian fusion of $p_1(\mathbf{m})$ and $p_2(\mathbf{m})$, for every possible combination ($w_1 = w_2 = 0.5$).

- Sensors with **supporting information**. This corresponds to the top-left and bottom-right regions, for every image. Again, we observe the exact same result for the Bayes' Rule and the IOP. As expected, the status of a cell is reinforced when two sensors agree on it. This is not the case for LOP and LIOP, and reason is the same as in the previous case: an average is performed and therefore the output probability cannot be greater than any of the individual probabilities.
- Sensors with **conflicting information**. This corresponds to the region around the diagonal that goes from bottom-left to top-right. When there is total conflict the output probability is 0.5 (`unknown`). Then, for every technique, what varies is the "width" of this area. It is especially interesting to compare Bayes/IOP to LIOP. The latter has a wider `unknown` area, which means it will be a bit more conservative: only when the conflict is small or one of the sensors is confident enough, the probability will be high. The LOP method is the most conservative in the high-conflict regions, since the output probability is confined in the `unknown` region. The drawback is that it might increase the false negative rate.

6.1.2 Conclusion

In order to select the most suitable method, we must first define the desirable features the fusion module must have:

- Computationally cheap.
- A sensor reporting `unknown` should not affect the final state of the map.
- In case of supporting information, the belief about the state (either `occupied` or `empty`) should increase.
- In case of conflicting information, the probability should tend towards the `unknown` state if two sensors are similarly confident.

First, we notice that Bayes' Rule and the Independent Opinion Pool produce exactly the same result. This makes sense since they come from the same formula —Bayes' rule. We would then prefer the Bayes' Rule in log-odds form over the IOP due to a much lower computational complexity. Second, we completely discard the Linear Opinion Pool given that `unknown` cells do contribute to the final output. The Logarithmic Independent Opinion Pool also suffers from this problem, but to a lesser extent, especially if the other sensors are very confident about the status of the cell. As can be observed in Figure 6.2, it provides a more conservative estimate when having conflicting information (the close-to-`unknown` region, in green, is wider). This in turn will result in a less informative grid, which is not interesting for our application. In an experimental evaluation, no significant qualitative differences could be appreciated. The computation time is however much larger in the LIOP than in the Bayes' Rule. Nevertheless, it is interesting in the sense that some weights can be employed to establish a different contribution for each sensor.

In conclusion, the **Bayes' Rule** in log-odds form is the most suitable fusion technique for this application. It is very cheap to compute since every individual grid is already computed in log-odds form and it tends to provide more information than the other grids (it is slightly less conservative).

6.2 Dempster-Shafer Fusion

The second framework studied in this project is the Dempster-Shafer theory of evidence, as described in Chapter 4. The core of the fusion process is the Dempster-Shafer combination formula (see Equation 4.21).

Following the same reasoning as in the Bayesian case, we can use the combination formula to fuse multiple sensors apart from performing temporal fusion. Furthermore, it can be shown that the Dempster-Shafer combination rule is *commutative* and *associative* [100]. Therefore, the fusion of K sensors can be performed **sequentially**, as follows:

$$\oplus_{i=1}^{K} m_i = (\dots((m_1 \oplus m_2) \oplus m_3) \oplus \dots \oplus m_K) \tag{6.14}$$

The order in which the sensors are fused is irrelevant given the commutativity property.

6.2.1 Modifications to the Classical Combination Rule

The fusion of an heterogeneous set of sensors must be done a bit more carefully than when performing temporal fusion using a single sensor, since situations of **conflicting information** are much more likely. The problem of conflicting information has been subject to criticism, as discussed in Section 4.2.5. Many authors in the literature have come up with their own combination formulas to solve the specific problems at hand, being quite *ad-hoc*, as shown by Bicheng *et al.* [79]. There are even more advanced solutions that involve a complex conflict management [101]. We investigate two modifications of the classical Dempster-Shafer combination formula:

- Introduction of **weights** to model sensor reliability, as proposed by Wu *et al.* [46]:

$$(m_1 \textcircled{\cap} m_2)(X) = \sum_{A,B \in 2^\Theta | A \cap B = X} w_1 m_1(A) \cdot w_2 m_2(B) \tag{6.15}$$

  where $0 \leq w_i \leq 1$. This operation is also known as *discounting factor*[102], similarly to the forgetting factor when performing temporal fusion. The weights w_i can be assigned manually to give more importance to some sensors than others. However, as we will see in Chapter A, these weights can be estimated automatically by a **sensor performance estimator** module.

- **High-conflict explicit management**. When fusing heterogeneous sensors it is possible that a large conflict $K \approx 1$ appears. In the extreme case in which $K = 1$, the combination formula yields a result of $0/0$, which is undetermined. Therefore, it is necessary to handle these cases manually. The key issue is the normalization factor in Equation 4.21. Since the null set must always have a zero mass, by definition, it must be redistributed over the rest of the power set of the frame of discernment. This is correct for small conflict K, but when it is large it will give counterintuitive results for the `empty` and `occupied` states.

  To solve this, we decide to analyze a popular conservative approach, according to the following reasoning: if two sensors are in high conflict about the state of occupancy of a grid, the true state is not really known. Thus, all the conflict mass should be distributed only to the `unknown` state, Θ.

Taking this into account, the Dempster-Shafer formula is modified as follows:

1. First, compute the conjunctive rule:

$$m_{12}(X) = \begin{cases} (m_1 \textcircled{\cap} m_2)(X) & X \neq \emptyset \\ 0 & X = \emptyset \end{cases} \tag{6.16}$$

$$K = (m_1 \textcircled{\cap} m_2)(\emptyset) \tag{6.17}$$

2. Then, the conflict is analyzed. If $(1 - K) > \epsilon_K$:

$$m_{12}(X) \leftarrow \frac{m_{12}(X)}{1 - K} \tag{6.18}$$

   otherwise:

$$m_{12}(\Theta) \leftarrow K \tag{6.19}$$

When $\epsilon_K = 0$, the previous condition is always fulfilled and the classical Dempster-Shafer rule is applied (always normalize). On the other hand, if $\epsilon_K = 1$, the normalization is never performed, which produces as a result **Yager's rule** [78], also quite studied in the Dempster-Shafer literature.

6.2.2 Analytical Evaluation

We perform a similar analysis as we did in the Bayesian case.

Single-sensor contribution

We first consider the fusion of two sensors, one of which provides absolutely no new information. Consider the following example, for $0 \leq a \leq 1$:

$$m_1 \begin{cases} m_1(O) = a \\ m_1(E) = 0 \\ m_1(\Theta) = 1 - a \end{cases} \quad ; \quad m_2 \begin{cases} m_2(O) = 0 \\ m_2(E) = 0 \\ m_2(\Theta) = 1 \end{cases} \tag{6.20}$$

If we apply either of the Dempster-Shafer rules discussed before, we obtain:

$$K = 0 \implies \begin{cases} m_{12}(O) = a = m_1(O) \\ m_{12}(E) = 0 = m_1(E) \\ m_{12}(\Theta) = 1 - a = m_1(\Theta) \end{cases} \tag{6.21}$$

We conclude that, with the Dempster-Shafer rule of combination, sensors that do not provide new information **do not alter** the mass distribution of the additional sensors they are fused with. This is one of the main properties we are looking for in a fusion algorithm, as discussed before.

Conflicting information

Similarly to what was done in the Bayesian case, we analyze the output for a single cell when fusing two sources m_1 and m_2, but paying attention to the case of conflicting information only. The following values are considered:

$$m_1 \begin{cases} m_1(O) = 0 : 0.01 : 1 \\ m_1(E) = 0 \\ m_1(\Theta) = 1 - m_1(O) \end{cases} \quad ; \quad m_2 \begin{cases} m_2(O) = 0 \\ m_2(E) = 0 : 0.01 : 1 \\ m_2(\Theta) = 1 - m_2(E) \end{cases} \tag{6.22}$$

where the MATLAB notation $0 : 0.01 : 1$ represents a vector of values ranging from 0 to 1 at fixed intervals of 0.01. Note that we are fusing values of m_1 and m_2 that are always in conflict, since one of them has mass function in the `occupied`(O) event and the other one in the `empty`(E) event.

First, we compute the conflict $K = (m_1 \textcircled{\cap} m_2)(\emptyset)$ for all combinations between m_1 and m_2, giving as a result the conflict matrix shown in Figure 6.3:

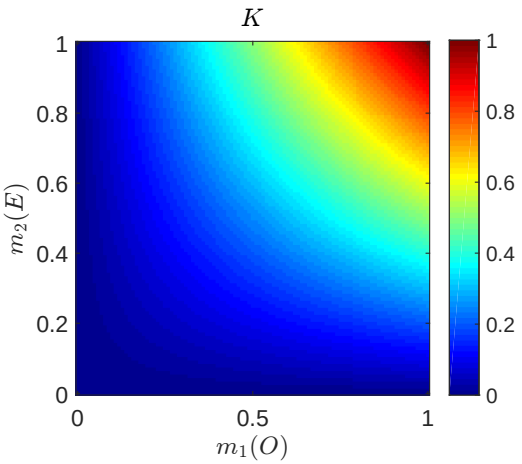


Figure 6.3: Conflict K as a result of fusing every combination of $m_1(O)$ and $m_2(E)$ in the Dempster-Shafer domain.

As expected, the conflict increases when both mass functions approach to one. This means that they become more and more confident about the probability of the event, and since they represent **opposite events** a high conflict appears. Now we evaluate the output of the fusion for three cases: $\epsilon_K = 0, 0.5$ and 1. Figures 6.4, 6.5 and 6.6 present the outputs $m_{12}(E)$, $m_{12}(O)$, $m_{12}(\Theta)$ and the probability $p_{12}(\mathbf{m})$, obtained through the pignistic transformation as described in Section 4.2.6.

First, let us discuss Figure 6.4, corresponding to the fusion using the classical Dempster-Shafer rule. We first notice some counter-intuitive result: a high conflict (K) does **not** produce an increase of uncertainty in the top-right corner of $m_{12}(\Theta)$, as it would be expected. If we analyze m_Θ, we observe that the uncertainty is only high when both sensors are not so confident. Finally, taking a look at the output grids $m_{12}(O)$ and $m_{12}(E)$ in the region where there conflict is largest (top-right corner), we observe that it is a very

sensitive area. In other words, $m_1(O)$ can dominate over $m_2(E)$ quite easily with a small increase of $m_1(O)$, and vice versa.

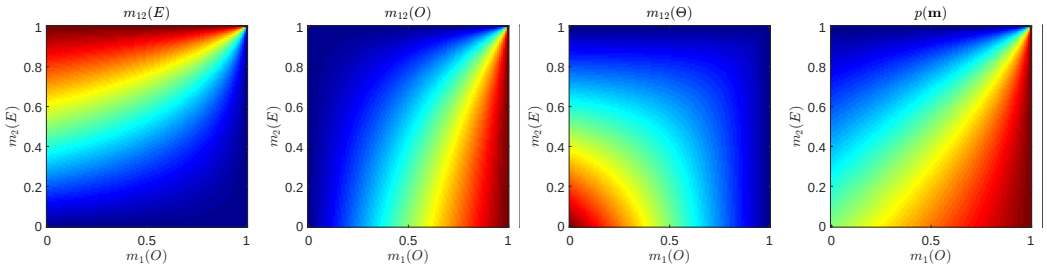


Figure 6.4: Fusion of conflicting information. $\epsilon_K = 0$ (Dempster-Shafer formula).

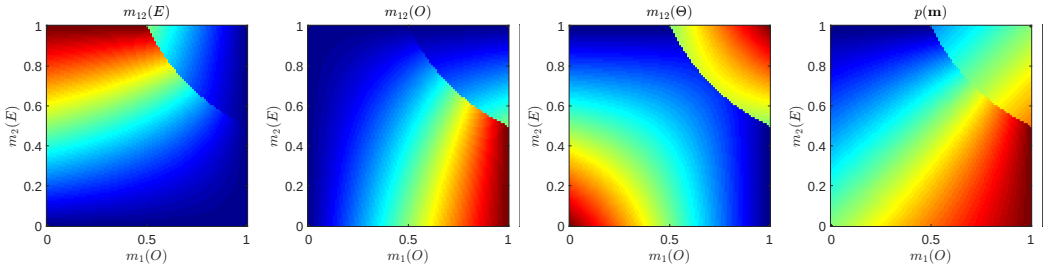


Figure 6.5: Fusion of conflicting information. $\epsilon_K = 0.5$.

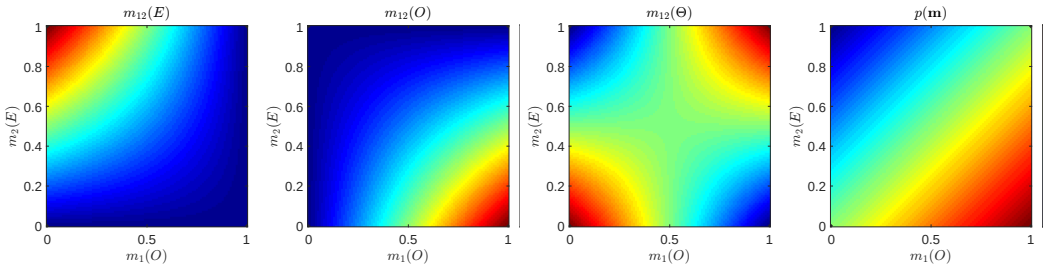


Figure 6.6: Fusion of conflicting information. $\epsilon_K = 1$ (Yager's rule).

The result is that cells with high conflict will most likely alternate between `occupied` and `empty` state after the decision rule. The reason for this situation is the normalization factor, as some authors pointed out. It is basically redistributing the conflict mass belonging to the null set into the other events, but a very small fraction is actually assigned to the **unknown** state since the sensors are very confident.

A solution to this is to apply the conflict redistribution only to the `unknown` state, as can be shown in Figure 6.6. The first noticeable difference with respect to the classical Dempster-Shafer rule is that there is a high value for m_Θ in the top-right corner, i.e. when the two sensors are in high conflict. Furthermore, the output values of $m_{12}(E)$ and $m_{12}(O)$

are only high when the conflict is low. This is a more conservative result: if two sensors contradict each other, and we rely equally on both, then we cannot rule out the true state of the cell and it should be set to unknown. In addition, the probability $p(\mathbf{m})$ is exactly the same as for the Linear Opinion Pool in the Bayesian case.

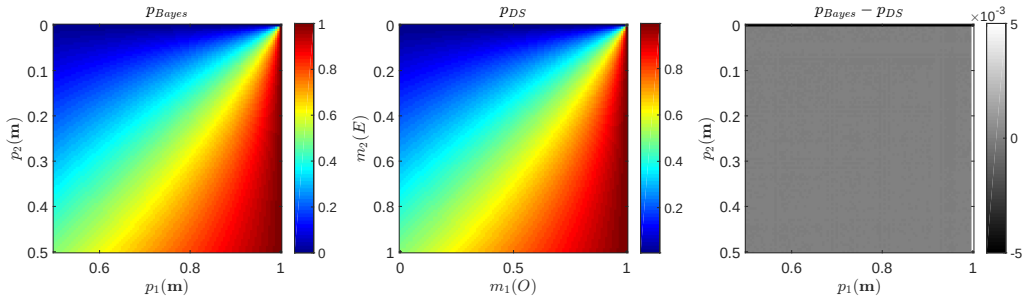
Finally, an intermediate solution, with $\epsilon_K = 0.5$, is presented in Figure 6.5. It only modifies the top-right corner of the matrix, which means it only affects the case of conflicting information. There is however a very sharp transition in the $m_{12}(E)$, $m_{12}(O)$ and $m_{12}(\Theta)$ matrices, which is not so desirable.

In experimental tests, we observed that in general both the classical Dempster-Shafer formula and Yager's rule give pretty similar results and the differences are negligible, so either of those would work just fine. We however choose the **classical Dempster-Shafer rule** since it is a bit less conservative and is able to return a higher probability value than Yager's rule when there is conflict. This often helps **reducing the false negative rate**, which tends to be large when there is conflicting information situations.

6.3 Bayesian and Dempster-Shafer Fusion: Comparison

The previous theoretical analysis fusing two data sources is very useful to graphically compare the Bayesian and Dempster-Shafer behaviour. This will be useful later on to be able to explain the experimental results.

In Figure 6.7 we put together the results of the previous analysis for both the Bayesian and Dempster-Shafer theories, for a clearer view. To simplify the comparison, Figure 6.7a corresponds to the top-right corner of Figure 6.2a, where there is **conflicting information** between the sensors, just the same as the Dempster-Shafer figure.



(a) Bayes's Rule (b) Dempster-Shafer (c) Difference

Figure 6.7: Comparison between Bayes and Dempster-Shafer fusion in a conflicting information scenario.

These figures prove analytically that the Bayesian and Dempster-Shafer approaches, applied to this particular problem of occupancy grid mapping, and using the proposed sensor models, will yield **pretty similar results**. Figure 6.7c shows the difference when subtracting the previous two images. We notice that they are not exactly the same, but the maximum absolute error per cell is 10^{-3}, which is negligible. This is an unexpected result, considering that the sensor models that we proposed for the Dempster-Shafer theory

are conceptually different, assigning the remaining mass to the unknown state. Therefore, we should not expect big differences when performing multi-sensor fusion using the vanilla Dempster-Shafer formula. Differences might come from the *temporal fusion*, where these little difference may accumulate over time and not be negligible anymore.

Nevertheless, in Chapter A we will propose new ideas about how to leverage the additional features of the Dempster-Shafer theory to improve the results. In particular, we will pay attention to a further analysis of conflict, which is obtained directly after applying the Dempster-Shafer combination formula.

6.4 Centralized and Decentralized Sensor Fusion

In the literature, the concept of sensor fusion is typically used in two different contexts. It is worth distinguishing those:

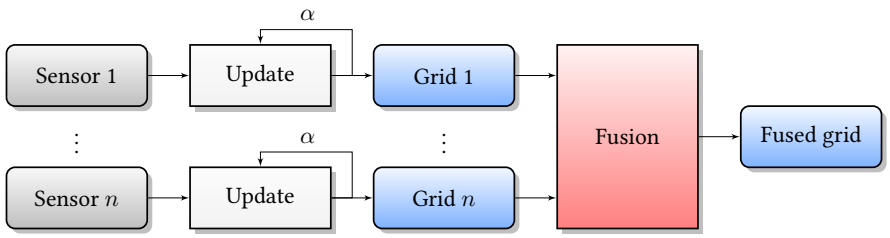
- **Temporal fusion**. This refers to the case in which only **one** sensor is used, and sensor fusion is used to integrate the information over time.
- **Multi-sensor fusion**. In this case, the information from a number of sensors is combined to form a new piece of information, at a **single time instant**.

In this project, the two concepts are used. First, individual occupancy grids are created to integrate the information from each sensor over time, independently (temporal fusion). Then, at every time step, all the sensor grids are fused together to create a combined grid (multi-sensor fusion). At this point, several questions could be asked. First, why separate the fusion process into two stages (temporal and multi-sensor fusion)? Second, in which order should those stages be applied? This gives two possible ways of implementing sensor fusion: **centralized** and **decentralized**, as illustrated in Figure 6.8. Even though the Bayesian and Dempster-Shafer fusion formulas are associative and commutative, the order does matter due to the **predict-update** loop in temporal fusion procedure.

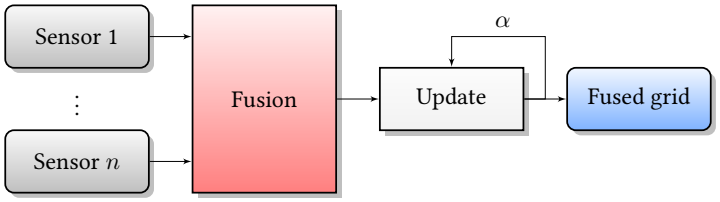
In this project, we chose the *decentralized* approach for several reasons. First, it allows us to filter each sensor individually through its own occupancy grid, therefore removing outliers more easily. In addition, it is easier to obtain a clean, readable and scalable code. Finally, it can be easily extended to, for instance, a **hybrid fusion** approach, where temporal fusion is performed in the Bayesian domain and multi-sensor fusion uses the Dempster-Shafer theory. The only drawback is, of course, the memory consumption, since multiple grids must be maintained. This is however not an issue in modern computers. This is the approach originally followed by Elfes [31] when he introduced the occupancy grid mapping paradigm. Nevertheless, centralized approaches can also be found in the literature, such as in Kurdej *et al.* [103], which use the Dempster-Shafer framework to fuse sensor and map information.

6.5 Time Synchronization

Another important difficulty within sensor fusion is **time synchronization**. Every sensor outputs information at a specific rate and time stamp, which might be different from the rest of the sensors'. In this project, the sensor data is registered by a central system



(a) Decentralized sensor fusion



(b) Centralized sensor fusion

Figure 6.8: Different grid-based sensor fusion schemes: centralized and decentralized.

which timestamps the data using GPS time. However, the sensors are *not* synchronized with each other. In addition, the data does not come from the sensors at a perfectly fixed rate; instead, it has some fluctuations due to the sensor itself or the CAN/Ethernet networks that connect the sensors with the on-board computers. Furthermore, the sensor fusion module outputs information at a fixed rate, usually higher than the sensors' rate. This is required for the following modules (planning and control) to be able to react quickly. This situation is illustrated in Figure 6.9.

The solution we propose for this particular project is based on a **predictive scheme**. The data that is used to compute the output of the sensor fusion module is a *prediction* based on the latest sensor data available. This prediction is based on two assumptions:

- The world is static.
- The vehicle moves following a constant velocity motion model.

The first assumption can be justified considering that we are only interested on detecting static objects on the road; dynamic objects are taken care of by other modules. The second assumption is reasonable considering the high output rate in comparison with normal accelerations in vehicles. Even if the velocity is not constant, it will be fairly well approximated by a piece-wise linear function over time. After these considerations, we present the prediction for each sensor. At the measurement time t_s, the measurement $\mathbf{z}_{t_s}$ is taken. The goal is to predict the measurement at output time $\mathbf{z}_{t_{\text{out}}}$, with $t_{\text{out}} \geq t_s$. This situation is depicted in Figure 6.10. It is easier to handle geometric transformations in Cartesian coordinates. Therefore, we first transform our polar measurements from radar and lidar to Cartesian coordinates. Next, the following steps are performed:

1. The points at sensor measurement time t_s, $\mathbf{z}_{t_s}^S = \begin{pmatrix} \mathbf{x}_{t_s}^S & \mathbf{y}_{t_s}^S \end{pmatrix}$, are transformed into the `vehicle` coordinate frame, knowing the corresponding transformation from

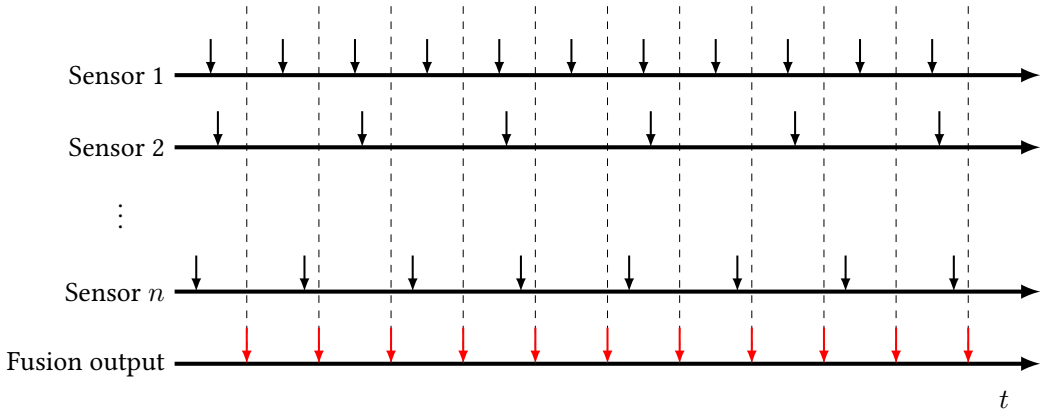


Figure 6.9: Time synchronization problem. The fusion module outputs information at a fixed rate (red). However, every sensor data is received at a different rate, which might also vary over time.

the `sensor` frame, T_v^s:

$$\mathbf{z}_{t_s}^V = T_v^s \cdot \mathbf{z}_{t_s}^S \tag{6.23}$$

2. Next, we compute the measurements $\mathbf{z}_{t_{\text{out}}}^V$ with respect to the new `vehicle` pose $T_w^{v'}$ at output time, t_{out}, assuming that the points of the object in `world` coordinates, $\mathbf{z}_{t_s}^W = \mathbf{z}_{t_{\text{out}}}^W$ have not changed, since they are static:

$$\begin{cases} \mathbf{z}_{t_s}^W = \mathbf{z}_{t_{\text{out}}}^W \\ \mathbf{z}_{t_s}^W = T_w^v \mathbf{z}_{t_s}^V \\ \mathbf{z}_{t_{\text{out}}}^W = T_w^{v'} \mathbf{z}_{t_{\text{out}}}^V \end{cases} \implies \mathbf{z}_{t_{\text{out}}}^V = \left(T_w^{v'}\right)^{-1} \cdot T_w^v \mathbf{z}_{t_s}^V \tag{6.24}$$

The ego-vehicle motion $\begin{pmatrix} \delta x & \delta y & \delta\theta \end{pmatrix}^\mathsf{T}$ between the times t_s and t_{out} is estimated as described in Section 4.6. The new pose of the vehicle is computed concatenating the transformation matrices. Here we use *post*-multiplication, since the motion is given in local `vehicle` coordinates:

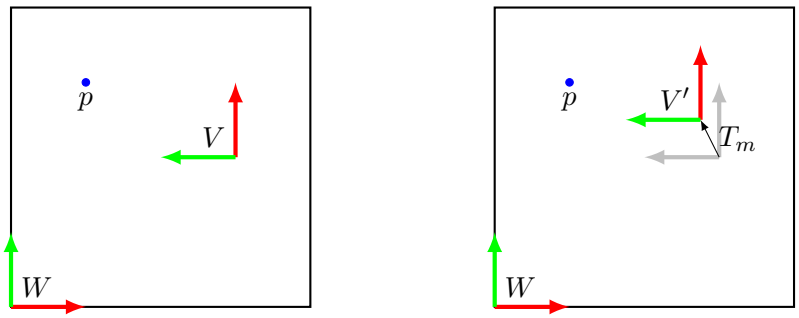
$$T_w^{v'} = T_w^v \cdot T_m \quad ; \quad T_m = \begin{pmatrix} \cos(\delta\theta) & \sin(-\delta\theta) & \delta x \\ \sin(\delta\theta) & \cos(\delta\theta) & \delta y \\ 0 & 0 & 1 \end{pmatrix} \tag{6.25}$$

where T_m is the motion transformation matrix, in `vehicle` coordinates. Finally:

$$\mathbf{z}_{t_{\text{out}}}^V = (T_w^v \cdot T_m)^{-1} \cdot T_w^v \cdot \mathbf{z}_{t_s}^V = T_m^{-1} \cdot \mathbf{z}_{t_s}^V \tag{6.26}$$

3. Finally, the points are transformed back to the local `sensor` coordinate frame, using $T_s^v = (T_v^s)^{-1}$.

$$\mathbf{z}_{t_{\text{out}}}^S = T_s^v \mathbf{z}_{t_{\text{out}}}^V \tag{6.27}$$



(a) $t = t_s$ (b) $t = t_{\text{out}}$

Figure 6.10: Sensor data prediction to compensate for imperfect time synchronization. The sensor gives a measurement at time $t = t_s$ (6.10a), but the system outputs the result at $t = t_{\text{out}}$ (6.10b). Meanwhile, the vehicle pose has been transformed by T_m. Therefore, sensor measurements need to be transformed to the new vehicle pose.

After converting back to polar coordinates, the predicted measurements are ready to be used to build the inverse sensor models and perform grid update, as described in previous chapters.

6.6 Spatial Calibration

Another important issue when working with multiple sensors is always their **extrinsic calibration**: to find the 6 degrees-of-freedom (DoF) transformation from every `sensor` to a common reference frame, which is normally the `vehicle` coordinate frame. If the calibration is not correct, the measurements will not fill the correct cells in the occupancy grid and the fusion will not be successful.

Fortunately, for this project the geometric sensor calibration is already available from previous work, so we limit ourselves to just use the transformation matrices that we are provided with. Otherwise, the calibration would normally involve setting up an environment with different objects, whose position is known accurately with respect to the vehicle. Then, measurements would be taken to try to match the sensor measurements to the actual object position. It might sometimes be necessary to move the vehicle to extract the complete 6 DoF pose, for which the odometry information must be included in the derivations.

Chapter 7

Obstacle Detection

After performing grid fusion, the information from the different sensors has been integrated into a single occupancy grid. The final step is to process this fused grid in order to detect potential objects and extract a high-level representation of them, which will be the final output of the developed system.

7.1 Overview

The pipeline for the obstacle detection module is represented in Figure 7.1.

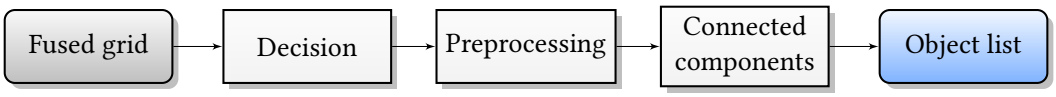


Figure 7.1: Obstacle detection pipeline.

The input is the fused grid resulting from the fusion of the individual occupancy grids. The first step in the process is to apply the **decision rule**, as described in Sections 4.1.5 and 4.2.6, to classify cells as `occupied` or `empty`. The result is therefore a binary image, suitable for image processing techniques. Prior to performing object detection, the resulting image is preprocessed to improve the performance of the subsequent stages. Finally, obstacles are extracted through the **connected components** technique. The output is an object list containing relevant information for the control logic of the vehicle to be able to correctly avoid the obstacles. All these steps are explained in more detail in the following sections.

7.2 Preprocessing

When analyzing the resulting decision grid, we observe that regions corresponding to objects are sometimes noisy and discontinuous. The first issue is caused by the intrinsic sensor noise; however, the discontinuities in the objects are due to the finite resolution of the grid and discretization errors when computing the proposed sensor models. An example illustrating this situation is presented in Figure 7.2.

Figure 7.2: Discontinuous object after applying the decision rule. The small gaps will hinder the connected components procedure, so a preprocessing step is required.

If the connected components technique was applied to this image it would extract lots of small objects instead of a single one, which is not desirable. This can be easily solved by means of **morphological operations**, a common technique within the image processing field to effectively reduce noise and improve the quality of the image for subsequent stages.

7.2.1 Image Morphological Processing

A brief summary of the most common morphological operations is presented. These concepts are covered in more detail in [66]. A morphological operation is composed by:

- **Structuring element** or **kernel**. A binary matrix that represents the neighbourhood around the central pixel to be considered in the operation. For instance, a 3×3 kernel where all elements are 1 will take into account all the pixels in the 8-neighbourhood of the central pixel.
- **Morphological function**. It indicates the output value of the central pixel in relation to the pixels contained in the kernel.

The most popular morphological operations using a binary image I and a structuring element e are the following:

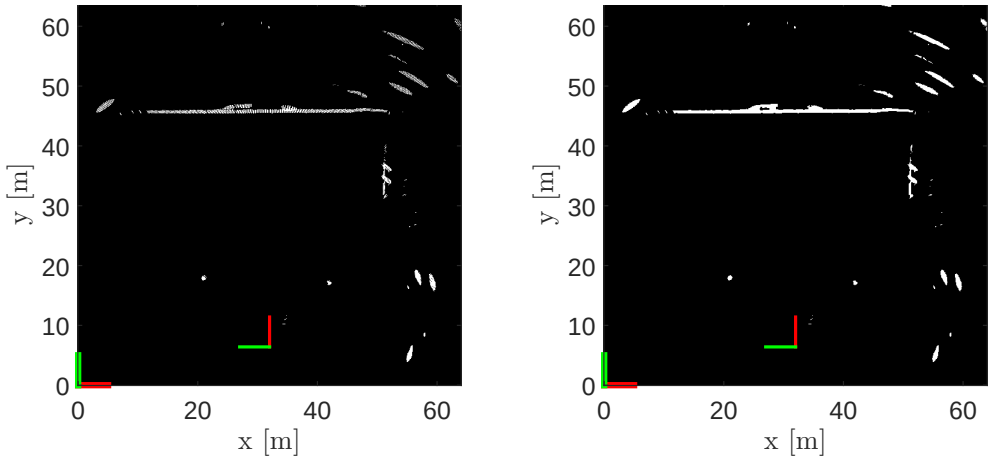
- **Dilation** ($I \oplus e$). The value at the central pixel is: $I = \max_{i,j \in e}(I_{ij})$.
- **Erosion** ($I \ominus e$). The value at the central pixel is: $I = \min_{i,j \in e}(I_{ij})$.
- **Closing**: $I \bullet e = (I \oplus e) \ominus e$.
- **Opening**: $I \circ e = (I \ominus e) \oplus e$.

The dilation and erosion operations make the objects thicker and thinner, respectively. The dilation operation is good to fill gaps inside objects and connect those that are close to each other. On the other hand, erosion removes small blobs, usually caused by sensor noise. If the original size of the objects is to be preserved, a closing or opening operation is preferred, since the resizing effects of the erosion and dilation operations are counteracted.

Taking this into consideration, we choose to apply a **closing** operation, which will effectively connect small blobs to form solid objects, since the dilation operation is performed first. This will greatly improve the results from the connected components stage afterwards. We use a square structuring element of size 3, so each pixel will get connected to the ones adjacent to it. This implies that the system will not be able to distinguish two objects separated by a distance of 1 cell = 10 cm. This is not a problem in a practical application: instead of detecting two objects, a single, large object will be detected, which should be avoided anyway.

7.2.2 Example

The result of the preprocessing is shown in Figure 7.3 from a test scenario.



(a) Decision grid

(b) Decision grid, post-processed

Figure 7.3: Morphological processing of the decision grid, prior to obstacle detection.

(a) Decision grid

(b) Decision grid, processed

Figure 7.4: Zoomed-in view of Figure 7.3 at $y = 45$ m. A continuous object is obtained.

It is interesting to observe the horizontal wall in front of the vehicle (see Figure 7.4). The small blobs are now fully connected to form a unique, solid object, which will be easily extracted as such afterwards.

7.3 Connected Components Extraction

After preprocessing the decision grid, objects are extracted from it. We observe that they tend to appear as blobs or clusters in the image, so we approach the problem via clustering techniques. As discussed in the Background chapter, the **connected components** algorithm is an efficient image processing technique for extracting clusters from a binary image. We implement it using the **flood fill algorithm**, as described in Algorithm 3. The result of the algorithm is a vector, each of its elements containing a list of the (u, v) image coordinates of the pixels belonging to the detected objects. A step-by-step example illustrating this iterative process is shown in Figure 7.5.

Algorithm 3 Connected Components Extraction: Flood Fill Algorithm

Input: `img` ▷ Binary image. Object pixels set to `true`
Output: `obj` ▷ List of objects

```
for each pixel p ∈ img do
    if img(p) == true then                          ▷ Found an object pixel
        Create object o(p)
        while p != null do
            img(p) ← false                          ▷ Do not count this pixel again
            p ← getValidNeighbours(p)               ▷ Neighbouring pixels where img is true
            o.add(p)                                ▷ Add them to the index list
        end while
    end if
    obj.add(o)                                      ▷ Add a new object
end for
```

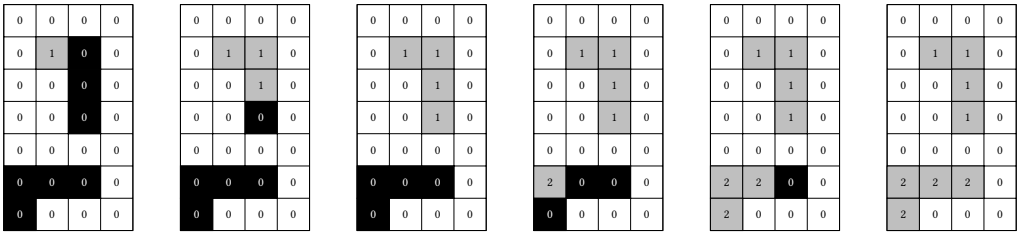


Figure 7.5: Flood-fill algorithm: step-by-step example. White: background pixels. Black: foreground (object) pixels. Gray: visited pixels. Each object is identified by a label: 0 (background), 1, 2 and so on.

7.4 High-Level Representation

Finally, a high-level representation of the objects is created. We are mainly interested in the geometry of the obstacle as well as its position in relation to the vehicle.This information is useful to filter spurious objects and better plan avoidance maneuvers. For every object we extract the following information, as depicted in Figure 7.6:

- **Centroid** of the obstacle (μ_x, μ_y).
- **Bounding box**, $\mathcal{B}$, which defines the maximum size of the object.
- **Confidence region**, defined as the 2σ ellipsoid around the object. This allows to treat the object detections in a probabilistic way in further computation stages.

All this information is always relative to the `vehicle` reference frame. Therefore, the (u, v) image coordinates must first be transformed into (x, y) points in `world` frame and afterwards to the `vehicle` frame. The following steps are applied to every cluster i to compute its geometrical information:

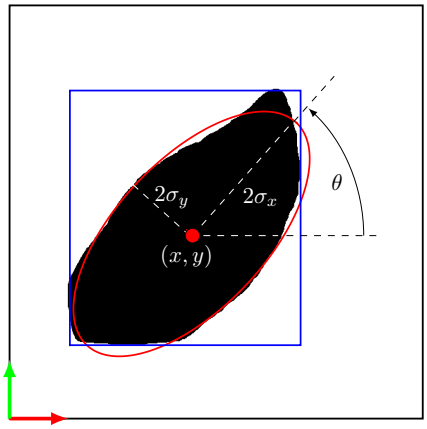


Figure 7.6: Obstacle representation. Red: confidence region. Blue: bounding box.

1. Input: the set of points S corresponding to the cluster i, extracted in the previous section. $S = (\mathbf{x}, \mathbf{y}), \mathbf{x} = \begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix}^\mathsf{T}, \mathbf{y} = \begin{pmatrix} y_1 & y_2 & \dots & y_n \end{pmatrix}^\mathsf{T}$

2. The centroid is computed by taking the mean over the position of all the pixels:

$$\mu_x = \frac{1}{n}\sum_{i=1}^{n} x_i \quad ; \quad \mu_y = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{7.1}$$

3. The confidence region is represented by the 2σ-ellipse. First, let $\overline{\mathbf{x}} = \mathbf{x} - \mu_x \quad ; \quad \overline{\mathbf{y}} = \mathbf{y} - \mu_y$. Then, the covariance matrix can be computed as follows:

$$C = \frac{1}{n-1}\sum_{i=1}^{n} \begin{pmatrix} \overline{x}_i^2 & \overline{x}_i\overline{y}_i \\ \overline{x}_i\overline{y}_i & \overline{y}_i^2 \end{pmatrix} = \frac{1}{n-1} \begin{pmatrix} \overline{\mathbf{x}}^\mathsf{T} \\ \overline{\mathbf{y}}^\mathsf{T} \end{pmatrix} \cdot \begin{pmatrix} \overline{\mathbf{x}} & \overline{\mathbf{y}} \end{pmatrix} \tag{7.2}$$

4. Afterwards, we compute the eigenvectors and eigenvalues of C, using the function `eig` in MATLAB. Note: it is assumed that the eigenvalues and the corresponding eigenvectors are sorted in descending order by the magnitude of the eigenvalue.

$$[V, \Lambda] = \texttt{eig}(C) \quad ; \quad V = \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{pmatrix} \quad ; \quad \Lambda = \begin{pmatrix} \lambda_1 & \lambda_2 \end{pmatrix}^\mathsf{T} \tag{7.3}$$

5. The largest eigenvalue represents the variance of the data in the direction of the mayor axis of the ellipse representing the points in the cluster. The final output of the algorithm provides the following data: obj $= (x, y, \theta, \sigma_x, \sigma_y)$, where:

$$x = \mu_x \quad ; \quad y = \mu_y \quad ; \quad \theta = \text{atan2}(v_{1_y}, v_{1_x}) \quad ; \quad \sigma_x = \sqrt{\lambda_1} \quad ; \quad \sigma_y = \sqrt{\lambda_2} \tag{7.4}$$

6. Finally, the bounding box $\mathcal{B}_i$ is defined as:

$$\mathcal{B}_i = \{p_x, p_y, w, h\} \tag{7.5}$$

where (p_x, p_y) represent the coordinates of the bottom-left corner of the bounding box, and w and h are the width and height, respectively:

$$p_x = \min_i x_i \quad ; \quad p_y = \min_i y_i \quad ; \quad w = \max_i x_i - p_x \quad ; \quad h = \max_i y_i - p_y \tag{7.6}$$

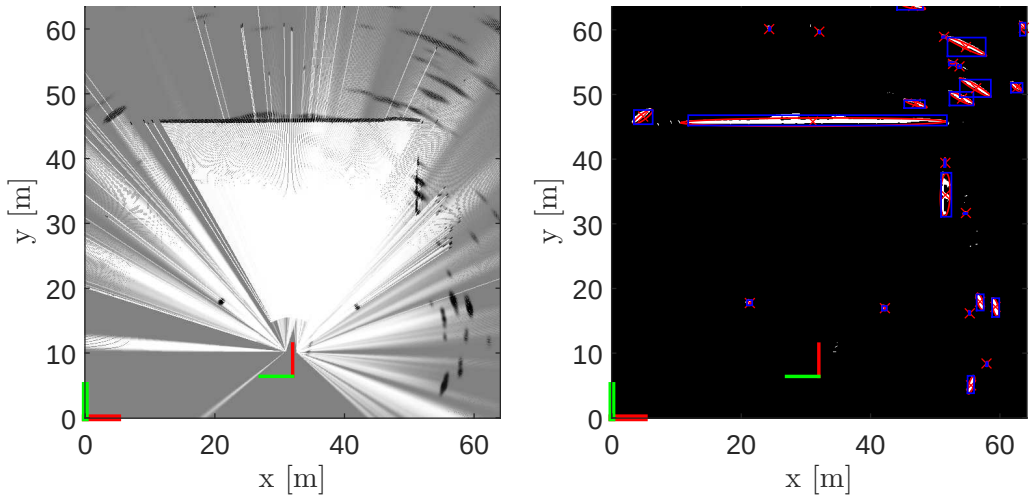
7.4.1 Object Filtering

The final step is to remove false or spurious detections, that do not belong to any real object. Many approaches to object recognition are based on shape detection, that work well to detect cars, pedestrians, etc. In this project, however, we consider objects of arbitrary shape, material and size, so these techniques cannot be applied. Instead, we filter out too small objects, that are likely to be caused by sensor noise. In particular, the following conditions are applied in order to consider a detection as a real object:

- The width and height of the bounding box must be greater than 0. This removes vertical and horizontal lines that are one pixel thick.
- The confidence region must have either of its axis larger than a given size: $\sigma_x > \sigma_{\min}$ or $\sigma_y > \sigma_{\min}$. This effectively removes small objects. We choose $\sigma_{\min} = 0.2$ m.

7.4.2 Example Results

The complete object detection pipeline is applied to our test scenario (see Figure E.1). The resulting objects are displayed in Figure 7.7.



(a) Fused grid

(b) Extracted obstacles

Figure 7.7: Detected obstacles from the fused grid.

The main extracted objects are the corner reflectors at around $(x, y) = (40, 20)$ m and $(20, 20)$ m respectively (objects 1 and 2 in Figure E.1). In addition, the front wall (3) at around $x = 45$ m is also extracted a single object. The parked truck (4) at $(x, y) = (50, 35)$ m is detected as two objects, but their bounding boxes cover the entire area occupied by it. Finally, there are many other detections that will just be ignored since they are too far away from the ego vehicle.

Chapter 8

Experimental Evaluation

In this chapter we present the experiments that have been carried out in order to evaluate the proposed implementation. The same experimental vehicle as described in Chapter 3 is used for all the experiments. The experiments vary in terms of the environment, the type and number of objects and also weather conditions, in order to analyze the robustness of the system.

8.1 Evaluation Metrics

The experimental evaluation is performed in a systematic way. For every scenario, we log data from every sensor while having the vehicle stopped. Given the large amount of data, we only record data for one minute to save disk space. Once the data has been collected, we run the object detection module offline, for both the Bayesian and Dempster-Shafer implementations. For faster simulations, we only process the first 30 seconds of each log file. Finally, we analyze the results of each simulation. We propose to use the following performance metrics for all the experiments:

- **Comparison to ground truth grid**, $\mathbf{m}^{GT}$, in cases where the Velodyne sensor is available. This is done qualitatively; it would however be possible to compute a quantitative measure, such as the distance between it and the **decision** grid, $\mathbf{m}^{D}$, for each sensor and the fused grid:

$$D = \sum_{i} \sum_{j} |m_{ij}^{D} - m_{ij}^{GT}| \tag{8.1}$$

  However, this measure would require a perfect alignment between the ground truth and the other grids, which we observed is not the case in our experiments due to a non-perfect sensor calibration. Therefore computing this measure D would not give useful results.

- **Entropy**, in order to measure certainty, as proposed by Moras *et al.* [60]. The entropy is a common indicator in Information Theory that represents the amount of information content of a random variable. The classical definition of entropy is

given as a function of the probability $p(x)$ of the random variable x:

$$H_p = -\sum_x p(x) \cdot \log_2 p(x) \tag{8.2}$$

A quick graphical analysis of this equation is presented in Figure 8.1:

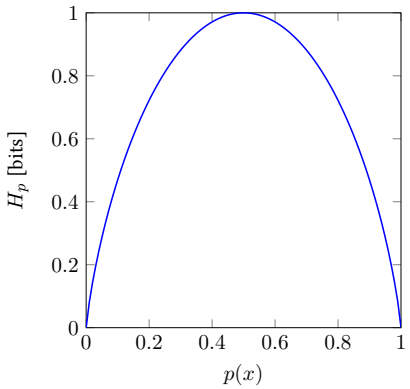


Figure 8.1: Entropy as a function of $p(x)$ for a binary random variable. The maximum entropy is obtained at $p(x) = 0.5$, equal to the unknown state.

It is clear that the entropy is maximum when the probability is 0.5, since in this case both states are equally likely. On the other hand, when the state is known with total certainty ($p(x) = 0$ or $p(x) = 1$) then the entropy is null: there is no uncertainty. This formula will be applied to the grids that are updated through the Bayesian inference framework.

A generalization of the entropy can be made for grids based on the Dempster-Shafer approach using belief masses instead, as shown by Yager [78]:

$$H_m = -\sum_{X \in 2^\Theta} m(X) \cdot \log_2(Pl(X)) \tag{8.3}$$

where $Pl(X)$ is the *plausability* of X, as shown in Section 4.2. However, it is not useful for comparison with respect to the classical entropy definition in this particular problem. For instance, consider a cell with probability of occupancy equal to 0.5. This would imply entropy equals 1 in the classical definition. The equivalent situation in the Dempster-Shafer theory would be $m(E) = m(O) = 0$, $m(\Theta) = 1$. If we compute the entropy, we realize that the final value is 0, because $Pl(\Theta) = 1$, as can be easily shown. Therefore, when there is belief mass in the unknown state the measures of entropy are not equivalent. To provide a better way of comparing the Bayesian and Dempster-Shafer approaches in terms of entropy, we will first compute the probability through the *pignistic transformation* and then apply Equation 8.2 (classical definition of entropy).

Finally, given the cell independence assumption, we can compute the **average entropy** per cell. This value will be computed at each iteration and the temporal evolution will be presented, for both the sensor and the fused grids.

- **Occupancy grids and detected objects**, which will also allow to qualitatively evaluate the performance of the fusion algorithm. The grids of the individual sensors as well as the fused grid will be presented. Objects will only be detected in the fused grid. In the case of the Dempster-Shafer grids, the pignistic transformation is applied to obstain probabilities from belief masses, which will allow for an easier comparison to the Bayesian approach. Note: the shown grids correspond to the time instant of **minimum entropy**, where the information content is maximum.

- Number of **false positives** (FP_o) and **false negatives** (FN_o) in terms of object detection (at a high level).

$$FP_o = \text{\#incorrect detections (false positives)} \tag{8.4}$$
$$FN_o = \text{\#incorrect misdetections (false negatives)} \tag{8.5}$$

  It is important to precisely define what we consider as false positives and false negatives. Given that data is usually quite noisy, we take a **practical criterion**. In particular, we will only analyze the regions of the grid that are close to the vehicle: 10 m to the left and right of the vehicle, and 50 m forward. This corresponds to the points in the range $20 \leq x \leq 40$, $10 \leq y \leq 60$ in global coordinates. The counting of objects is performed after extracting the high-level representation of obstacles.

  A *false negative* corresponds to the non-detection of an obstacle in the region described above. Similarly, a *false positive* is any detected object within this region that does not actually exist. In addition, if multiple small objects are extracted for the same physical object, those will be considered as a single object (not as false positives). Finally, false detections behind true detections will also not be considered: if the physical object was detected, we do not care if there was a false detection behind it; the car should avoid it anyway.

- **Computation time**. For each implementation (Bayes, Dempster-Shafer) the global computation time per iteration is obtained (not including the sensor data acquisition). Since it varies significantly, we present the minimum, maximum, average and standard deviation. The simulations are run on a laptop with a CPU Intel i5-4300U @1.90GHz (2.50 GHz boost) and 8 GB RAM.

Experiments

In the following sections, the performed experiments and their results are presented. The overall performance of the system will be discussed in Chapter 9. Pictures of the environment and the objects' placement can be found in Appendix E.

8.2 Test 1 (Scenario 1)

Description

The first experiment is a simple test case where we verify the detection capabilities of each sensor individually, placing a single object within each sensor's field of view. This is also useful to evaluate the capabilities of the system to fuse **complementary** sensor information.

- **Sensors**: corner radars, front lidar. The Velodyne lidar was not available, so a ground truth will not be generated for this experiment.
- **Objects**: see Figure E.1. Two corner reflectors (objects 1 and 2 in the figure), of RCS $=$ 20 dBsm, were placed approximately in the center of the field of view of their respective radar. In addition, there was a short concrete wall in front of the vehicle (3) and a parked truck to the right (4).
- **Environment**: an empty parking space. The vehicle was stopped at around 10 m from the corner reflectors during the whole experiment. Weather conditions: cloudy.

Sensor grids

The individual sensor occupancy grids are presented in Figures 8.2 and 8.3 for the Bayesian and Dempster-Shafer implementation, respectively.

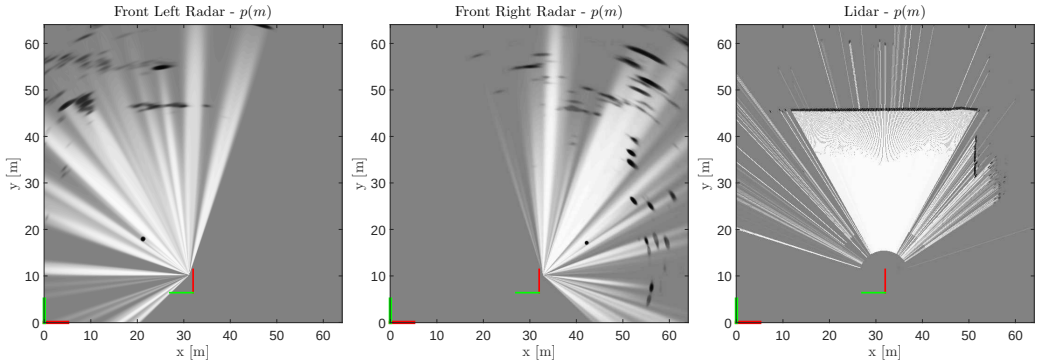


Figure 8.2: Test 1. Sensor occupancy grids. Bayesian implementation.

First, it can be observed that there are two regions of high probability of occupancy at approximately $(x, y) = (20, 20)$ m and $(40, 20)$ m in the left and right radar grids, which correspond to the corner reflectors (1) and (2), respectively. There are additional detections at larger distances, corresponding to other vehicles that were parked in nearby areas. The right radar returned detections belonging to the parked truck (4) at approximately $(x, y) = (50, 35)$ m. The lidar, on the other hand, detected mostly the short wall (3) in front of the vehicle. The typical Moirée effect appears for the empty space detected by the lidar at large distances, close to the wall. This is caused by the raycasting operation over a discrete grid. Furthermore, some detections of the truck (3) are returned by the lidar as well.

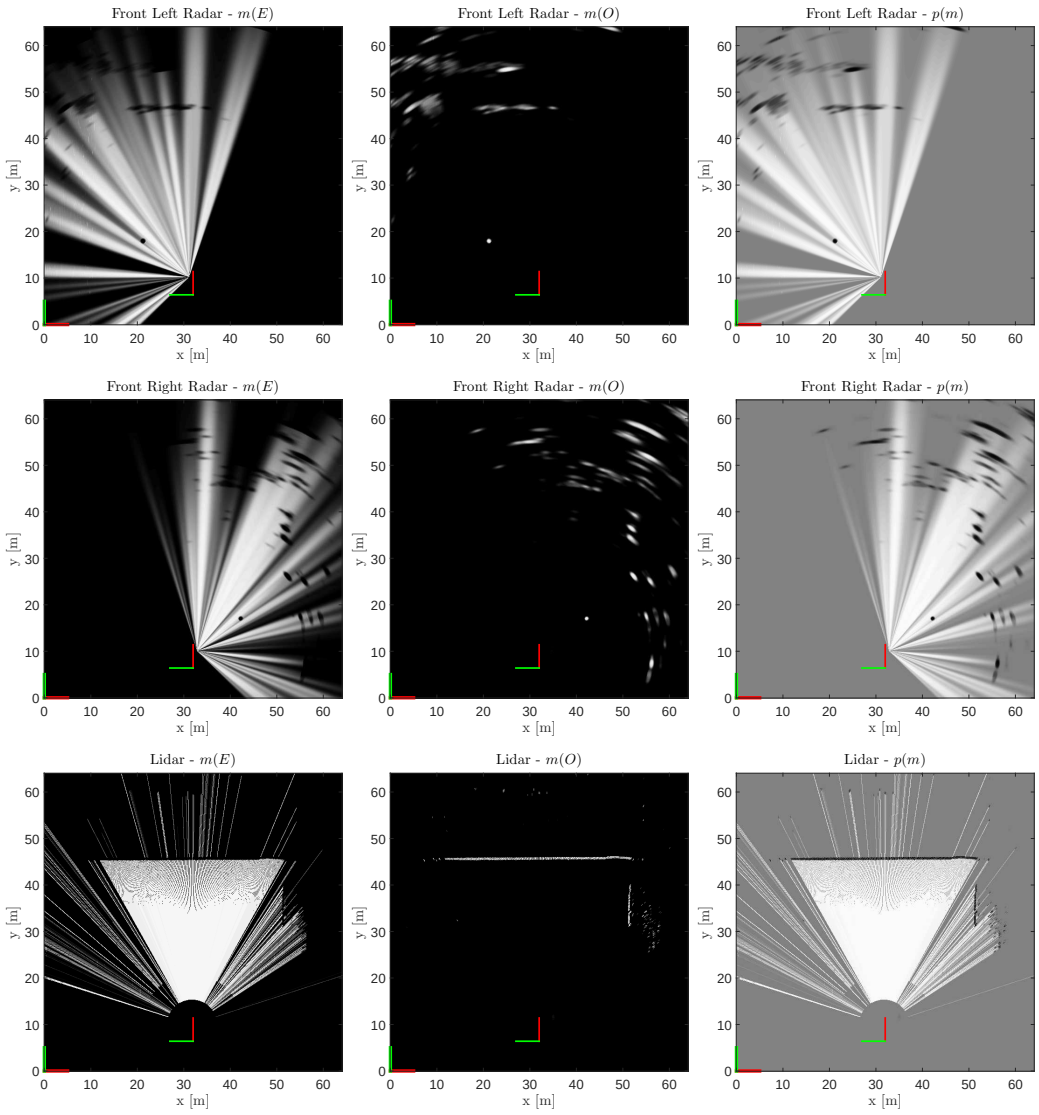
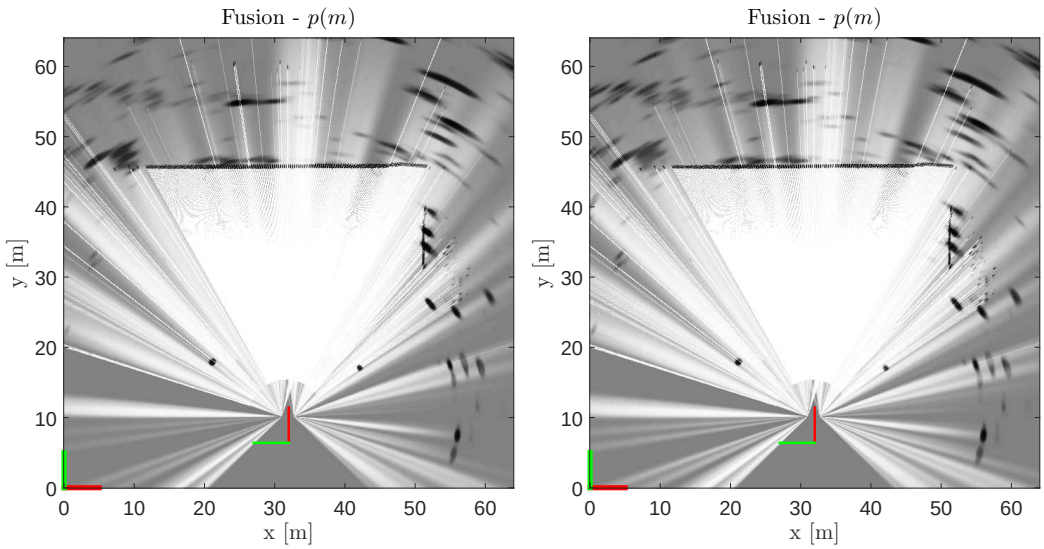


Figure 8.3: Test 1. Sensor occupancy grids. Dempster-Shafer implementation.

The detected objects can be more clearly observed in the Dempster-Shafer grids, as presented in Figure 8.3, since the `occupied` and `empty` states have their own belief mass. We can clearly see the four objects (corner reflectors, wall, truck) as main detections in the proximity of the vehicle. Finally, when comparing both implementations very little differences, if any, can be found from a qualitative point of view.

Fused grid

The results of performing sensor fusion are shown in Figure 8.4.

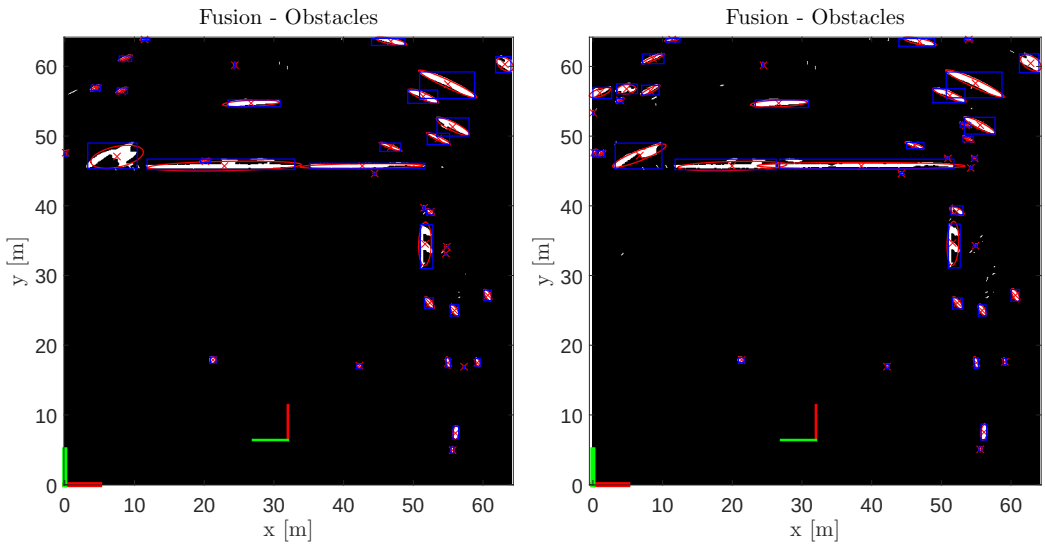


(a) Bayesian (b) Dempster-Shafer

Figure 8.4: Test 1. Fused grid.

Detected objects

Finally, the decision grid and the detected obstacles are shown in Figure 8.5.



(a) Bayesian (b) Dempster-Shafer

Figure 8.5: Test 1. Detected objects.

In addition, the false positive and false negative rates are presented in Table 8.1.

Framework	False positives	False negatives	Real number of objects
Bayesian	0	0	3
Dempster-Shafer	0	0	3

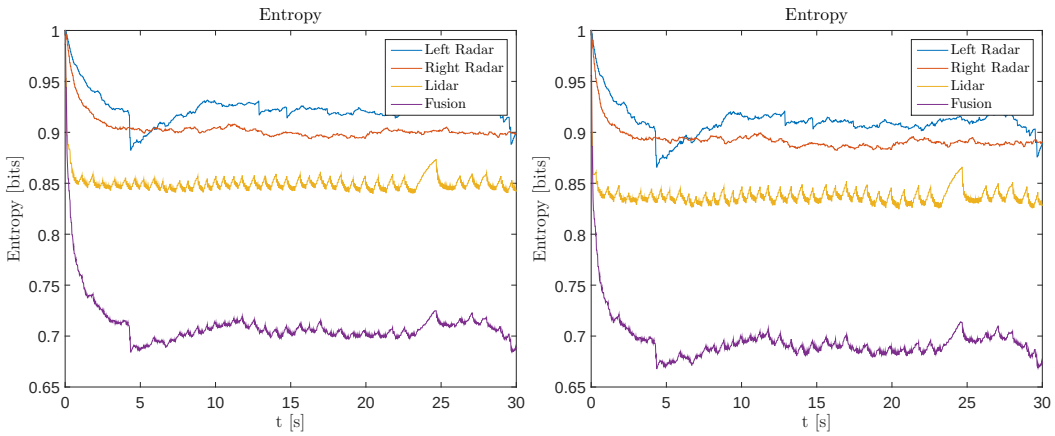
Table 8.1: Test 1. False positives and false negatives.

In this case, we only consider the two corner reflectors (1,2) and the wall (3) to be detected. We observe that a single object was successfully extracted for each of the corner reflectors. In the case of the wall, two objects were extracted; however, since they belong to the same object and the bounding boxes almost overlap this is not an issue in practice. Therefore, we have no false negatives. The remaining region was classified as free space, as it should, so there are no false positives. Even if it was not the goal of this experiment, the truck (4) was also correctly detected, although two clusters instead of one were extracted.

When comparing the Bayesian and Dempster-Shafer implementations we find subtle differences, mostly when sensors enter in conflict. The most noticeable appears in the front wall, at x = 30 m, where the radar empty space contradicts the occupied space detected by the lidar. After applying the decision rule, the Dempster-Shafer approach handles this situation much better. Even though the wall is split into two objects, the separation between them is smaller in the Dempster-Shafer grid. The opposite situation happens with the right corner reflector. In this case a lidar free-space detection overlaps with the occupied space detected by the radar. Now the Bayesian grid gives a larger likelihood, as can be observed in the fused grid. Nevertheless, the object is successfully detected in the decision grid in both cases.

Entropy

The average entropy of the grids over time is shown in Figure 8.6.



(a) Bayesian

(b) Dempster-Shafer

Figure 8.6: Test 1. Entropy.

In this case, we notice that the radars provide less information, mostly because the only objects that they detect are the corner radars. The lidar is more informative since it detects the wall and the free space to the car. As expected, by performing fusion we gather all the information and the total uncertainty in the grid is reduced, so it is more informative. It is also interesting to see how fast the entropy decreases, especially for the lidar, which reaches a stable point in a few hundred milliseconds. This shows that the proposed system is able to perceive the environment quite fast, which is essential for further obstacle avoidance actions.

Computation time

Last, the computation time per iteration is presented in Table 8.2.

Implementation	$t_{\min}$ (ms)	$t_{\max}$ (ms)	$t_{avg} \pm \sigma_t$ (ms)
Bayesian	100.13	1406.19	489.42 $\pm$ 213.24
Dempster-Shafer	400.28	3133.34	976.75 $\pm$ 364.44

Table 8.2: Test 1. Computation time per iteration.

The computation time varies quite a lot from iteration to iteration: almost an order of magnitude between the maximum and minimum time. This is related to the number of measurements returned by the sensors, especially the radar. On average, the Bayesian implementation is able to run under 500 ms, whereas in the Dempster-Shafer case it takes almost twice as much time, but still below 1 s on average.

8.3 Test 2 (Scenario 2)

Description

Here we evaluate the system capabilities for detecting different small objects.

- **Sensors**: corner radars, front lidar. The Velodyne lidar was not available, so a ground truth will not be generated for this experiment.
- **Objects**: see Figure E.2. Two corner reflectors (objects 1 and 7 in the figure), with RCS = 20 dBsm, were placed in a similar position as in Test 1. A third corner reflector (3), with RCS = 10 dBsm, was placed right in front of the vehicle. Finally, small objects such as cardboard boxes (2), backpacks (4,6) and plastic bags (5) were distributed in front of the vehicle.
- **Environment**: an empty parking space, as in Test 1. The vehicle was stopped at around 20 m from the corner reflectors during the whole experiment. However, in this experiment the objects were placed further away from the front wall (object number 3 in the previous test), to avoid detecting it with the sensors. Weather conditions: sunny.

Sensor grids

The individual sensor occupancy grids are presented in Figures 8.7 and 8.8 for the Bayesian and Dempster-Shafer implementation, respectively.

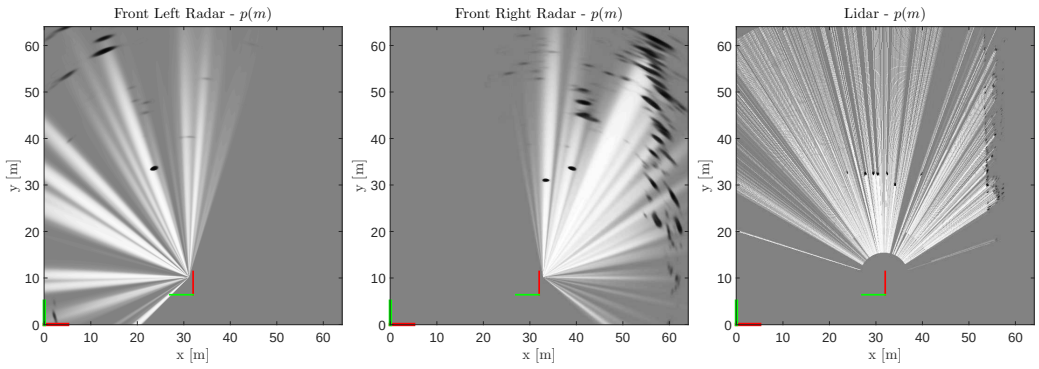


Figure 8.7: Test 2. Sensor occupancy grids. Bayesian implementation.

Starting with the front left radar, we observe a clear detection of the left corner reflector (7), at around $(x, y) = (22, 35)$ m. There are a few more detections belonging to parked cars further away, so free space area is also shown in the grid. The right radar returns more detections, as can be seen in the figure. The right corner reflector (1) is detected at $(x, y) = (38, 35)$ m, while the smaller corner reflector appears at approximately $(x, y) = (32, 32)$ m. There are multiple detections to the right of the vehicle, belonging to a small hill. The detections behind the right corner reflector do not actually belong to any placed object. Finally, the lidar sensor accurately detects all the objects (1-7). The backpacks (4,6) and plastic bags (5) are more clearly detected given their size. The detections are weaker

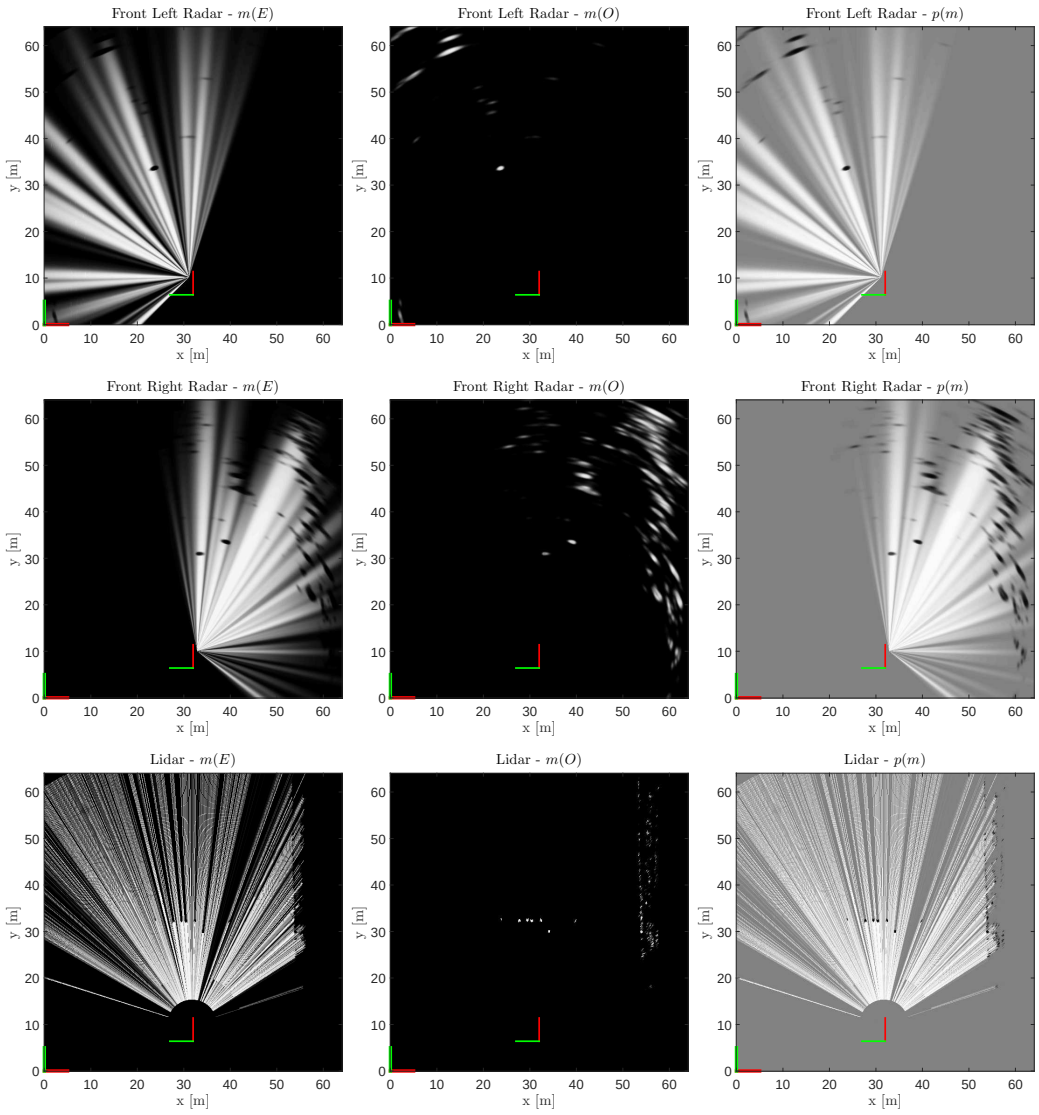


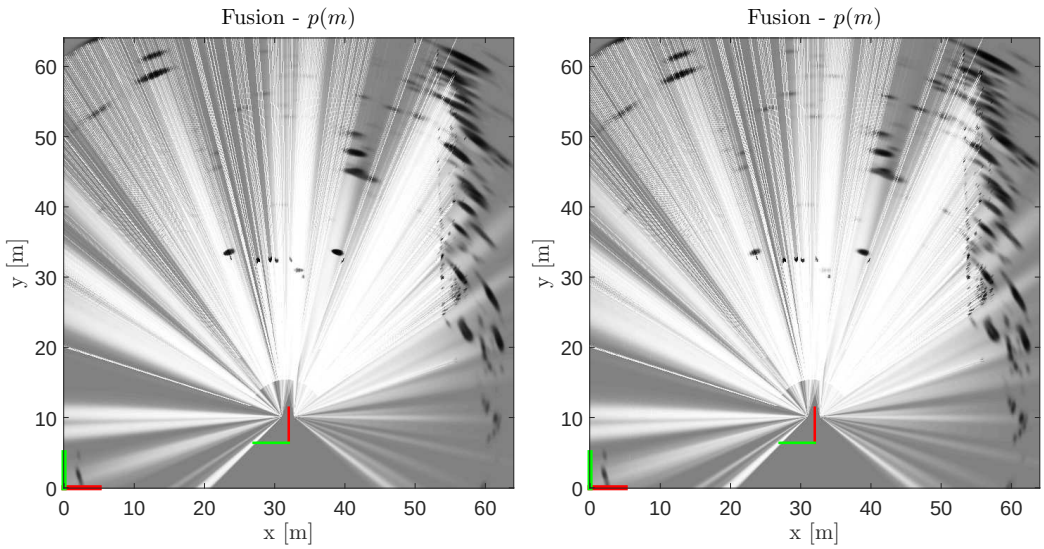
Figure 8.8: Test 2. Sensor occupancy grids. Dempster-Shafer implementation.

for the three corner reflectors (1,3,7), mostly due to the specular reflections they produce. We can see that even the small carton boxes (2) are captured by the lidar measurements, below $y = 30$ m. Similarly, the small hill to the right of the vehicle is also detected. It can also be seen that the front wall that was detected in Test 1 is not detected now, since the vehicle is further away from it than before.

The most noticeable difference between the Bayesian and Dempster-Shafer implementations can be observed in the right radar grid. The detection of the central corner reflector, at $(x, y) = (32, 32)$ m, has a smaller likelihood in the Dempster-Shafer grid. In this case, the conflict is produced by another radar beam that detected an object behind the corner reflector. The lidar grid does not have any conflicts since the beams do not overlap. Last, it is worth observing the grid Lidar - $m(O)$, where all the objects are clearly detected.

Fused grid

The results of performing sensor fusion are shown in Figure 8.9.

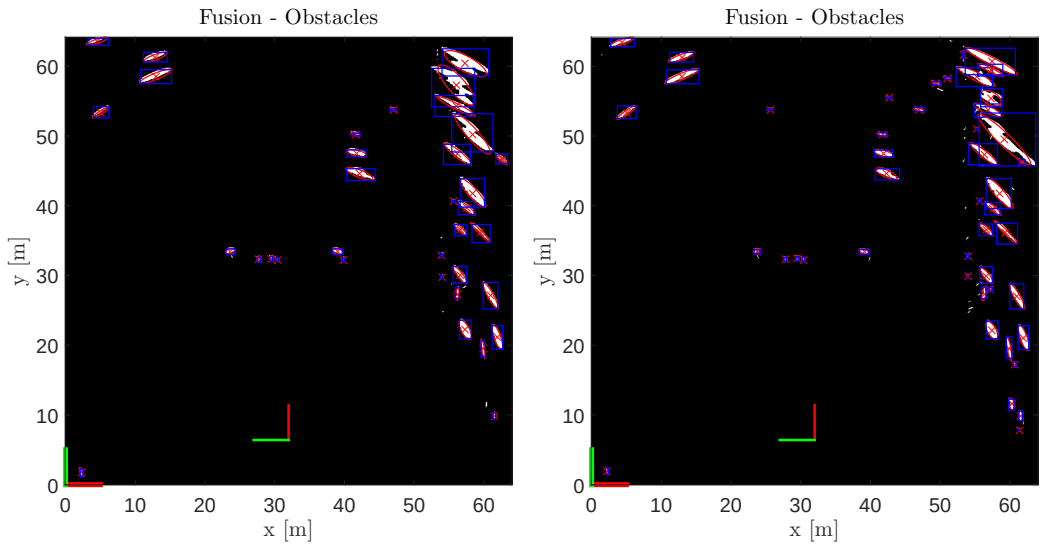


(a) Bayesian (b) Dempster-Shafer

Figure 8.9: Test 2. Fused grid.

Detected objects

Finally, the decision grid and the detected obstacles are shown in Figure 8.10.



(a) Bayesian (b) Dempster-Shafer

Figure 8.10: Test 2. Detected objects.

In addition, the false positive and false negative rates are presented in Table 8.3.

Framework	False positives	False negatives	Real number of objects
Bayesian	0	2	7
Dempster-Shafer	1	2	7

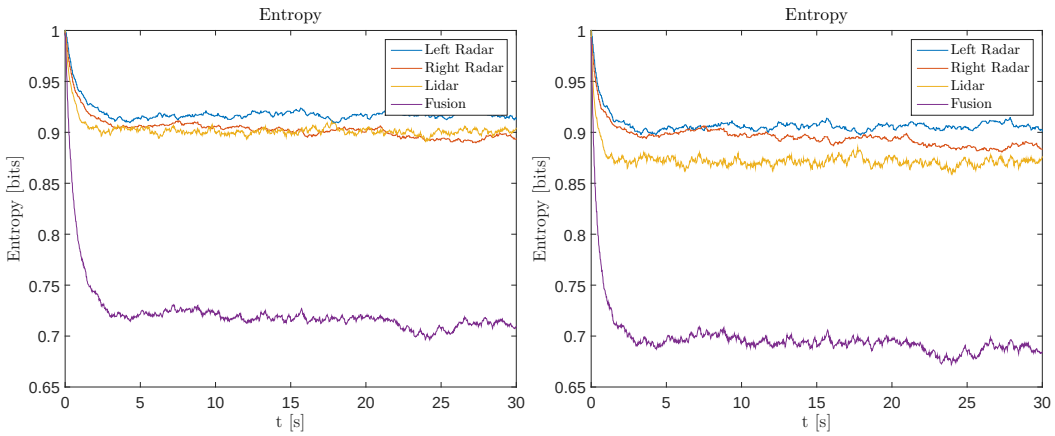
Table 8.3: Test 2. False positives and false negatives.

The goal in this experiment was to detect the seven objects that were placed in front of the vehicle. After performing sensor fusion and applying the decision rule, we observe that only five objects were successfully extracted (1, 4, 5, 6 and 7 in Figure E.3). Objects 2 and 3, which were the smallest ones, were not detected. In addition, there is a small false positive at around $(x, y) = (25, 55)$ m in the Dempster-Shafer decision grid. The additional detections to the right of the vehicle are not considered in the table, since they are too far from the vehicle.

When analyzing the fused grids (Figure 8.9), we observe some interesting differences between the Bayesian and Dempster-Shafer grids. First, we notice that the radar detection for the central corner reflector, at around $(x, y) = (32, 32)$ m, does not quite match the corresponding one for the lidar, so it seems like there are three objects in that region. This is caused by calibration errors. The Dempster-Shafer grid provides a better result since the lidar consistently **corrects** the radar measurement and sets that regions to free space. On the other hand, the detections belonging to the side corner reflectors have a slightly smaller likelihood in the Dempster-Shafer grid, but are nevertheless detected in the end. Finally, objects 2 and 3 were not detected because of the conflict between lidar and radar, which decreased the likelihood to a level below the decision threshold.

Entropy

The average entropy of the grids over time is shown in Figure 8.11.



(a) Bayesian

(b) Dempster-Shafer

Figure 8.11: Test 2. Entropy.

In this experiment we observe that a larger entropy for the individual sensor grids. This can be explained by the fact that now there is more empty space and therefore fewer detections. In Test 1, the front wall was detected and a large part of the grid was set to free space. As before, the fusion of information greatly reduces the total uncertainty on the grid, with a lower entropy. Last, we observe a slightly larger fall time, mainly because the objects were smaller and therefore harder to be detected. In any case, it still takes less than one second to obtain the maximum certainty.

Computation time

Last, the computation time per iteration is presented in Table 8.4.

Implementation	$t_{\min}$ (ms)	$t_{\max}$ (ms)	$t_{avg} \pm \sigma_t$ (ms)
Bayesian	146.30	1042.61	399.56 $\pm$ 210.74
Dempster-Shafer	401.43	3074.41	1085.89 $\pm$ 242.04

Table 8.4: Test 2. Computation time per iteration.

We can see many similarities with respect to the previous case, probably because it is the same environment. The computation time is slightly smaller compared to Test 1, mainly because the sensors return fewer measurements. Again, the Bayesian implementation is clearly superior in terms of efficiency, with an average computation time below 400 ms. The Dempster-Shafer implementation, on the other hand, takes a bit more than twice the time: around 1000 ms.

8.4 Test 3 (Scenario 3. Configuration 1)

Description

This experiment took place in a much larger open space area. This allows us to evaluate the performance of the system in almost ideal conditions, where the sensors should in theory only return measurements corresponding to the objects placed in front of the vehicle. The obstacles were larger than in the previous experiment, so they were more dangerous to crash into and therefore a good detection rate is required.

- **Sensors**: corner radars, front lidar, Velodyne lidar. Therefore, a ground truth will be generated for this experiment.
- **Objects**: see Figure E.3. Two corner reflectors (objects 1 and 8 in the figure), of RCS $=$ 20 dBsm, were placed approximately in the center of the field of view of their respective radar. A smaller corner reflector, of RCS = 10 dBsm, is placed in front of the vehicle, slightly to the left (looking from the vehicle). Finally, a variety of objects were distributed in front of the vehicle: wooden planks (2), a bicycle (3), a wooden box (4), two small tires (5, 9) and a big tire (7).
- **Environment**: a completely empty space of more than 50 m radius. The vehicle was stopped at around 15 m from the objects during the whole experiment. Weather conditions: moderate to intense **rain**.

Sensor grids

The individual sensor occupancy grids are presented in Figures 8.12 and 8.13 for the Bayesian and Dempster-Shafer implementation, respectively.

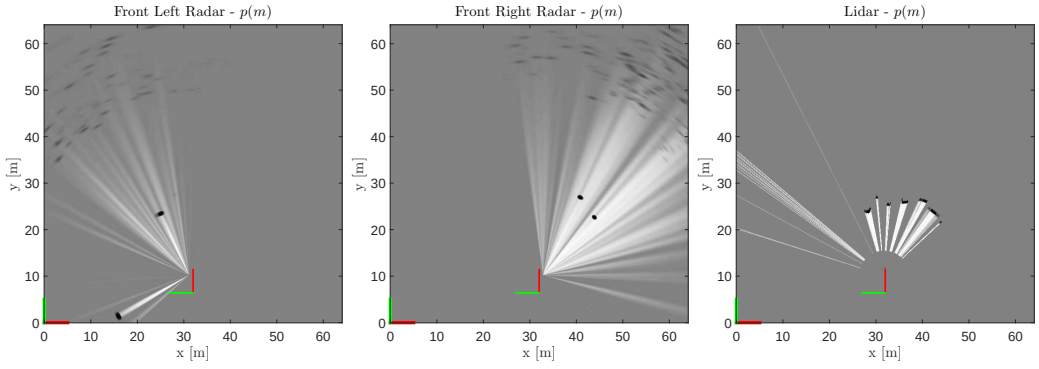


Figure 8.12: Test 3. Sensor occupancy grids. Bayesian implementation.

We start by analyzing the radar grids. Surprisingly, the left radar is only able to detect the 20 dBsm reflector (object 8), and not the 10 dBsm reflector (object 6), as we would have expected. There is also a strong measurement behind the vehicle, at $(x, y) = (15, 2)$ m, which most likely corresponds to another car that stopped by while we where performing the experiment. The right radar, on the other hand, detects the right corner reflector (1)

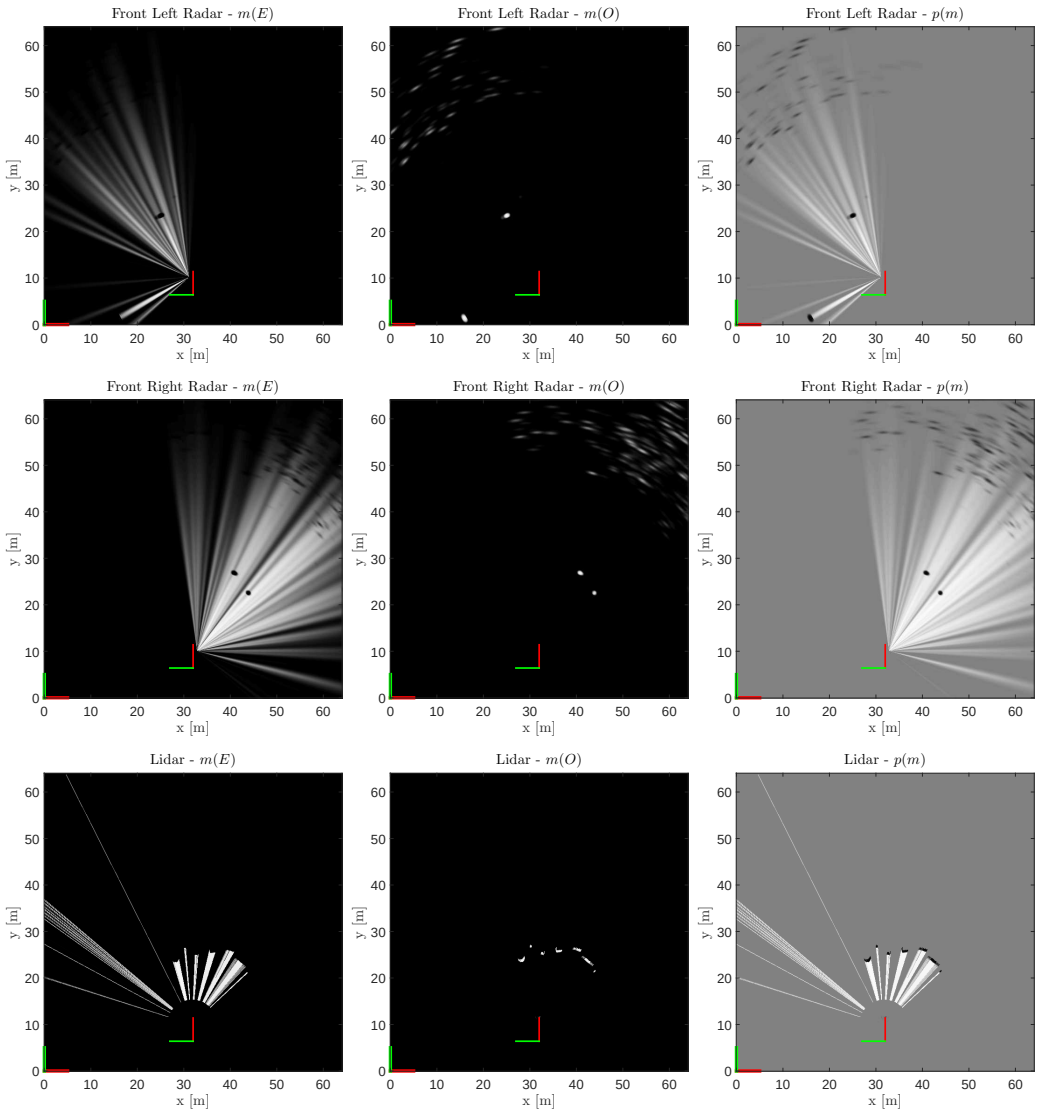


Figure 8.13: Test 3. Sensor occupancy grids. Bayesian implementation.

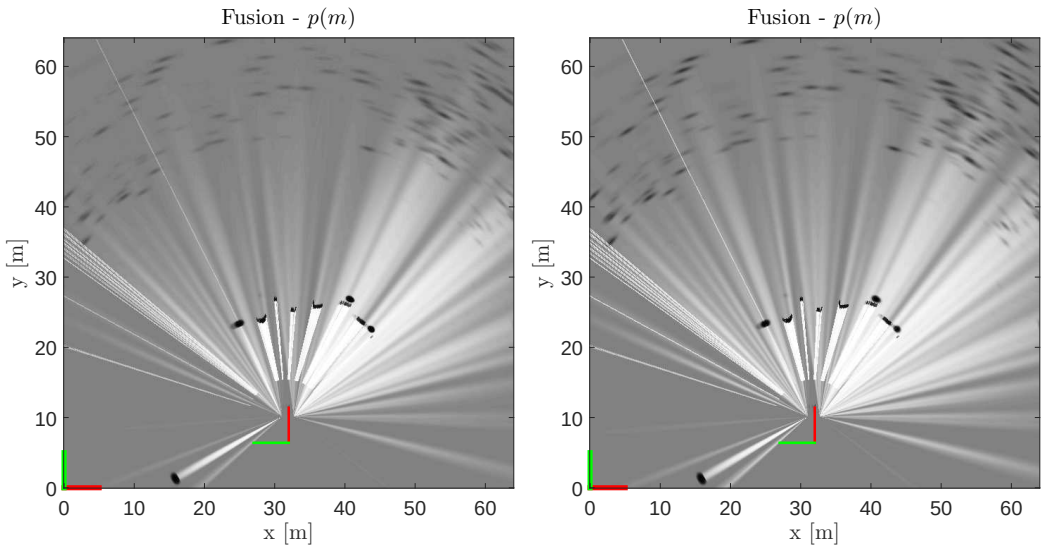
and the bicycle (3). In both radar grids there are multiple noisy detections at far distances. However, their likelihood is greatly reduced due to the effect of the **forgetting factor**, given that these measurements are usually spurious and non constant over time.

The lidar on the other hand has an excellent performance, detecting most of the objects. In particular, objects 1-7 were successfully detected. Only the left corner reflector and the small tire (8) were missed. We also checked the dimensions of the detected objects, and we observed that they match within a 0.1 m error to the result on the grid.

The Dempster-Shafer grids provide quite similar results, since there are very few conflicts. In the case of the radar we can say that the detections have a slightly larger likelihood. This can be clearly observed in the case of the right radar, for the detections at $(x, y) = (40, 30), (42, 25)$ m. This is caused by the overlapping free-space beams.

Fused grid

The results of performing sensor fusion are shown in Figure 8.14.

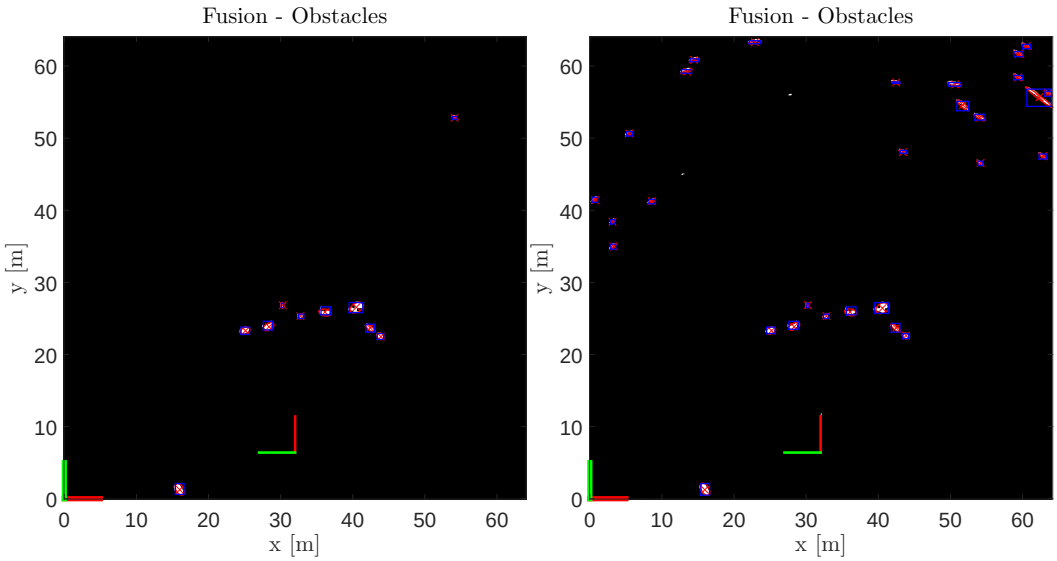


(a) Bayesian (b) Dempster-Shafer

Figure 8.14: Test 3. Fused grid.

Detected objects

Finally, the decision grid and the detected obstacles are shown in Figure 8.15.



(a) Bayesian (b) Dempster-Shafer

Figure 8.15: Test 3. Detected objects.

In addition, the false positive and false negative rates are presented in Table 8.5.

Framework	False positives	False negatives	Real number of objects
Bayesian	0	1	9
Dempster-Shafer	0	1	9

Table 8.5: Test 3. False positives and false negatives.

Observing the decision grid (Figure 8.15) we can clearly see that eight objects (1-8) were successfully detected. The only false negative, both in the Bayesian and Dempster-Shafer implementation, was the left tire (9), which was not detected by any sensor at all. We believe that the position and the low height of the object might be the reasons why the lidar could not detect it. The Velodyne lidar (see Figure 8.16) detected all the objects.

It is also interesting to analyze the fused grid (Figure 8.14). Similarly to Test 1, we observe how well the radar and lidar **complement** each other. In particular, the radars were able to detect the corner reflectors, while the lidar detected the rest of non-metallic objects. In this experiment we also notice that sensors can also provide **supporting information**, i.e. they detect the same object. For example, the bicycle (3) was detected by both sensors, at $(x, y) = (40, 28)$ m, approximately. The likelihood is therefore slightly larger in this region, which makes it easier for the detection stage to to detect the object.

Last, we also notice that the noisy measurements far away, behind the obstacles, more effectively filtered out in the Bayesian grid, with only one detection at around $(x, y) = (55, 55)$ m. On the other hand, many more noisy detections appear in the Dempster-Shafer grid. They are not counted as false positive because they are outside our interest area, and behind the actual obstacles.

Ground truth

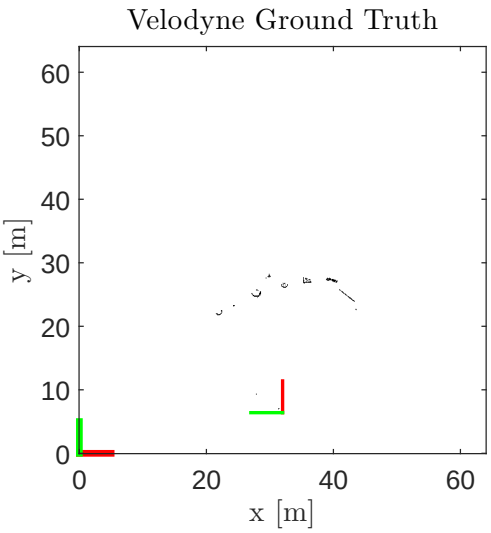
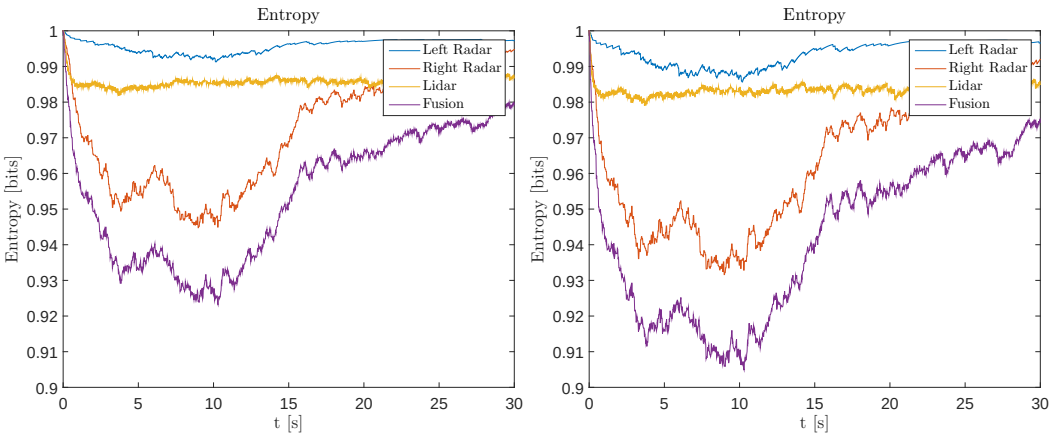


Figure 8.16: Test 3. Ground truth

Entropy

The average entropy of the grids over time is shown in Figure 8.17.



(a) Bayesian (b) Dempster-Shafer

Figure 8.17: Test 3. Entropy.

In this experiment we get the largest average entropy, mostly because the sensors return measurements only for the small objects that we place; everything else in the grid is unexplored. The least informative sensor is the left radar, since it only detects the left reflector. It is interesting to notice how **stable** the lidar is, despite the raining conditions. As mentioned throughout the report, the multi-echo capabilities allow it to get correct measurements in this case. The right radar, on the other hand, is much noisier.

Computation time

Last, the computation time per iteration is presented in Table 8.6.

Implementation	$t_{\min}$ (ms)	$t_{\max}$ (ms)	$t_{avg} \pm \sigma_t$ (ms)
Bayesian	100.08	548.75	204.02 $\pm$ 77.03
Dempster-Shafer	400.83	883.51	517.68 $\pm$ 83.68

Table 8.6: Test 3. Computation time per iteration.

In this we obtain the lowest computation times, although still far from real-time operation. The Bayesian implementation takes on average around 200 ms per iteration, whereas the Dempster-Shafer variant takes a bit more than 500 ms. The reduction in computation time is related to the small number of measurements to be processed, especially in the case of the radar. In any case, the maximum computation time is lower than 900 ms, which is a great difference with respect to the previous tests. The variance is also greatly reduced compared to the other experiments: around 80 ms.

8.5 Test 4 (Scenario 3. Configuration 2)

Description

We repeated the previous experiment, this time placing the vehicle further away from the obstacles. By evaluating the accuracy and speed of the algorithm we can determine the degree of anticipation that the vehicle will have when detecting far obstacles.

- **Sensors**: corner radars, front lidar, Velodyne lidar. Therefore, a ground truth will be generated for this experiment.
- **Objects**: see Figure E.3. Two corner reflectors (objects 1 and 8 in the figure), of RCS $=$ 20 dBsm, were placed approximately in the center of the field of view of their respective radar. A smaller corner reflector, of RCS = 10 dBsm, was placed in front of the vehicle, slightly to the left (looking from the vehicle). Finally, a variety of objects were distributed in front of the vehicle: wooden planks (2), a bicycle (3), a wooden box (4), two small tires (5, 9) and a big tire (7).
- **Environment**: a completely empty space of more than 50 m radius. The vehicle was stopped at around 40 m from the objects during the whole experiment. Weather conditions: moderate to intense rain.

Sensor grids

The individual sensor occupancy grids are presented in Figures 8.18 and 8.19 for the Bayesian and Dempster-Shafer implementation, respectively.

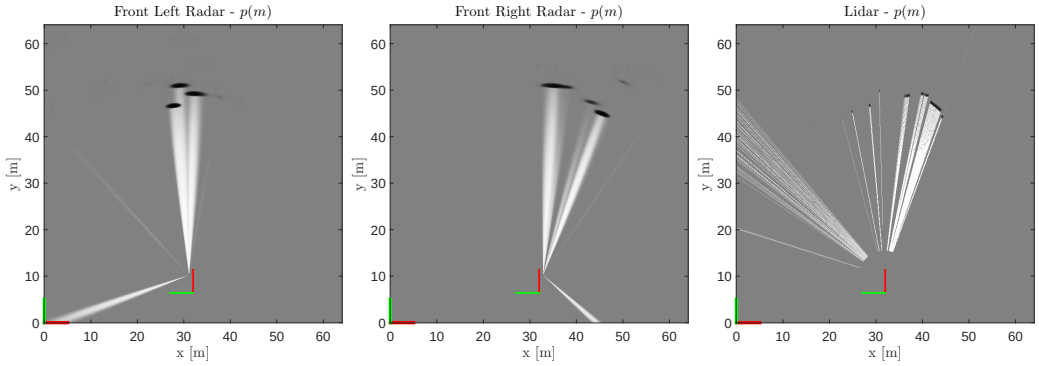


Figure 8.18: Test 4. Sensor occupancy grids. Bayesian implementation.

The first thing that we immediately observe in the sensor grids is that the number of detections is greatly reduced, especially for the radars, which pretty much only return measurements belonging to the placed objects. The large distance also increases the uncertainty in the detections, which results in wide detection cones that span a couple of meters when they reach the objects. It is therefore hard to associate detections to objects. The front corner reflector is detected by both radars at around $(x, y) = (32, 50)$. The left and right reflectors are also detected by each radar, respectively. There is a detection at $(x, y) = (27, 52)$ m in the left radar grid that could belong to the tire (8), but the range

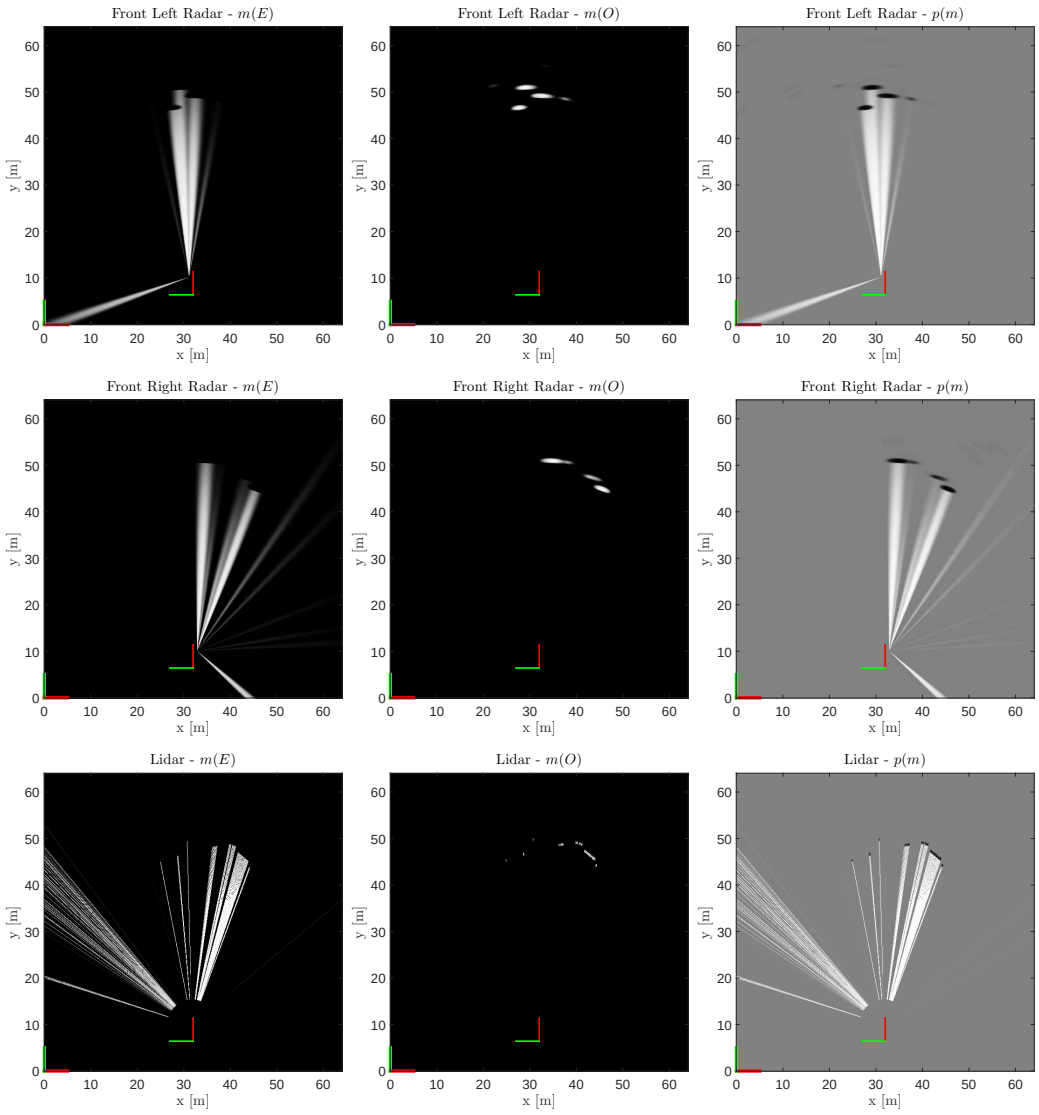


Figure 8.19: Test 4. Sensor occupancy grids. Dempster-Shafer implementation.

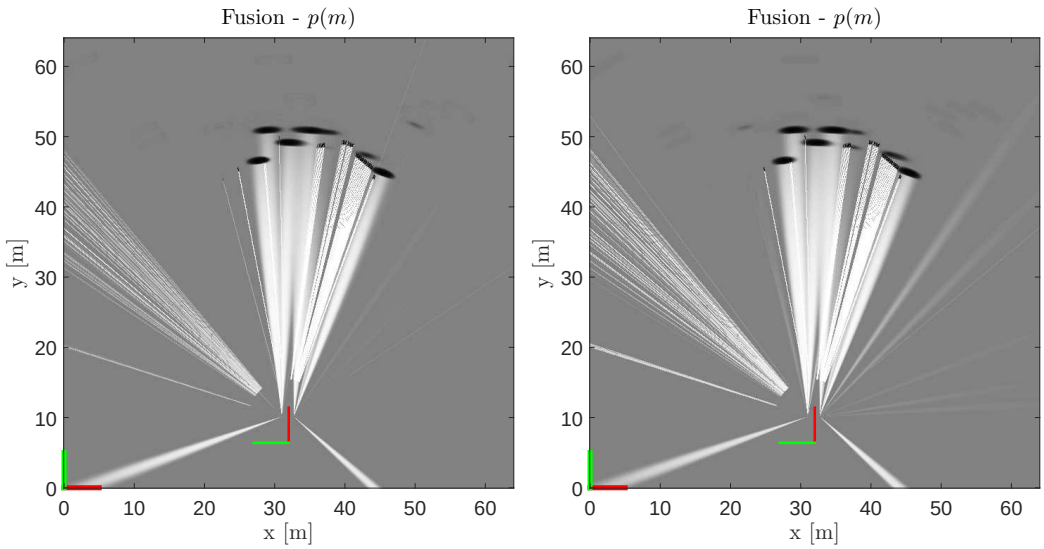
is not correct, according to Figure E.3. The bicycle is also detected by the right radar, although with smaller likelihood.

The lidar keeps detecting most of the objects (all of them except for the front tire, number 5), but the angular resolution is a bit poor at such large distances. Indeed, the smallest objects (6-9) are detected by only one or two laser beams.

Since there are almost no conflicts in this experiment, the Dempster-Shafer and Bayesian grids are pretty similar, and no significant differences worth mentioning can be observed.

Fused grid

The results of performing sensor fusion are shown in Figure 8.20.

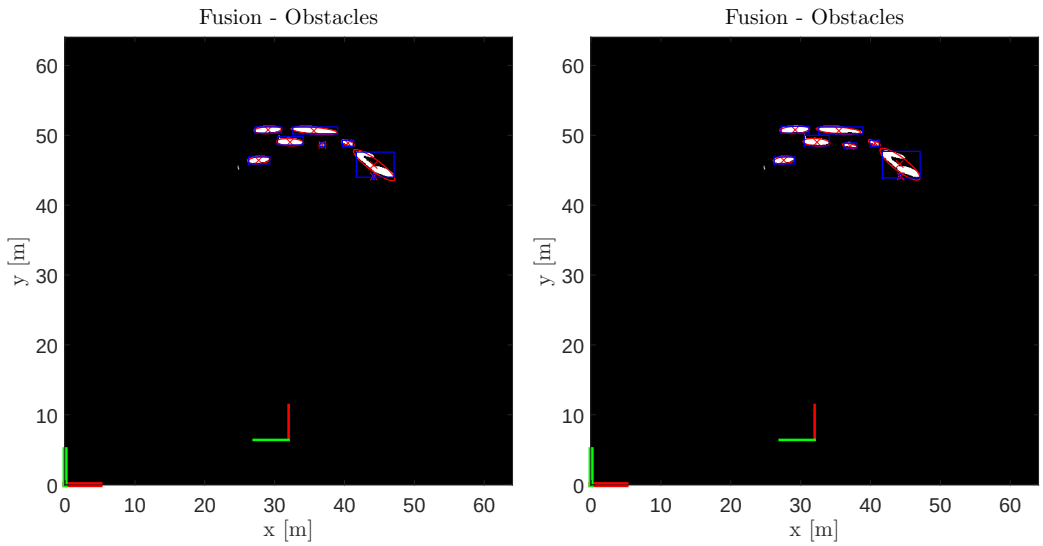


(a) Bayesian (b) Dempster-Shafer

Figure 8.20: Test 4. Fused grid.

Detected objects

Finally, the decision grid and the detected obstacles are shown in Figure 8.21.



(a) Bayesian (b) Dempster-Shafer

Figure 8.21: Test 4. Detected objects.

In addition, the false positive and false negative rates are presented in Table 8.7.

Framework	False positives	False Negatives	Real number of objects
Bayesian	1	2	9
Dempster-Shafer	1	2	9

Table 8.7: Test 4. False positives and false negatives.

Analyzing the decision grid from right to left, we first observe that objects (1,2) are detected as a single large object, with centroid at around $(x, y) = (43, 45)$ m. We do not count this a one false positive since the size of the group of objects is correctly determined and therefore it is not a risk for the vehicle. The bicycle (3) and the wooden box (4) are also correctly detected, thanks to the lidar sensor.

Four additional objects to the left of the wooden box are extracted from radar detections. By carefully analyzing the radar and lidar grids, and also the environment picture (Figure E.4), we reach the conclusion that the detections at approximately $(x, y) = (27, 47), (28, 52)$ and $(30, 50)$ m belong to objects 7, 6 and 5, respectively. The last detection, at around $(x, y) = (35, 52)$ m cannot be associated to any physical object, so we count it as a false positive. We can also notice that objects 8 and 9 are missing, which makes two false negatives. It is worth mentioning that **not even the Velodyne** is able to detect all of the left-most objects (see Figure 8.22).

As before, the Dempster-Shafer and Bayesian implementations provide exactly the same result in terms of detected obstacles. Last, by analyzing the fused grids we realize the the lidar and radar measurements do not match as good as at small distances. This again is caused by an imperfect sensor extrinsic calibration.

Ground truth

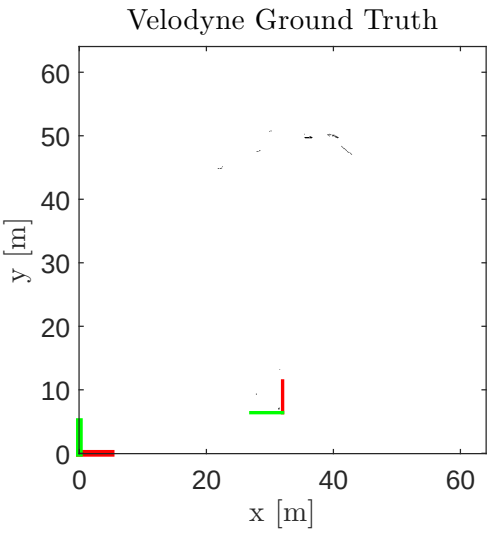
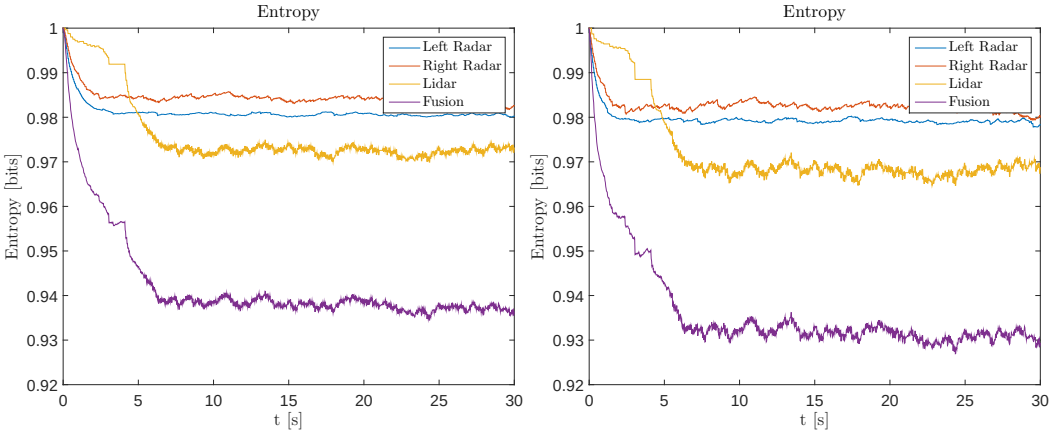


Figure 8.22: Test 4. Ground truth.

Entropy

The average entropy of the grids over time is shown in Figure 8.23.



(a) Bayesian (b) Dempster-Shafer

Figure 8.23: Test 4. Entropy.

Similarly to Test 3, the entropy is very high given the small number of measurements returned by the sensors. It is even larger since fewer objects are successfully detected. The most important thing to highlight is the quite **high transition time for the lidar grid**, which takes more than five seconds to reach the stable level. This can be explained by the fact the left-most objects were usually detected by only one beam, therefore a not very consistent and robust measurements over time. The rain might have also affected the detection rate, since many measurements were probably filtered internally by the sensor. We can conclude from this experiment that for large distances the radar will provide a faster and more robust detection rate than the lidar, although much less accurate.

Computation time

Last, the computation time per iteration is presented in Table 8.8.

Implementation	$t_{\min}$ (ms)	$t_{\max}$ (ms)	$t_{avg} \pm \sigma_t$ (ms)
Bayesian	100.01	283.21	162.00 $\pm$ 33.89
Dempster-Shafer	400.07	645.35	435.89 $\pm$ 40.79

Table 8.8: Test 4. Computation time per iteration.

We observe little significant differences with respect to the previous experiment. Since even fewer detections were returned by the sensors the computation time is even smaller than in the previous experiment, under 300 ms for the Bayesian implementation and 650 ms in the Dempster-Shafer case. The Dempster-Shafer implementation manages to get under 500 ms per iteration on average, but it is still more than twicer slower than the Bayesian one. Moreover, the variance is greatly reduced: around 40 ms.

8.6 Test 5 (Scenario 4. Configuration 1)

Description

This experiment was performed in a **city-like** environment. This allows us to evaluate the performance of the system in urban scenarios, usually harder to analyze given the wide variety of objects, buildings and structures that appear.

- **Sensors**: corner radars, front lidar, Velodyne lidar. Therefore, a ground truth will be generated for this experiment.
- **Objects**: see Figure E.4. Two corner reflectors (objects 1 and 8 in the figure), of RCS = 20 dBsm, were placed at the edges of the road, close to the buildings. A smaller corner reflector, of RCS = 10 dBsm, was placed in front of the vehicle, slightly to the right (looking from the vehicle). Finally, a variety of objects were distributed in front of the vehicle: a bicycle (2), a wooden box with planks on top of it (3), two small tires (4,7) and a big tire (5).
- **Environment**: a road intersection in a city-like environment. The road was 20 m wide approximately. The obstacles were placed at around 20 m from the vehicle. Weather conditions: light rain.

Sensor grids

The individual sensor occupancy grids are presented in Figures 8.24 and 8.25 for the Bayesian and Dempster-Shafer implementation, respectively.

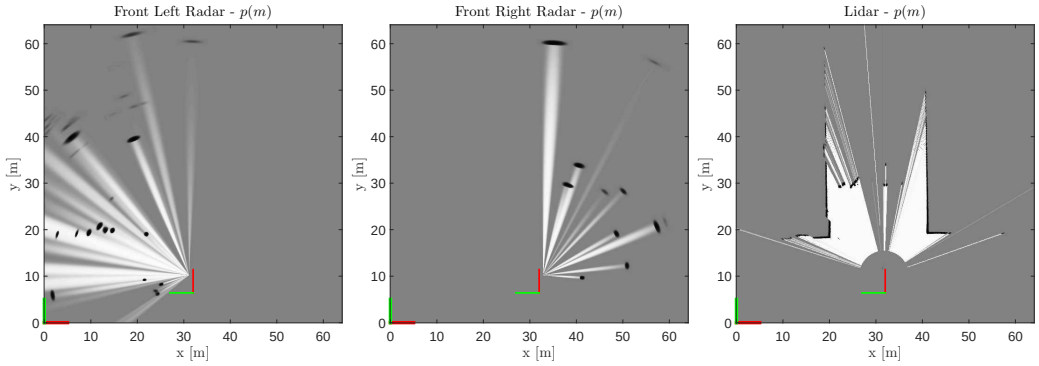


Figure 8.24: Test 5. Sensor occupancy grids. Bayesian implementation.

The first thing we notice when analyzing the sensor grids is the large number of measurements obtained for all the sensors. This proves the higher complexity of urban scenarios. We also realize how noisy the radar measurements are for either of the radars. Some measurements do correspond to objects. For instance, the left radar returns a few detections over $y = 20$ m, that belong to one of the walls, as can be observed in the ground truth (Figure 8.28). A sign post was also detected, at $(x, y) = (20, 20)$ m, but unfortunately not the left corner reflector (1). However, there are many detections that do not correspond to the true environment, like the ones over $(x, y) = (25, 10)$ m (except for the left-most

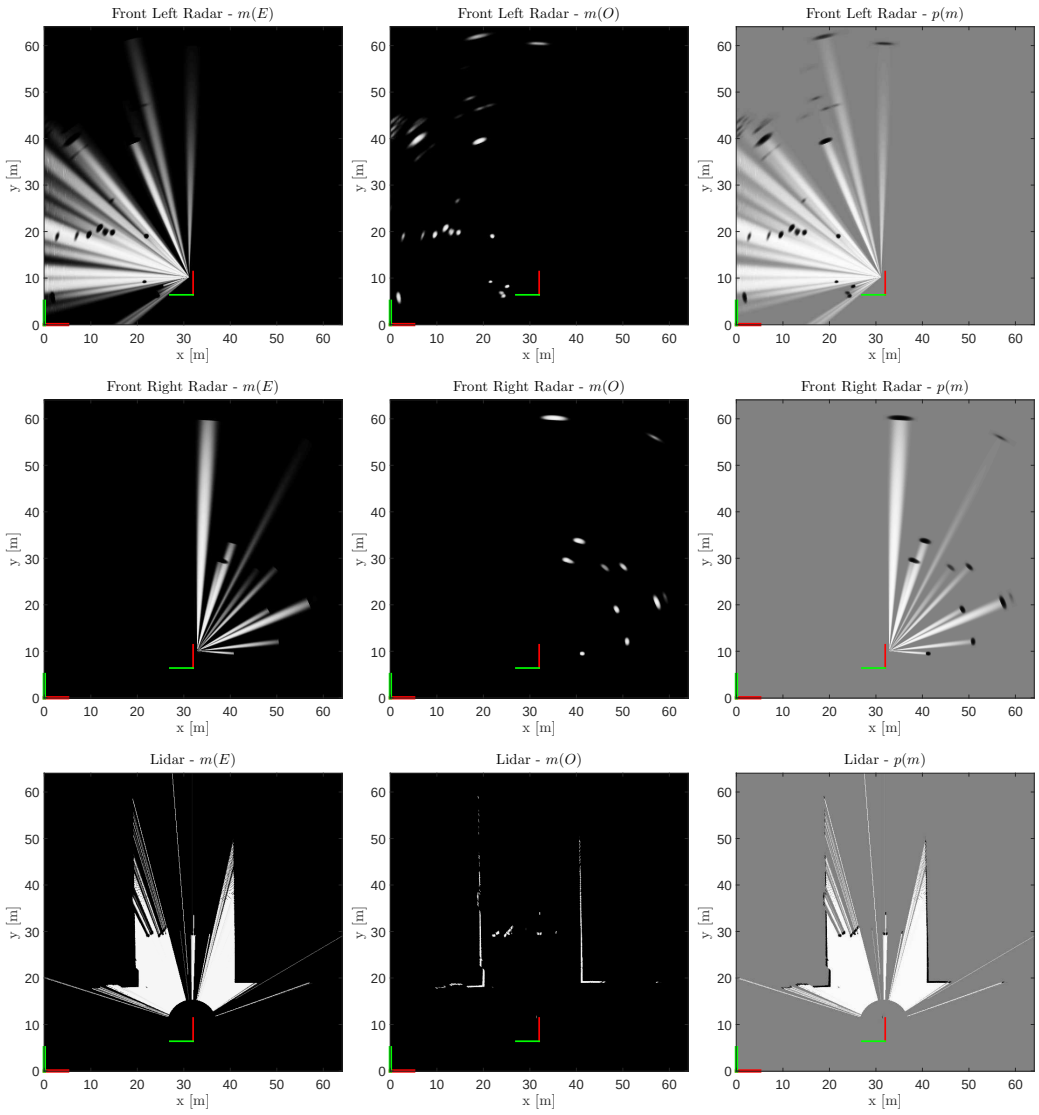


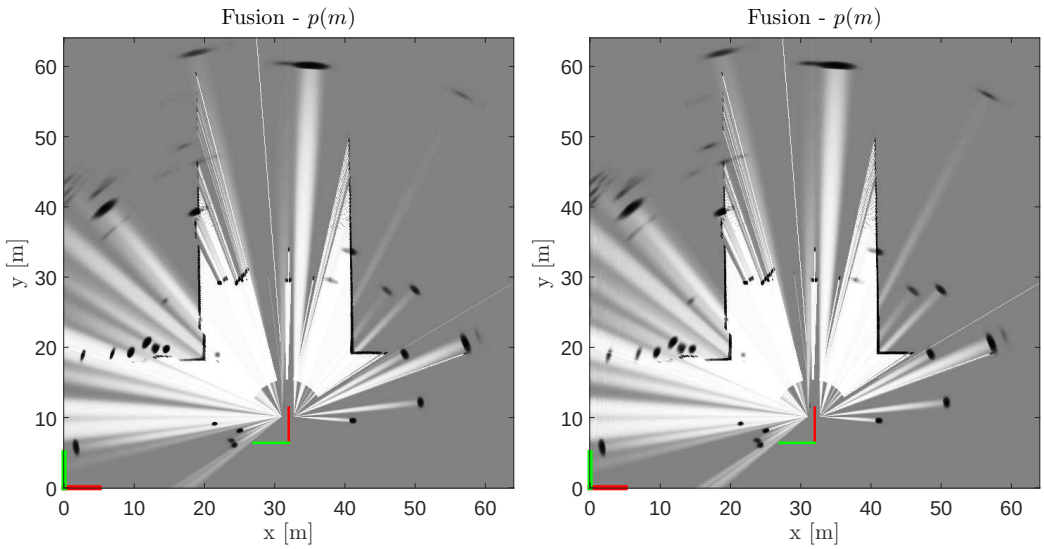
Figure 8.25: Test 5. Sensor occupancy grids. Dempster-Shafer implementation.

one, which belongs to a sign post). There are also many measurements that seem to detect objects through the fake walls. A similar situation can be observed with the right radar, although it does detect the right corner reflector (8), at $(x, y) = (35, 30)$ m.

On the other hand, the lidar grid provides much more accurate and reliable detections, with a similar performance to the Velodyne sensor. It was able to clearly detect the bicycle (2) at $(x, y) = (22, 30)$ m, the wooden box (3) next to it and the big tire at $(x, y) = (32, 30)$ m. The smaller tires were harder to detect: only one beam detected each of them. Last, it is interesting to observe that even the front corner reflector (6), behind the big tire was also detected, thanks to the **multi-layer** technology. As in the previous experiments, no significant differences between the Bayesian and Dempster-Shafer grids can be observed, due to the low number of conflicts.

Fused grid

The results of performing sensor fusion are shown in Figure 8.26.

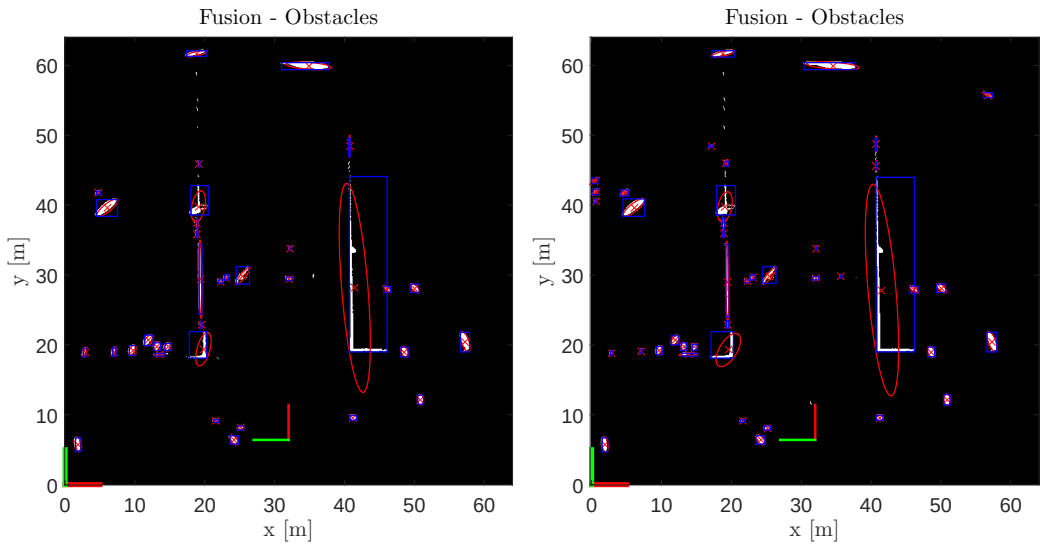


(a) Bayesian (b) Dempster-Shafer

Figure 8.26: Test 5. Fused grid.

Detected objects

Finally, the decision grid and the detected obstacles are shown in Figure 8.27.



(a) Bayesian (b) Dempster-Shafer

Figure 8.27: Test 5. Detected objects.

In addition, the false positive and false negative rates are presented in Table 8.9.

Framework	False positives	False negatives	Real number of objects
Bayesian	2	4	8
Dempster-Shafer	2	3	8

Table 8.9: Test 5. False positives and false negatives.

From the decision grids, we first count two false positives to the left of the vehicle. The left-most detection is a true detection of a post sign, as well as the one at around $(x, y) = (42, 10)$ m. Two obstacles are extracted for the bicycle. The wooden box (3), the big tire (5) and the small corner reflector (6) are also accurately detected. We notice the first difference between the Bayesian and Dempster-Shafer decision grids: the latter managed to detect the right-most tire. The walls were also extracted as objects, although the left wall was divided into multiple, smaller objects. Finally, the false negatives are: the corner reflectors (1,8) and the small tire (4). The Velodyne sensor managed to detect all the objects except for the right corner reflector (see Figure 8.28).

Interesting information can be obtained from the fused grids. We find situations where sensors **correct each other** to obtain a more accurate map. For instance, both the lidar and the radar detect the right wall at $(x, y) = (40, 35)$ m. However, the lidar refines the free-space area that was marked as occupied by the radar. However, we observe many cases of **conflicting information**. For instance, the right corner reflector is not detected because the lidar detected a wall behind it and considered that region as free space. A similar case appears with the sign post at $(x, y) = (22, 20)$ m. We leverage the conflict information to improve the results in this scenario in Appendix A.

Ground truth

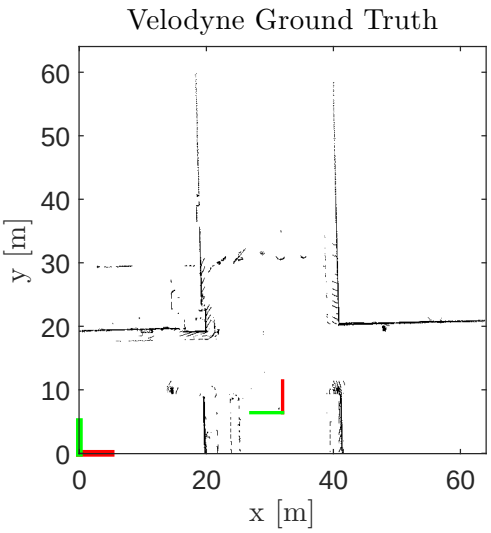
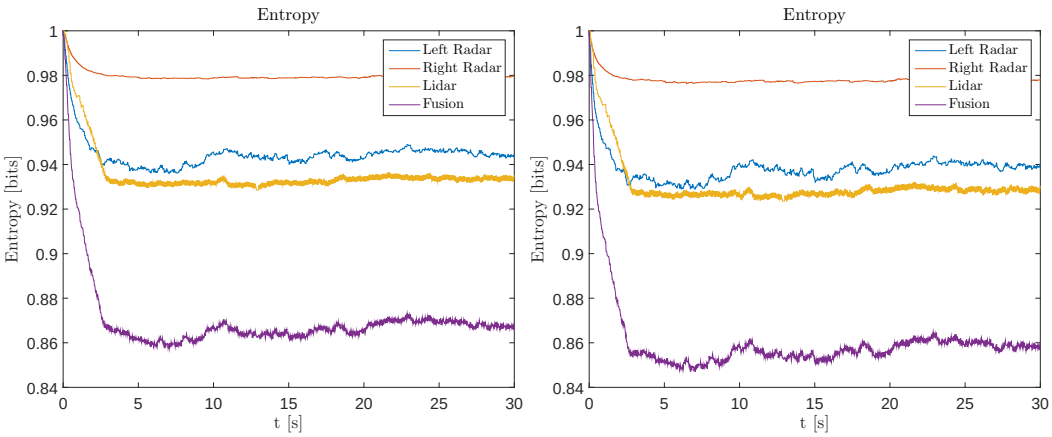


Figure 8.28: Test 5. Ground truth.

Entropy

The average entropy of the grids over time is shown in Figure 8.29.



(a) Bayesian (b) Dempster-Shafer

Figure 8.29: Test 5. Entropy.

By analyzing the entropy we observe smaller values compared to the tests in open space, as expected: more measurements are integrated in the map. In addition, the sensors seem to be more stable, returning the same measurements over time (for instance, the entropy for the right radar barely changes). The main difference with respect to open spaces is that the transience time is quite larger, around 2.5 s. This might be too much for the car to react quickly. However, given the small maximum speed in cities it might be sufficient.

Computation time

Last, the computation time per iteration is presented in Table 8.10.

Implementation	$t_{\min}$ (ms)	$t_{\max}$ (ms)	$t_{avg} \pm \sigma_t$ (ms)
Bayesian	102.21	825.63	398.36 $\pm$ 151.75
Dempster-Shafer	400.60	2841.01	911.50 $\pm$ 329.39

Table 8.10: Test 5. Computation time per iteration.

The computation time has similar levels compared to the previous experiments. The Bayesian implementation run under 400 ms on average, whereas it took a bit more than 900 ms for the Dempster-Shafer case. We observe that this tendency is maintained over all experiments. The variance is increase with respect to the tests in the open area, given the larger number of detections.

8.7 Test 6 (Scenario 4. Configuration 2)

Description

We repeated the previous experiment with the vehicle placed further away from the obstacles. Therefore we evaluate the system's ability to detect obstacles in a city-like environment at large distances.

- **Sensors**: corner radars, front lidar, Velodyne lidar. Therefore, a ground truth will be generated for this experiment.
- **Objects**: see Figure E.4. Two corner reflectors (objects 1 and 8 in the figure), of RCS = 20 dBsm, were placed at the edges of the road, close to the buildings. A smaller corner reflector, of RCS = 10 dBsm, was placed in front of the vehicle, slightly to the right (looking from the vehicle). Finally, a variety of objects were distributed in front of the vehicle: a bicycle (2), a wooden box with planks on top of it (3), two small tires (4,7) and a big tire (5).
- **Environment**: a road intersection in a city-like environment. The obstacles were placed at approximately 40 m from the vehicle. Weather conditions: light rain.

Sensor grids

The individual sensor occupancy grids are presented in Figures 8.30 and 8.31 for the Bayesian and Dempster-Shafer implementation, respectively.

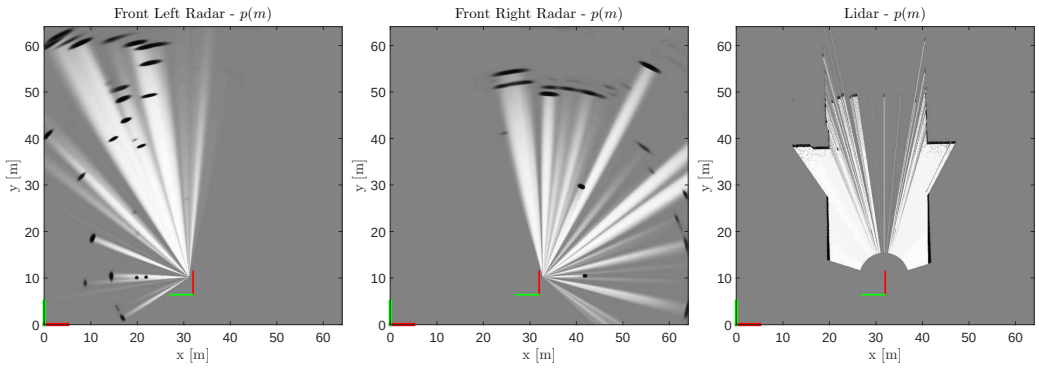


Figure 8.30: Test 6. Sensor occupancy grids. Bayesian implementation.

The more cluttered environment and the larger distance makes it harder for the sensor to detect the placed obstacles. The left radar returns quite many noisy detections that seem to penetrate through the walls. Nevertheless the bicycle (2) is correctly detected at around $(x, y) = (22, 50)$ m. A sign post is also detected by this radar, at $(x, y) = (20, 10)$ m. There seems to be a false positive a couple meters tot the right of the previous detection. The right radar presents a similar performance, but it gives more detections in the region where the obstacles are. The clearest detections correspond to the sign posts at $(x, y) = (40, 10)$ and $(40, 30)$ m. If we further analyze the right radar grid, we notice that it successfully detects the big tire (5) and the reflector behind it (6), as well as the small tire (7) and the

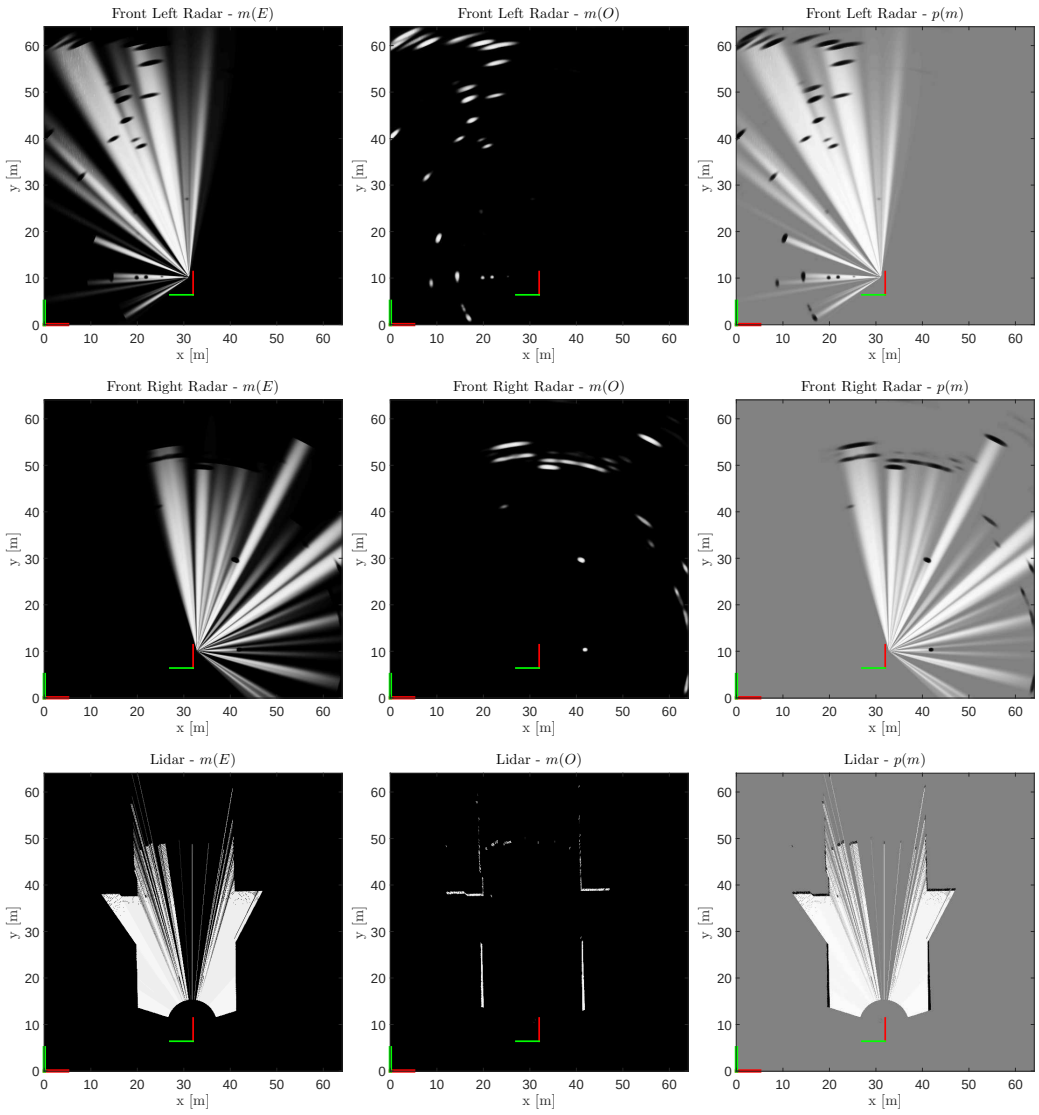


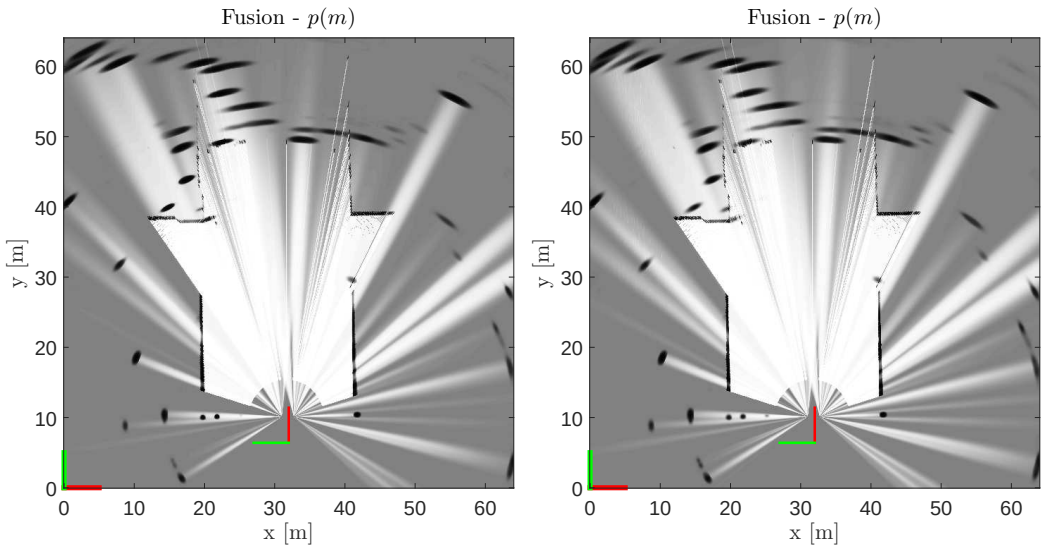
Figure 8.31: Test 6. Sensor occupancy grids. Dempster-Shafer implementation.

right corner reflector (8). The dimensions of the objects are of course not correct due to the angular uncertainty in the radar measurements. There are a couple more measurements at around $(x, y) = (28, 50)$ m, but they do not match with the detections on the other grids, probably due to calibration errors.

The lidar, on the other hand, detects the walls very nicely, but it missed most of the objects, especially the short ones. We clearly observe detections for the left corner reflector (1) at around $(x, y) = (20, 50)$ m, followed by the bicycle (2) and the wooden box (3). Then, the big tire (5) and the right corner reflector (8) are detected but only with one laser beam. Last, the small tires are sometimes detected with one laser beam each. Since the detections are not constant over time, the forgetting factor reduces the likelihood.

Fused grid

The results of performing sensor fusion are shown in Figure 8.32.

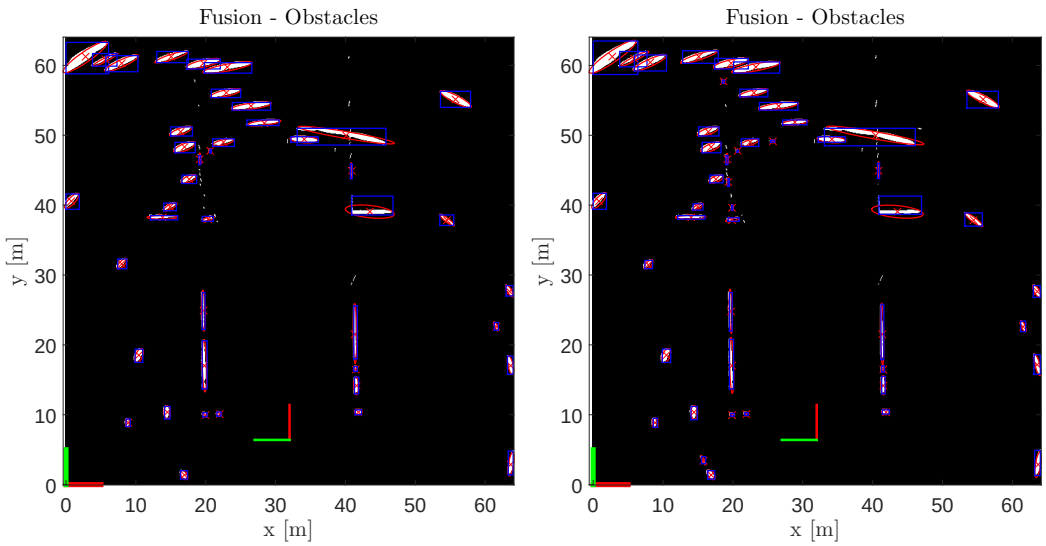


(a) Bayesian (b) Dempster-Shafer

Figure 8.32: Test 6. Fused grid.

Detected objects

Finally, the decision grid and the detected obstacles are shown in Figure 8.33.



(a) Bayesian (b) Dempster-Shafer

Figure 8.33: Test 6. Detected objects.

In addition, the false positive and false negative rates are presented in Table 8.11.

Framework	False positives	False negatives	Real number of objects
Bayesian	2	2	8
Dempster-Shafer	2	1	8

Table 8.11: Test 6. False positives and false negatives.

In this experiment we have many object detections which are hard to associate to the real objects. We first notice in the decision grids that the left corner reflector (1) and the bicycle (2) are successfully detected. In addition, the Dempster-Shafer implementation handles better a conflicting situation and the wooden box is also detected, so there is one less false negative. Therefore, we can say that the Dempster-Shafer implementation performs better in this experiment. Next, there is two detections between $x = 30$ and 40 m, at $y = 50$ m, which correspond to objects 5, 6, 7 and 8. We do not consider here that there are two false negatives. From a practical point of view, our system detected a large obstacle that must be avoided covering the region where all the obstacles are. Furthermore, we do consider as false positives the detections at $(x, y) = (22, 10)$ m (next to the sign post) and the one at $(x, y) = 30, 52$ m, which is too far to correspond to the small tire (4), which is not detected.

Finally, we also analyze the Velodyne grid (see Figure 8.34). It can clearly detect only the bicycle (2) and the wooden box (3). The big tire (5) is barely detected, but there is not a single detection for the rest of the objects. In this case, we can say that our multi-sensor approach clearly outperforms the single-sensor technique.

Ground truth

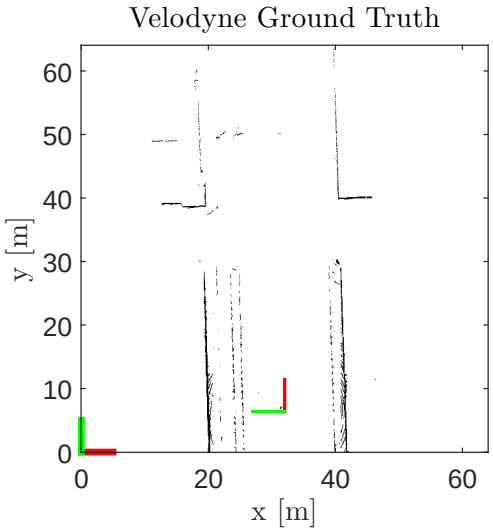
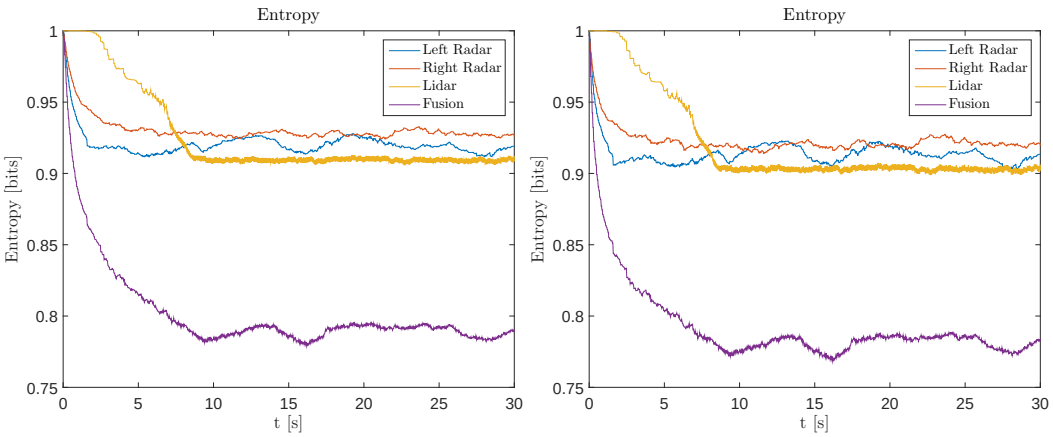


Figure 8.34: Test 6. Ground truth.

Entropy

The average entropy of the grids over time is shown in Figure 8.35.



(a) Bayesian (b) Dempster-Shafer

Figure 8.35: Test 6. Entropy.

The entropy values in this experiment are slightly smaller than in the previous test, since now the vehicle is further away, with a larger field of view, and therefore more measurements are obtained. It is interesting to observe the temporal evolution: we can clearly see that the lidar is quite slow compared to the radars, taking around 7 seconds to reach the maximum information content. The radars are pretty much unaffected and are as fast as in the rest of the experiments. Given this large transient time for the lidar, it would be better to rely mostly on the radars for a fast response at large distances.

Computation time

Last, the computation time per iteration is presented in Table 8.12.

Implementation	$t_{\min}$ (ms)	$t_{\max}$ (ms)	$t_{avg} \pm \sigma_t$ (ms)
Bayesian	100.25	1139.40	488.30 $\pm$ 188.20
Dempster-Shafer	400.44	2025.72	984.41 $\pm$ 302.26

Table 8.12: Test 6. Computation time per iteration.

The computation time is pretty similar to the previous experiment, although the maximum time for the Dempster-Shafer implementation was only around two seconds. Most importantly, the average time remains under 500 ms for the Bayesian approach and 1000 ms in the Dempster-Shafer case. In this case we therefore notice there is a trade-off between detection rate (better with Dempster-Shafer) and computation time (Bayesian is twice faster). The variance is also large, as in the previous experiment, compared to experiments in open space.

Chapter 9

Discussion

The results from the different experiments can be jointly analyzed focusing on several key aspects. We will first evaluate the main differences between the Bayesian and Dempster-Shafer approaches to grid building and sensor fusion. We will continue analyzing the performance of the system in terms of the detection properties of the individual sensors and types of objects, as well as weather conditions and distance to the obstacles.

9.1 Bayesian and Dempster-Shafer

One of the main goals of this project was to compare the Bayesian and Dempster-Shafer approaches towards multi-sensor fusion. The latter seemed interesting because it can explicitly handle the uncertainty and the conflicting information. We evaluate the accuracy of the grids as well as the detection rate for the different scenarios. We pay special attention to the cases of **conflicting** information.

In general we have observed very **little differences** in the results from the Bayesian and Dempster-Shafer when building the individual sensor grids, especially since sensors rarely generate conflicting information when integrating measurements over time. However, the objects detected in the Dempster-Shafer fused grid had a slightly **larger likelihood**, as could be seen from the entropy figures. In fact, we have observed three cases where the **Dempster-Shafer fusion outperformed the Bayesian implementation** in terms of detection rate. In Test 1, the wall was more accurately detected despite the incorrect radar measurement. Moreover, in Tests 5 and 6 there is one less false negative using the Dempster-Shafer approach. The differences are however really subtle and the detection rate might easily change also due to the probability threshold when applying the decision rule or the way objects are filtered by size. In addition, the advanced Dempster-Shafer concepts discussed in Appendix A show that the detection rate can be further improved within this framework, especially when performing multi-sensor fusion.

When it comes to the entropy, the values and evolution are pretty similar, but the Dempster-Shafer case has a lower entropy in all the experiments, meaning that it is more informative, with less uncertainty. The difference is small, of 0.05-0.1 bits at most, but it made a difference when detecting objects, obtaining a couple fewer false negatives than the Bayesian implementation.

The real difference, which greatly impacts the practical application, is the computation

time. We observe that the **Dempster-Shafer implementation is usually around twice slower** than the Bayesian one. The main reason for this is the expensive Dempster-Shafer combination formula, full of multiplications, whereas the efficient log-odds formulation of the Bayes filter is much cheaper.

In conclusion, we could say that the Dempster-Shafer theory provides similar accuracy and slightly better detection rate than the Bayesian approach, mostly due to the way the conflict is handled. The results are still very similar, which is not surprising since this particular problem is easily manageable by both theories —only the states `occupied` and `empty` are considered—. Nonetheless, the Dempster-Shafer theory can be further leveraged to more clearly outperform the Bayesian theory. To show this, a small proof of concept about advanced conflict management is presented in Appendix A, which provides extra functionality and allows to solve cases which the Bayesian theory cannot.

9.2 Sensor Properties and Limitations

Each sensor has its own strengths and weaknesses, which can be observed with the obtained results:

- **Radar**. The radar sensor is quite limited for the purpose of obstacle detection. It can mainly detect metallic objects, such as the corner reflectors, guardrails, bicycles etc. However, the rest of the objects that were placed in front of the vehicle, such as concrete walls, wooden boxes, textile suitcases and so on were not detected by this sensor. In addition, it is clear that the radar is very noisy in city-like environments, where the walls cause specular reflections and the received echoes do not actually correspond to real objects. This is clearly observed in Scenario 4, where the radar reduced the quality of the final fused grid. It would therefore not be advisable to use the corner radars in these environments. It is however very powerful for detecting objects at large distances, where the lidar can barely see the objects, as shown in Tests 4 and 6. The all-weather capability is also a remarkable advantage.

- **Lidar**. The front lidar performs very well since it detects most of the objects. It is also very accurate both in angular resolution and range. Its main limitations are objects with mirror-like surfaces, which produce specular reflections of the laser beams and therefore they never get back to the detector. This is clearly observed when trying to detect the corner reflectors. Depending on their orientation, it is possible that sometimes they are detected, by only by a couple of laser beams. In addition, the lidar struggles to detect short objects, such as the small tires in Scenario 4. This is related to the multi-layer hardware configuration of the lidar and its placement on the vehicle. The two bottom layers are pointed to the ground, so one can expect that at some distance the beams will end up hitting the ground and not travelling any further. The upper layers, on the other hand, point upwards, so it is also possible that they go over an obstacle without detecting it. Another reason for this might be the fact that dark objects absorb most of the energy of the laser beam, so not much energy is returned back and therefore it is not detected. The performance of the lidar is worse at large distances, since only a couple of laser beams reach the object. Finally, adverse weather conditions can also difficult the detection.

- **Velodyne**. Even though the Velodyne lidar was not used in the fusion process, it is interesting to evaluate its performance as well to determine what are realistic objectives for the whole obstacle detection module. Due to its mounting position on top of the car and the large number of layers, the problems of the front lidar were not reproduced here: most of the objects were detected at reasonable distances. It had some difficulties with the corner reflectors, but some measurements were still received. However, we were surprised to realize that not even this sensor is capable of properly detecting objects at large distances, such as 40 m. The received signal contained just a few points per object (1-10 points), which would most likely just be considered as random noise within a point cloud of millions of points. Therefore, difficulties are to be expected when trying to detect small objects at distances larger than 40 m. In this case, our multi-sensor approach outperforms the Velodyne sensor in terms of detection rate.

9.3 Fusion Performance

We now discuss the performance of the low-level multi-sensor fusion process. In general, the resulting grid provides more information given that the fields of view of the sensors are complementary and the working principles are different. This is very clear in Scenario 1, where each sensor (two radars and lidar) detects only one object (corner reflectors and wall). The fused grid nicely integrates the information from the sensors and the objects can be extracted. The same occurs in Scenario 3, test 1.

The fusion performance is degraded mostly when conflicts between the radar and the lidar are produced, especially when the obstacles are relatively far from the vehicle (more than 20 m). The clearest example is Scenario 2, where the radar returns some detections behind the real obstacles, which are accurately detected by the lidar. The result is that the empty cone of the radar completely overrides the detections of the lidar for a couple of objects, which are missed after applying the decision rule. Other examples of this situation are the city-environment tests (Scenario 4), where the radar detections create openings in the walls, as discussed earlier.

9.4 Detection Rate and Accuracy

The performance can also be evaluated in terms of the detection rate and accuracy of the detections. There are usually quite many false detections due to the radar measurements, especially in city-like environments. However, since they are not a hazard for the vehicle they are not counted towards the analysis. In general, most of the objects are properly detected when the vehicle is not too far from them (around 20 m). Between one and four objects are missed in general, mostly the smallest and/or lowest ones, which are less dangerous. When the vehicle is located further away from the obstacles, at around 40 m, the false negative rate greatly increases and objects are rarely detected. Normally the radar returns some detections, although with a large angular uncertainty, not being able to accurately estimate the size of the object. The result is however an overestimation of the size, which is useful for safety purposes. This would not be possible to achieve with the lidar since the beams are very narrow and obstacles are hardly detected.

The accuracy can also be measured in terms of how well the size of the obstacles is determined. As expected, objects detected by the lidar have an excellent definition due to the small angular resolution. It is possible to measure the size of the objects on the grid and verify that it is a fair enough approximation, in the order of 10 cm error (equal to the resolution of the grid). The opposite happens with the radar, especially if the object is far away, as mentioned before. Nevertheless, for the particular application the detection rate is more important than accurately defining the shape and size of the objects.

9.5 Types of Objects

In the previous experiments a diverse set of objects have been considered to evaluate what kind of obstacles are more likely to be detected. We can analyze the results according to the following characteristics:

- **Size**. Not surprisingly, large objects are easier to detect than small objects. This is usually not a problem for the lidar sensor, given the high angular resolution, although if only a few measurements are obtained from an object they will most likely be filtered out afterwards to remove noise and reduce the number of false positives. The main difficulty, as discussed before, is the **height** of the object: if it is too short it might not be detected by any of the layers. The radars' performance greatly depends on the radar cross-section of the object, which is not only related to the size of the object, but also its material and shape. For instance, they successfully detect the 20 dBsm side corner reflectors, which are relatively small (around 30 cm side). Since the radar waveform has a wider elevation field of view (7º versus 3.2º of the lidar), the height of the objects is usually not a problem for this sensor.
- **Material**. Different sensors are sensitive to different materials. However, the chosen sensors (radar, lidar) are based on different physical principles and therefore complement each other very well. As discussed, the radar is quite limited to metallic objects only, whereas the lidar cannot handle specular surfaces and transparent materials.

9.6 Environment

Furthermore, the obstacle detection module was tested in different environments, both in terms of the scenario and the weather conditions:

- **Scenario**. Two main scenarios where tested: open-space and city-like environments, with buildings, traffic signs and so on. As expected, open scenarios are easier to handle since the sensors are less likely to return noisy measurements. This is particularly important in the case of the radars, since the supplier only provides a set of 64 raw measurements. The information about how these measurements are selected from the whole set of measurements is unfortunately not provided, but we have observed that closer objects and with larger RCS are prioritized. This would likely be an issue in roads with guardrails, since they are metallic (large RCS) and very close to the vehicle, so most of the measurements would belong to them.

Measurements corresponding to objects in front of the car would therefore not be returned, even though they might be physically detected. In this case, it would be better to have a forward-looking radar rather than two corner radars. The city-like environment was especially tricky for the radar as well, due to the specular reflections of the radar waveforms on the buildings, as mentioned before.

- **Weather conditions**. The testing under different weather conditions is usually difficult since it would require a controlled chamber. Therefore we only could test the two main weather conditions that can be experimented in spring: sunny and rainy conditions. The best performance is of course obtained in sunny conditions, for all the sensors. The main difficulties could come under rain and especially snow conditions. The radar is pretty much unaffected by the weather conditions due to the large wavelength (77 GHz), so the waveform can penetrate the rain drops without any major problems. This is usually not the case for the lidar (905 nm), where the laser beam can suffer from back-scattering when penetrating a rain drop or snowflake. In Scenario 3 the weather conditions were moderate to intense rain. The main observable effect on the lidar is that the first echoes were not received. However, due to the **multi-echo technology**, the second and third echoes were successfully received and therefore the objects were still detected without any major issue. Since there was no great noise in the received data, the sensor performance estimation module did not output a low weight for this sensor, since indeed it was performing pretty well.

9.7 Detection Distance

Finally, we evaluate the performance of the system with respect to the required distance to be able to detect obstacles. For that, some experiments where made with the vehicle located between 10 and 40 m from the obstacles.

From the previous experiments we observe that the best results are obtained for small distances (20 m or less), both in terms of the number of detected obstacles and also the accuracy in estimating their dimensions. In some scenarios it is still possible to detect the lagger obstacles at 40 m, but the small ones remain undetected by the radar and the lidar. Finally, at 40 m the lidar (either the front lidar or the Velodyne) cannot really detect any obstacle in front of the vehicle.

Another observation to highlight is that the radar is capable of detecting obstacles at such large distances. However, the sensor model assumes a detection cone of $\pm 1^{\circ}$ width, which produces as a result extremely large occupied regions for each detections. At 40 m, the radar grid clearly worsens the performance in terms of distinguishing obstacles: instead, one single object is detected due to the overlapping of the large occupied area with the lidar detections. This is quite bad for the map accuracy. On the other hand, it is somewhat beneficial from a practical point of view: small metallic objects in front of the car will be detected and therefore a collision avoidance action will be triggered. In addition, we are overestimating the size of the object in this case so this approach will generate safer trajectories, further away from the object than what would be required.

Chapter 10

Conclusions and Future Work

We finish this report with a summary of the work that has been done, the obtained results and some final conclusions. Finally, some future lines of research are suggested.

The goal of the project was to design a module for an autonomous vehicle, capable of detecting small static obstacles on the road. The approach to this problem was to perform sensor fusion at a low level (i.e. using raw sensor data whenever possible) using the occupancy grid framework. In addition, a comparison of the Bayesian and Dempster-Shafer theories to update the grids and perform fusion was to be made, in terms of performance and computational complexity.

The complete system has been efficiently implemented in MATLAB. The running time greatly depends on the environment; in the performed tests it is roughly 500 ms per iteration in the Bayesian case, and 1000 ms in the Dempster-Shafer framework. It was not possible to create a real-time solution as was originally desired, due to the large number of computations required. As can be seen in the literature, GPU-based implementations are essential in order to be able to run in real-time.

The implementation was tested under a wide variety of scenarios, including objects of different sizes, materials and placement with respect to the vehicle. The weather conditions and the environment where the car was located also varied throughout the experiments. The results show that the objects were correctly detected when the vehicle was sufficiently close to them (below 40 m). After this distance, the performance degraded significantly. In addition, open areas were easier to handle, especially since the radar sensor returned quite many noisy measurements due to the multi-path received signals in city-like environments. At large distances (40 m and beyond) it was hard for the system to accurately detect objects. Not even the most accurate sensor —the Velodyne lidar— was able to detect them, so additional sensors or different techniques would be required in this case.

When comparing the Bayesian and Dempster-Shafer approaches, we observed very little differences among them, mostly due to the equivalent implementation and the simplicity of the problem at hand —classify cells as occupied or empty—. The computation time is clearly an advantage of the Bayesian implementation. Nevertheless, the Dempster-Shafer measure of conflict can be further exploited to provide additional functionality, such a sensor performance estimator. In addition, highly conflicting situations can be solved to reduce the false negative rate, unlike in the Bayesian approach. In these cases,

the Dempster-Shafer approach proved to be superior.

We have also demonstrated how suitable the occupancy grid framework is for fusing heterogeneous information sources. The raw sensor data from radar and lidar was transformed into a common framework through inverse sensor models. The problem of data association was skipped since the fusion was automatically performed on the grid. In addition, the system is easily scalable to include more sensors by just adding and updating an additional occupancy grid and fusing it with the others.

Future work

The scope of this project was limited due to the time constraints and the availability of resources. Nevertheless, we propose some lines of future work that could extend the functionality of the developed system or improve its performance:

- **Full exploitation of the Dempster-Shafer framework**. After analyzing the performance of both the Bayesian and Dempster-Shafer frameworks, we reach the conclusion that the latter provides little improvement in relation to the much higher computational cost. This is related to the simple frame of discernment that we have defined for this project, where each cell can only be either empty or occupied, and the similar implementation in both cases. However, one could think of more complex classifications to improve the situational awareness. For instance, Kurdej *et al.* [103] recently presented a work where they define multiple cell status: drivable space, free non-drivable space, buildings, moving objects and so on. This requires richer sources of information apart from exteroceptive sensors, such as a detailed map of the environment. The Dempster-Shafer framework is suitable for fusing any kind of information as long as it can formulated properly in terms of the frame of discernment and belief masses. For instance, information about other driver's intentions could be included. This information could be easily obtained through inter-vehicle communication.

- **Hybrid sensor fusion**. From the results, we observe that when performing *temporal fusion* (updating the individual sensor grids), the results from the Bayesian and Dempster-Shafer implementations are pretty similar. However, when performing *multi-sensor* fusion, Dempster-Shafer improves the detection rate by better handling the conflicts. Since we chose *decentralized fusion* in this project, the following implementation could be done to get the best from both theories: perform temporal fusion in the Bayesian domain and transform to the Dempster-Shafer domain afterwards for grid fusion.

- **Dynamic obstacles**. In this project we limited ourselves to the detection of static obstacles due to the limitations of the classical occupancy grid mapping. It would be desirable to extend the work detecting and tracking dynamic obstacles as well. For this, the Bayesian Occupancy Filter might be a good approach, although rather computationally expensive. Other authors, like Moras *et al.* [60], explored the possibility of detecting moving objects in the Dempster-Shafer framework through the analysis of conflict.

- **Cooperative mapping**. Sometimes the vehicle cannot see potential obstacles that other vehicles can, for instance due to occlusions. It would then be an interesting idea to combine all the knowledge from the vehicles in the surroundings of the ego-vehicle. Li *et al.* [104] propose a cooperative grid-based sensor fusion based on this principle, proving its feasibility and usefulness.

- **Parallel computing implementation**. The main drawback of the current implementation is the computational time. The main bottlenecks are found when computing the inverse sensor models and performing sensor fusion. This is reasonable since the grids used in this project are quite dense: over 400.000 elements for a grid of 64×64 m with a resolution of 0.1 m/cell. Many similar papers found in the literature take advantage of the GPU massive computation capabilities to provide efficient, real-time implementations of occupancy grid mapping [105][106]. GPUs' architectures are specifically designed for parallelization, being capable of executing thousands of threads in parallel. For this particular problem, given the *cell independence* assumption, each cell on the grid can be updated independently, and therefore the problem is perfectly suited for GPUs.

  A small research about the possibility of such implementation was performed. The main option is to use the CUDA programming model, in C [107]. It is however NVIDIA propietary, and therefore not portable to systems where another GPU is used. Instead, a more general framework would be desirable. An interesting approach is OpenCL [108], which allows to distribute the workload into any computational unit: CPU, GPU, DSP and so on. It is less popular in the literature but it is worth exploring in order to design a general solution. Recently, Rakotovao *et al.* [109] show how it is possible to efficiently implement a Bayesian Occupancy Filter (BOF) using OpenCL on a multi-core system. The comparison to the CUDA implementation shows that it is a bit slower but still capable of real-time operation.

  This proposal was not implemented mainly due to time constraints. A complete port from MATLAB to C code would imply a significant time draw, especially when it comes to debugging. In addition, the use of the CUDA/OpenCL is not straightforward; instead, advanced parallel programming skills are required to better structure the algorithm towards a parallel implementation. It is also important to have a careful memory management in order not to spend much time in data transfer between CPU and GPU.

References

[1] W. Jones. *Self-driving Cars: Saving Lives AND Energy*. Ed. by I. S. C. T. Think. [Online; posted 18-November-2014]. 2014. URL: http://spectrum.ieee.org/cars-that-think/transportation/self-driving/selfdriving-cars-saving-lives-and-energy.

[2] T. Litman. *Autonomous Vehicle Implementation Predictions - Implications for Transport Planning*. Ed. by V. T. P. Institute. 2015.

[3] S. Thrun et al. "Stanley: The Robot That Won the DARPA Grand Challenge: Research Articles". In: *J. Robot. Syst.* 23.9 (Sept. 2006), pp. 661–692.

[4] E. Ackerman. *Google's Autonomous Cars Are Smarter Than Ever at 700,000 Miles*. Ed. by I. S. C. T. Think. [Online; posted 29-April-2014]. 2014. URL: http://spectrum.ieee.org/cars-that-think/transportation/self-driving/google-autonomous-cars-are-smarter-than-ever.

[5] A. Discant et al. "Sensors for Obstacle Detection - A Survey". In: *Electronics Technology, 30th International Spring Seminar on*. 2007, pp. 100–105.

[6] B. ElHalawany et al. "Vision-based obstacles detection for a mobile robot". In: *Informatics and Systems (INFOS), 2012 8th International Conference on*. 2012, pp. MM–93–MM–99.

[7] J. Li and M. Chen. "On-Road Multiple Obstacles Detection in Dynamical Background". In: *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2014 Sixth International Conference on*. Vol. 1. 2014, pp. 102–105.

[8] U. Meis, W. Ritter, and H. Neumann. "Detection and classification of obstacles in night vision traffic scenes based on infrared imagery". In: *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*. Vol. 2. 2003, 1140–1144 vol.2.

[9] N. Bernini et al. "Real-time obstacle detection using stereo vision for autonomous ground vehicles: A survey". In: *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. 2014, pp. 873–878.

[10] Z. Khalid, E.-A. Mohamed, and M. Abdenbi. "Stereo vision-based road obstacles detection". In: *Intelligent Systems: Theories and Applications (SITA), 2013 8th International Conference on*. 2013, pp. 1–6.

[11] Z. Zhang et al. "Real-time obstacle detection based on stereo vision for automotive applications". In: *Education and Research Conference (EDERC), 2012 5th European DSP*. 2012, pp. 281–285.

[12] K. Kaliyaperumal, S. Lakshmanan, and K. Kluge. "An algorithm for detecting roads and obstacles in radar images". In: *Vehicular Technology, IEEE Transactions on* 50.1 (2001), pp. 170–182.

[13] G. Brooker, M. Bishop, and S. Scheding. "Millimetre waves for robotics". In: *Australian Conference for Robotics and Automation* (2001).

[14] J. Ryde and N. Hillier. "Performance of laser and radar ranging devices in adverse environmental conditions". In: *Journal of Field Robotics* 26.9 (2009), pp. 712–727.

[15] U. Nickel. "Applications of superresolution for radar: Examples, problems and solutions". In: *Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European*. 2013, pp. 1–5.

[16] J. Han et al. "Enhanced Road Boundary and Obstacle Detection Using a Downward-Looking LIDAR Sensor". In: *Vehicular Technology, IEEE Transactions on* 61.3 (2012), pp. 971–985.

[17] O. Yalcin et al. "Detection of road boundaries and obstacles using LIDAR". In: *Computer Science and Electronic Engineering Conference (CEEC), 2014 6th*. 2014, pp. 6–10.

[18] R. H. Rasshofer, M. Spies, and H. Spies. "Influences of weather phenomena on automotive laser radar systems". In: *Advances in Radio Science* 9 (2011), pp. 49–60.

[19] C. Huihai, L. Shuqiang, and Z. Yingsheng. "An obstacle detection algorithm used sequential sonar data for Autonomous Land Vehicle". In: *Electronic Measurement Instruments (ICEMI), 2011 10th International Conference on*. Vol. 4. 2011, pp. 255–259.

[20] B. Khaleghi et al. "Multisensor Data Fusion: A Review of the State-of-the-art". In: *Inf. Fusion* 14.1 (Jan. 2013), pp. 28–44.

[21] R. Luo, C. C. Chang, and C. C. Lai. "Multisensor Fusion and Integration: Theories, Applications, and its Perspectives". In: *Sensors Journal, IEEE* 11.12 (2011), pp. 3122–3138.

[22] M. Perrollaz et al. "Long Range Obstacle Detection Using Laser Scanner and Stereovision". In: *Intelligent Vehicles Symposium, 2006 IEEE*. 2006, pp. 182–187.

[23] N. Floudas et al. "High Level Sensor Data Fusion Approaches For Object Recognition In Road Environment". In: *Intelligent Vehicles Symposium, 2007 IEEE*. 2007, pp. 136–141.

[24] X. Wang et al. "Bionic vision inspired on-road obstacle detection and tracking using radar and visual information". In: *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. 2014, pp. 39–44.

[25] T. Kubertschak, M. Maehlisch, and H.-J. Wuensche. "Towards a unified architecture for mapping static environments". In: *Information Fusion (FUSION), 2014 17th International Conference on*. 2014, pp. 1–8.

[26] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe. "Sonar tracking of multiple targets using joint probabilistic data association". In: *Oceanic Engineering, IEEE Journal of* 8.3 (1983), pp. 173–184.

[27] D. B. Reid. “An algorithm for tracking multiple targets”. In: *Automatic Control, IEEE Transactions on* 24.6 (1979), pp. 843–854.

[28] G. W. Kim and B. H. Lee. “Hierarchical Sensor Fusion for Building an Occupancy Grid Map using Active Sensor Modules”. In: *SICE-ICASE, 2006. International Joint Conference*. 2006, pp. 2600–2605.

[29] P. Lindner and G. Wanielik. “Multi level fusion for an automotive pre-crash safety system”. In: *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*. 2008, pp. 143–146.

[30] D. Nuss, M. Stuebler, and K. Dietmayer. “Consistent environmental modeling by use of occupancy grid maps, digital road maps, and multi-object tracking”. In: *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. 2014, pp. 1371–1377.

[31] A. Elfes and L. Matthies. “Sensor integration for robot navigation: combining sonar and stereo range data in a grid-based representation”. In: *Decision and Control, 1987. 26th IEEE Conference on*. Vol. 26. IEEE. 1987, pp. 1802–1807.

[32] H. Moravec and A. E. Elfes. “High Resolution Maps from Wide Angle Sonar”. In: *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*. 1985, pp. 116 –121.

[33] M. Kumar, D. Garg, and R. Zachery. “Multi-sensor fusion strategy to obtain 3-D occupancy profile”. In: *Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE*. 2005, 6 pp.–.

[34] Q. Baig and O. Aycard. “Low level data fusion of laser and monocular color camera using occupancy grid framework”. In: *Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on*. 2010, pp. 905–910.

[35] Q. Baig, M. Perrollaz, and C. Laugier. “A Robust Motion Detection Technique for Dynamic Environment Monitoring: A Framework for Grid-Based Monitoring of the Dynamic Environment”. In: *Robotics Automation Magazine, IEEE* 21.1 (2014), pp. 40–48.

[36] F. Homm, N. Kaempchen, and D. Burschka. “Fusion of laserscannner and video based lanemarking detection for robust lateral vehicle control and lane change maneuvers”. In: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. 2011, pp. 969–974.

[37] M. Kang et al. “Map Building Based on Sensor Fusion for Autonomous Vehicle”. In: *Information Technology: New Generations (ITNG), 2014 11th International Conference on*. 2014, pp. 490–495.

[38] D. Nuss et al. “Fusion of laser and monocular camera data in object grid maps for vehicle environment perception”. In: *Information Fusion (FUSION), 2014 17th International Conference on*. 2014, pp. 1–8.

[39] I. Paromtchik, M. Perrollaz, and C. Laugier. “Fusion of telemetric and visual data from road scenes with a lexus experimental platform”. In: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. 2011, pp. 746–751.

[40] J. Adarve et al. “Computing occupancy grids from multiple sensors using linear opinion pools”. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. 2012, pp. 4074–4079.

[41] R. Garcia et al. “High level sensor data fusion for automotive applications using occupancy grids”. In: *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on.* 2008, pp. 530–535.

[42] P. Stepan, M. Kulich, and L. Preucil. “Robust data fusion with occupancy grid”. In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 35.1 (2005), pp. 106–115.

[43] R. Grover, G. Brooker, and H. F. Durrant-Whyte. “A low level fusion of millimeter wave radar and night-vision imaging for enhanced characterization of a cluttered environment”. In: *Proceedings of Australian Conference on Robotics and Automation* (2001), pp. 98–103.

[44] E. Ivanjko, I. Petrović, and M. Brezak. “Experimental comparison of sonar based occupancy grid mapping methods”. In: *AUTOMATIKA: Journal for Control, Measurement, Electronics, Computing and Communications (0005-1144)* 1-2 (2009), pp. 65–79.

[45] H. Wu et al. “Sensor fusion using Dempster-Shafer theory [for context-aware HCI]”. In: *Instrumentation and Measurement Technology Conference, 2002. IMTC/2002. Proceedings of the 19th IEEE.* Vol. 1. IEEE. 2002, pp. 7–12.

[46] H. Wu, M. Siegel, and S. Ablay. “Sensor fusion using Dempster-Shafer theory II: static weighting and Kalman filter-like dynamic weighting”. In: *Instrumentation and Measurement Technology Conference, 2003. IMTC ’03. Proceedings of the 20th IEEE.* Vol. 2. 2003, 907–912 vol.2.

[47] Z. Xiang and U. Ozguner. “Environmental perception and multi-sensor data fusion for off-road autonomous vehicles”. In: *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE.* 2005, pp. 420–425.

[48] S. Soleimanpour, S. Ghidary, and K. Meshgi. “Sensor fusion in Robot localization using DS-Evidence Theory with conflict detection using Mahalanobis distance”. In: *Cybernetic Intelligent Systems, 2008. CIS 2008. 7th IEEE International Conference on.* 2008, pp. 1–6.

[49] M. Delafosse et al. “A dempster-shafer fusion architecture for the incremental mapping paradigm”. In: *Information Fusion, 2007 10th International Conference on.* 2007, pp. 1–7.

[50] H. Cao et al. “Laser-scanner grid map building based on Dempster-Shafer evidence theory”. In: *Mechatronics and Automation, 2009. ICMA 2009. International Conference on.* 2009, pp. 4930–4935.

[51] A. C. Plascencia and J. D. Bendtsen. “Sensor Data Fusion Architecture for a Mobile Robot Application Using SIFT and Sonar Measurements”. In: ().

[52] G. Oriolo, G. Ulivi, and M. Vendittelli. “Fuzzy Maps: A New Tool for Mobile Robot Perception and Planning”. In: *Journal of Robotic Systems* 14 (1997), pp. 179–197.

[53] J. Borenstein and Y. Koren. “Histogramic in-motion mapping for mobile robot obstacle avoidance”. In: *Robotics and Automation, IEEE Transactions on* 7.4 (1991), pp. 535–539.

[54] K. Konolige. "Improved Occupancy Grids for Map Building". In: *Autonomous Robots* 4 (1997), pp. 351–367.

[55] C. Coué et al. "Bayesian occupancy filtering for multitarget tracking: an automotive application". In: *The International Journal of Robotics Research* 25.1 (2006), pp. 19–30.

[56] C. Chen et al. "Dynamic Environment Modeling with Gridmap: A Multiple-Object Tracking Application". In: *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on.* 2006, pp. 1–6.

[57] A. Negre, L. Rummelhard, and C. Laugier. "Hybrid sampling Bayesian Occupancy Filter". In: *Intelligent Vehicles Symposium Proceedings, 2014 IEEE.* 2014, pp. 1307–1312.

[58] D. Meyer-Delius, M. Beinhofer, and W. Burgard. "Occupancy Grid Models for Robot Mapping in Changing Environments". In: *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI).* Toronto, Canada, 2012.

[59] R. Danescu. "Obstacle Detection Using Dynamic Particle-Based Occupancy Grids". In: *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on.* 2011, pp. 585–590.

[60] J. Moras, V. Cherfaoui, and P. Bonnifait. "Evidential grids information management in dynamic environments". In: *Information Fusion (FUSION), 2014 17th International Conference on.* 2014, pp. 1–7.

[61] R. Jungnickel and F. Korf. "Object tracking and dynamic estimation on evidential grids". In: *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on.* 2014, pp. 2310–2316.

[62] Y. Zhou and H. Leung. "Minimum entropy approach for multisensor data fusion". In: *Higher-Order Statistics, 1997., Proceedings of the IEEE Signal Processing Workshop on.* 1997, pp. 336–339.

[63] M. Kumar, D. Garg, and R. Zachery. "A Method for Judicious Fusion of Inconsistent Multiple Sensor Data". In: *Sensors Journal, IEEE* 7.5 (2007), pp. 723–733.

[64] J. Carlson and R. R. Murphy. "Use of Dempster-Shafer conflict metric to detect interpretation inconsistency". In: *arXiv preprint arXiv:1207.1374* (2012).

[65] J. D. Gage. "An Inconsistency-based Approach for Sensing Assessment in Unknown Environments". AAI3420536. PhD thesis. Tampa, FL, USA, 2009.

[66] R. C. Gonzalez and R. E. Woods. "Digital image processing, 2nd". In: *SL: Prentice Hall* 2 (2002).

[67] T.-N. Nguyen et al. "Stereo-Camera-Based Urban Environment Perception Using Occupancy Grid and Object Tracking". In: *Intelligent Transportation Systems, IEEE Transactions on* 13.1 (2012), pp. 154–165.

[68] D. Vasquez et al. "Fast Object Extraction from Bayesian Occupancy Grids using Self Organizing Networks". In: *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on.* 2006, pp. 1–6.

[69] K. Mekhnacha et al. "The fast clustering-tracking algorithm in the Bayesian occupancy filter framework". In: *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on.* 2008, pp. 238–245.

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

[71] A. P. Dempster. "A generalization of Bayesian inference". In: *Journal of the Royal Statistical Society* 30.B (1968), pp. 205–247.

[72] G. Shafer et al. *A mathematical theory of evidence.* Vol. 1. Princeton university press Princeton, 1976.

[73] R. R. Murphy. "Dempster-Shafer theory for sensor fusion in autonomous mobile robots". In: *Robotics and Automation, IEEE Transactions on* 14.2 (1998), pp. 197–206.

[74] B. Hua. "Fusion of uncertain information using vague sets and Dempster-Shafer theories". In: *Information Reuse Integration, 2009. IRI '09. IEEE International Conference on.* 2009, pp. 17–22.

[75] B. Grabe, T. Ike, and M. Hoetter. "Evidence based evaluation method for grid-based environmental representation". In: *Information Fusion, 2009. FUSION '09. 12th International Conference on.* 2009, pp. 1234–1240.

[76] P. Smets. "Decision making in the TBM: the necessity of the pignistic transformation". In: *International Journal of Approximate Reasoning* 38.2 (2005), pp. 133–147.

[77] L. Zadeh. *On the Validity of Dempster's Rule of Combination of Evidence.* Memorandum UCB/ERL-M.

[78] R. R. Yager. "On the Dempster-Shafer Framework and New Combination Rules". In: *Inf. Sci.* 41.2 (Mar. 1987), pp. 93–137.

[79] L. Bicheng, H. Jie, and Y. Hujun. "Two Efficient Combination Rules for Conflicting Belief Functions". In: *Artificial Intelligence and Computational Intelligence, 2009. AICI '09. International Conference on.* Vol. 3. 2009, pp. 421–426.

[80] P. Wang. "The Reliable Combination Rule of Evidence in Dempster-Shafer Theory". In: *Image and Signal Processing, 2008. CISP '08. Congress on.* Vol. 2. 2008, pp. 166–170.

[81] G. Xin, Y. Xiao, and H. You. "An improved Dempster-Shafer algorithm for resolving the conflicting evidences". In: *International Journal of Information Technology* 11.12 (2005), pp. 68–75.

[82] F. Campos and S. Cavalcante. "An extended approach for Dempster-Shafer theory". In: *Information Reuse and Integration, 2003. IRI 2003. IEEE International Conference on.* 2003, pp. 338–344.

[83] P. Smets and R. Kennes. "The transferable belief model". In: *Artificial intelligence* 66.2 (1994), pp. 191–234.

[84] R. Haenni. "Shedding new light on Zadeh's criticism of Dempster's rule of combination". In: *Information Fusion, 2005 8th International Conference on.* Vol. 2. 2005, 6 pp.–.

[85] J. Daniel and J.-P. Lauffenburger. “Conflict management in multi-sensor dempster-shafer fusion for speed limit determination”. In: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. 2011, pp. 987–992.

[86] R. Merali and T. Barfoot. “Occupancy grid mapping with Markov Chain Monte Carlo Gibbs sampling”. In: *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. 2013, pp. 3183–3189.

[87] V. Dhiman et al. “Modern MAP inference methods for accurate and fast occupancy grid mapping on higher order factor graphs”. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. 2014, pp. 2037–2044.

[88] G. K. Kraetzschmar et al. “Probabilistic Quadtrees for Variable-Resolution Mapping of Large Environments”. In: *Eds.), Proceedings of the 5th IFAC/EURON*. 2004.

[89] Y. Li and Y. Ruichek. “Building variable resolution occupancy grid map from stereoscopic system: A quadtree based approach”. In: *Intelligent Vehicles Symposium (IV), 2013 IEEE*. 2013, pp. 744–749.

[90] M. A. Fischler and R. C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. In: *Commun. ACM* 24.6 (June 1981), pp. 381–395.

[91] D. Joubert. “Adaptive occupancy grid mapping with measurement and pose uncertainty”. MA thesis. Stellenbosch University, Dec. 2012.

[92] Q. Baig et al. “Using fast classification of static and dynamic environment for improving Bayesian occupancy filter (BOF) and tracking”. In: *Control Automation Robotics Vision (ICARCV), 2012 12th International Conference on*. 2012, pp. 656–661.

[93] C. Yu, V. Cherfaoui, and P. Bonnifait. “An Evidential Sensor Model for Velodyne Scan Grids”. In:

[94] E. Brookner. *Phased-array Radars*. National Emergency Training Center, 1985.

[95] M. I. Skolnik. *Introduction to Radar Systems 2nd Edition*. 2nd ed. New York: McGraw Hill Book Co., 1980.

[96] B. Clarke et al. “Sensor modelling for radar-based occupancy mapping”. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. 2012, pp. 3047–3054.

[97] J. Berger. *Statistical decision theory and Bayesian analysis*. 2. ed. Springer series in statistics. New York, NY [u.a.]: Springer. XVI, 617.

[98] M. Abdulghafour and M. A. Abidi. *Data fusion through nondeterministic approaches: a comparison*. 1993.

[99] K. Pathak et al. “3D forward sensor modeling and application to occupancy grid based sensor fusion”. In: *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. 2007, pp. 2059–2064.

[100] J. W. Guan and D. A. Bell. *Evidence Theory and Its Applications*. New York, NY, USA: Elsevier Science Inc., 1992.

[101] V. Lohweg and U. Monks. "Sensor fusion by two-layer conflict solving". In: *Cognitive Information Processing (CIP), 2010 2nd International Workshop on*. 2010, pp. 370–375.

[102] A.-C. Le et al. "Weighted combination of classifiers for word sense disambiguation based on dempster-shafer theory". In: *Research, Innovation and Vision for the Future, 2006 International Conference on*. 2006, pp. 133–138.

[103] M. Kurdej et al. "Map-Aided Evidential Grids for Driving Scene Understanding". In: *Intelligent Transportation Systems Magazine, IEEE* 7.1 (2015), pp. 30–41.

[104] H. Li and F. Nashashibi. "A new method for occupancy grid maps merging: Application to multi-vehicle cooperative local mapping and moving object detection in outdoor environment". In: *Control Automation Robotics Vision (ICARCV), 2012 12th International Conference on*. 2012, pp. 632–637.

[105] M. Yguel, O. Aycard, and C. Laugier. "Efficient GPU-based Construction of Occupancy Girds Using several Laser Range-finders". In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. 2006, pp. 105–110.

[106] F. Homm et al. "Efficient occupancy grid computation on the GPU with lidar and radar for road boundary detection". In: *Intelligent Vehicles Symposium (IV), 2010 IEEE*. 2010, pp. 1006–1013.

[107] NVIDIA. *CUDA C Programming Guide*. Ed. by NVIDIA. 2015. URL: http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf.

[108] K. O. W. Group. *The OpenCL Specification*. Ed. by A. Munshi. 2012. URL: https://www.khronos.org/registry/cl/specs/opencl-1.2.pdf.

[109] T. A. Rakotovao et al. "Intelligent Vehicle Perception: Toward the Integration on Embedded Many-core". In: *Proceedings of the 6th Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures*. PARMA-DITAM '15. Amsterdam, Netherlands, 2015, pp. 7–12.

[110] J. Moras, V. Cherfaoui, and P. Bonnifait. "Credibilist occupancy grids for vehicle perception in dynamic environments". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. 2011, pp. 84–89.

Appendix A

Advanced Dempster-Shafer

In this project we have implemented the occupancy grid mapping within the Dempster-Shafer framework as equivalent as possible to the Bayesian formulation, in order to do an even comparison. However, it is interesting to try to leverage the additional features of Dempster-Shafer framework to add functionality to the system, which would not possible to accomplish in the classical Bayesian framework.

In particular, the analysis of the **conflict** generated when updating the individual sensor grids and performing fusion can be used for some applications. Here we investigate two of them. First, we propose a sensor performance estimator, which can detect whether the sensor is returning noisy measurements (for instance, under adverse weather conditions). Second, we can solve situations where the sensors are in conflict with each other in order to reduce the false negative rate.

A.1 Sensor Performance Estimation

The sensor fusion process can be refined by estimating how much each sensor should contribute to the final output, according to its performance, which is directly related to the working conditions of the sensor. Several cases are possible:

a) **Normal conditions**. The sensor is working within the specifications set by the manufacturer.

b) **Undetectable objects**. The sensor cannot detect a given object due to its physical working principle. For instance, the radar will mostly detect metallic objects only, whereas the lidar cannot handle transparent objects or specular reflections.

c) **Adverse environmental conditions**. They prevent the sensor from giving correct measurements. For instance, under heavy rain, snow or dust —i.e. atmospheric clutter— the lidar sensor will return noisy readings due to light scattering.

d) **Total sensor failure**. The sensor is simply defective.

Cases a) and b) are covered by the classical sensor fusion algorithm, implemented up to this point. The sensors are considered to be reliable (with some noise) and it is accepted that some objects will not be detected by all the sensors. The fusion should compensate for

each sensor's weaknesses. Case d) is normally handled by the sensors themselves: they are able to determine whether they are faulty or not. Finally, case c) is tackled in this section. The goal is to design a system that, for every sensor, allows to estimate whether the sensor is returning correct measurements given the current environmental conditions.

Assumptions

The following two assumptions are made:

- The world is **consistent**: a cell cannot be occupied and empty at the same time.
- Adverse conditions greatly increase the sensor noise beyond the specifications provided by the manufacturer. The proposed system should be able to handle any kind of **random** noise without the need to derive a complex model of it, as long as noisy, non-constant measurements appear. In other words, if the sensor reports constant or no measurements this module will not be able to detect it.

A.1.1 Algorithm

The key idea of the proposed system is the analysis of the *conflict* metric provided by the Dempster-Shafer combination rule when performing grid update, for each sensor's occupancy grid. A block diagram depicting the algorithm is shown in Figure A.1.

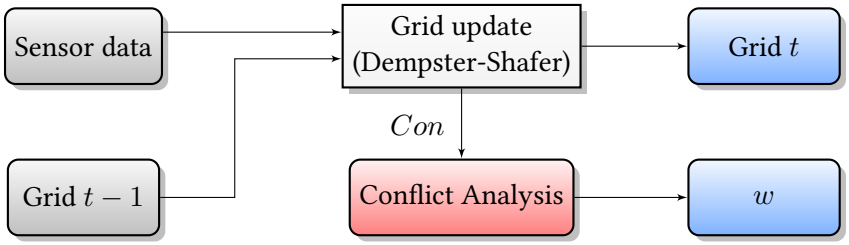


Figure A.1: Block diagram for the Sensor Performance algorithm.

For every sensor, the conflict matrix Con is obtained when performing the temporal fusion of information (see Equation 4.23). The weight of the sensor, w, will be related to the conflict. It is important to note that a measure of conflict is obtained for every sensor, and they are treated **independently**. The motivation is clear: in order to detect whether a sensor is giving wrong measurements, it cannot be compared to other sensors, since the physical principles are different. Instead, we compare the new measurements with its own **history** of measurements, which have been integrated over time in the occupancy grid.

A.1.2 Implementation

The following steps are implemented in the Conflict Analysis module, as illustrated in Figure A.2.

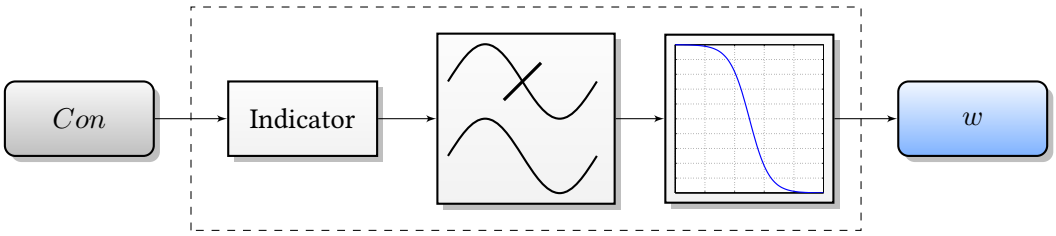


Figure A.2: Detailed description of the Conflict Analysis module.

1. The input is the conflict matrix, Con, containing the conflict value for every cell, as presented in Equation 4.23. We restate it here for convenience:

$$Con = \log\left(\frac{1}{1-K}\right) \quad ; \quad K = (m_1 \textcircled{\cap} m_2)(\emptyset) = \sum_{A,B \in 2^{\Theta} | A \cap B = \emptyset} m_1(A) \cdot m_2(B) \tag{A.1}$$

2. A scalar conflict indicator, Ind, is computed:

$$Ind = \frac{\sum_i \sum_j Con(i,j)}{N_{\text{updated}}} \tag{A.2}$$

where N_{updated} is the number of cells that have been updated due to the latest sensor measurement.

3. Since the sensor is assumed to be quite noisy, the conflict indicator is low-pass filtered, for which a simple first-order filter is designed:

$$Ind_f(t_k) = \alpha \cdot Ind_f(t_{k-1}) + (1-\alpha) I(t_k) \quad ; \quad \alpha = e^{-\frac{\Delta_t}{\tau_f}} \tag{A.3}$$

where Ind_f is the low-pass filtered signal. The transfer function in the Z-domain is:

$$H(z) = \frac{1-\alpha}{1-\alpha z^{-1}} \implies z_p = e^{j\Omega_p} = \alpha^{-1} = e^{\frac{\Delta t}{\tau_f}} \tag{A.4}$$

where z_p is the pole and Ω_p its associated digital frequency. This implies that the cut-off frequency is $\omega_c = \frac{1}{\tau_f}$ [rad/s] in the equivalent continuous-time filter. The choice α is not very critical; it should be small enough to filter most of the noise while allowing the signal to react to relatively fast changes. For this project, we choose $\alpha = e^{-\Delta t/\tau_f}$, where $\Delta t = 25$ ms is the sampling time of the sensor fusion module, and $\tau_f = 0.1$ s is a chosen time constant related to how fast the changes can be detected in the system. This corresponds to a cut-off frequency of 1.59 Hz.

4. Last, the weight is computed using a sigmoid function. A graphical example of this function is depicted in Figure A.3.

$$w(Ind_f) = 1 - f(Ind_f) \quad ; \quad f(x) = \frac{1}{1+e^{-k(x-x_0)}} \tag{A.5}$$

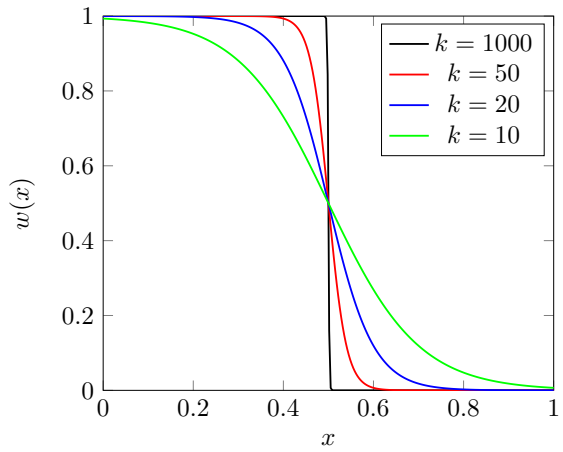


Figure A.3: Example of the designed function. $x_0 = 0.5$ for all the graphs.

The parameter k controls the slope of the curve, whereas x_0 is the value of x such that $f(x_0) = 0.5$, as can be observed in Figure A.3. These parameters must be tuned experimentally for each sensor independently. This must only be performed once, since the system should be robust to different scenarios. Machine Learning techniques (artificial neural networks, SVM and so on) would be suitable to automatically perform this operation given sensor data.

Motivation

First, we motivate the choice of the conflict metric as proposed in Equation A.2. We sum all the conflict contributions to account for the fact that lots of measurements are generating conflicts independently. An indicator based on $\max(Con)$ is not reliable since small sensor noise sometimes produces high levels of conflicts in small regions. Another key point is the **normalization**, dividing by the number of cells that have been updated in the last iteration, N_{updated}. This is directly related to the number of objects detected by the sensor, and ultimately to the particular scenario. By normalizing we create an indicator that is robust against different scenarios.

In addition, the choice of a sigmoid function provides additional robustness. The most straightforward choice would be a binary threshold function ($k \to \infty$). However, this would make the system very sensitive to the choice of the threshold value, and it might not even be possible to find a perfect one, due to the wide variety of scenarios. The sigmoid function allows to compute a real-valued weight, instead of a binary value. If the indicator is close to the threshold, Ind_0, there will not be 0-1 oscillations, but a value around 0.5 will remain instead, which will still reduce the contribution of the sensor in the fusion process.

Note: the magnitude of Ind depends also on the decay factor τ (see Section 4.5). With a large τ, the occupancy grid will have a larger history of measurements and the new ones will be in conflict more easily. On the other hand, a small τ will make the grid quickly forget the old measurements and new measurements will not generate many conflicts.

A.1.3 Concept Test

A small concept test is performed to check the viability of this approach. We take real lidar data from different experiments with a static scenario and a static ego-vehicle. The experiments were performed under normal sensor working conditions (i.e. no rain, snow or dust). To simulate an environment where the sensor would return wrong measurements, we corrupt the **range** component with noise, as follows:

$$r_i \leftarrow r_i + \delta \tag{A.6}$$

The system should not be limited to a particular noise model. For this concept test we limit ourselves to the two most common types of additive noise: uniform (δ_u) and Gaussian (δ_g), but an extended analysis with other sensor models could be done as well.

$$\delta_u \sim \mathcal{U}(-\sigma, \sigma) \tag{A.7}$$
$$\delta_g \sim \mathcal{N}(r; 0, \sigma) \tag{A.8}$$

where $\mathcal{U}$ is the uniform distribution and $\mathcal{N}$ is the Gaussian distribution.

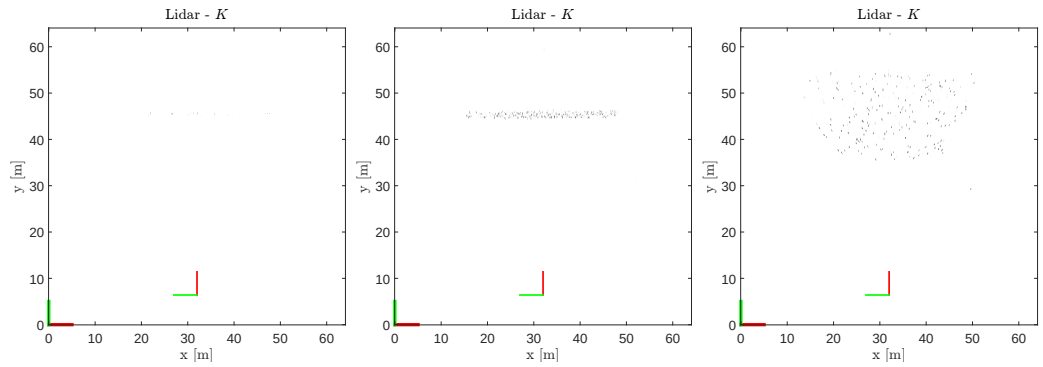
The error variance is expected to be large enough in order to be distinguishable from the intrinsic sensor noise. We try the values $\sigma = 1, \sigma = 10$ m in the experiments. To analyze the evolution of the conflict over time, only a subset of the measurements are corrupted: between the time intervals 4000 and 8000 ms approximately.

For every scenario, the whole grid update algorithm is run. The conflict indicator Ind is stored for further analysis. The first thing worth showing is the conflict matrix, K, directly related to Con, before and after the noisy measurements are processed, as shown in Figure A.4.

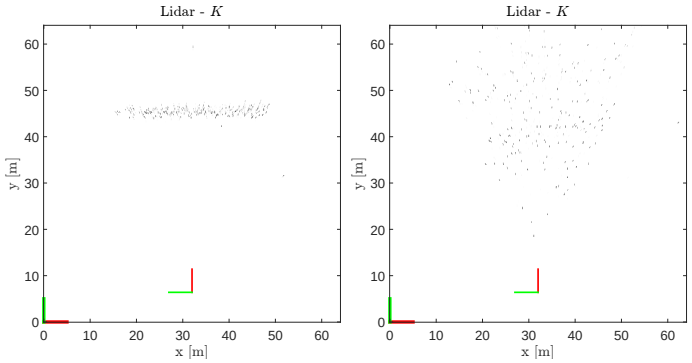
It can be seen that under normal conditions the conflict is almost null, due to the small intrinsic sensor noise. However, when measurements are corrupted with larger noise most of them produce conflict with the previous information, greatly increasing K. From K, the conflict matrix Con and the conflict indicator Ind are computed at every iteration. The result is a temporal series for all the experiments, as shown in Figure A.5. Here we can also observe the benefits of the low-pass filter mentioned before. Finally, the weights are computed using Equation A.5, with sigmoid parameters $k = 1200$, $Ind_0 = 0.0055$, tunned manually. The result is shown in Figure A.6.

Discussion

The experiment shows that the proposed approach is a feasible solution to the problem of sensor performance estimation. During timestamps 4000 ms and 8000 ms, there is a large increase on Ind, which is directly affects the sensor weight in order to reduce the sensor contribution in the fusion process. In addition, the chosen indicator measure provides a similar default value for all the scenarios, so it is not dependent on the environment at hand. Finally, the sigmoid function is a good solution to compute a weight associated with the current conflict indicator. Even in the case where Ind is close to the threshold Ind_0, the weight is reduced to 0.2-0.4, low enough so that the contribution of the sensor in the fusion process is still not relevant.

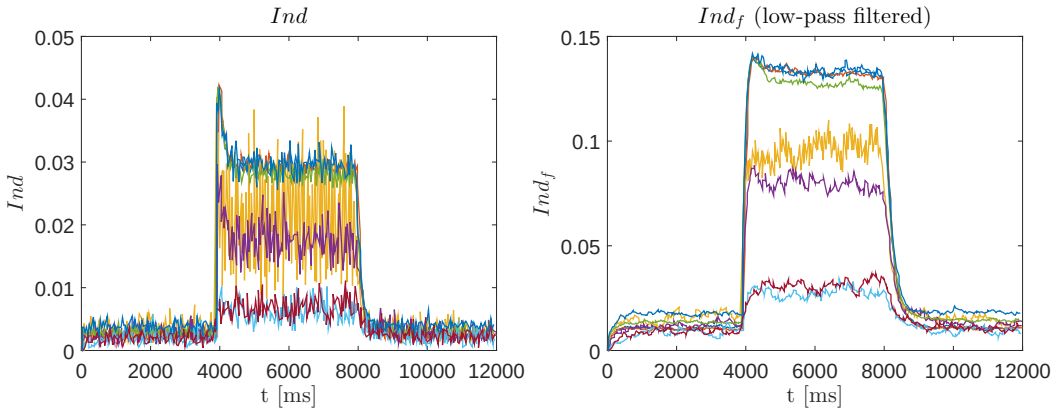


(a) No additional noise. (b) Uniform. $\sigma = 1$. (c) Uniform. $\sigma = 10$.



(d) Gaussian. $\sigma = 1$. (e) Gaussian. $\sigma = 10$.

Figure A.4: Conflict matrix K for different scenarios. White: no conflict; black: high conflict. Almost no conflict is observed under normal working conditions, whereas a large conflict appears immediately when the noise is large.



(a) Ind (b) Ind after low-pass filtering

Figure A.5: Sensor performance test for the lidar sensor. Each color represents a different scenario. Uniform random noise with $\sigma = 10m$ is added to the range measurements between t = 4000 and 8000 ms, approximately.

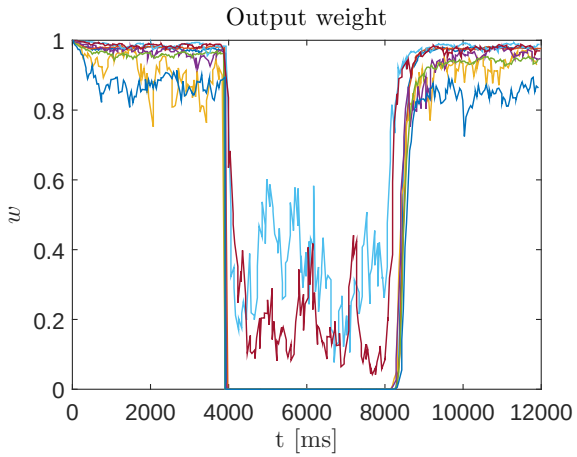


Figure A.6: Resulting weights for the concept test. Sigmoid parameters: $k = 1200, Ind_0 = 0.0055$. The weight goes to 0 when a large noise is present. Even if the noise is not too large (light blue, dark red), the weights are reduced to around 0.2-0.4, effectively reducing the impact of the sensor in the fusion process.

A.1.4 Experimental Testing

After a successful concept test, we tried to evaluate its applicability on a real test scenario. In particular, we considered the lidar sensor, whose weaknesses against heavy rain and snow are well-known. Ideally it would be required to have a controlled chamber where rain and snow could be artificially generated at will. Unfortunately we did not have access to these resources. In addition, it was very unlikely to have heavy rain or snow conditions at the moment of writing this report (spring season). Therefore, our best choice was to create artificial rain using sprinklers, as shown in Figures E.6 and E.7. Two different types of sprinklers were tested. They were located right in front of the lidar sensor to cover most of its field of view in order to generate as much noise as possible.

Unfortunately, when analyzing the received data, we realized that the **multi-echo** capabilities of the lidar allow it to overcome the common back-scattering problems with heavy rain. In the tests, we observed that there are usually no measurements corresponding to the first echo, which are probably filtered out by the lidar internally. The received measurements correspond to the second and third echoes, which successfully reached the targets without any noise. Since there was no noise to be detected (the sensor is working properly), we **could not test the proposed algorithm experimentally** with the available resources.

In conclusion, the proposed algorithm seems promising to detect sensor noise, analyzing when the sensor is contradicting itself over time. Further work would be devoted to test this algorithm under much heavier weather conditions (e.g. : in winter) to evaluate if it is an applicable solution in practice.

A.2 Advanced Conflict Management

The conflict metric returned from the Dempster-Shafer combination formula can be further leveraged, this time when performing grid fusion combining information from multiple sensors. In this case, it allows us to solve situations of **constant conflict**, i.e. situations where two sensors are continuously in conflict just because of their different physical working principles.

In Chapter 6 we evaluated some possibilities for grid fusion within the Dempster-Shafer framework. In particular, the classical Dempster-Shafer formula and the Yager's formula were compared. We observed that Yager's formula is a bit more conservative since it always transfers the conflict mass to the `unknown` state. The classical Dempster-Shafer formula gives very similar results compared to the ones in the Bayesian case.

The problem with these two approaches appears when two sensors have very similar contributions and are in conflict, giving as output an `unknown` state. From a theoretical point of view, it makes sense: if two sensors are supposed to be reliable and they are in conflict, then it is not really possible to know the actual state.

After performing some experiments, we realized that these conflicting situations increased the *false negative rate*, i.e. objects that were actually detected by one sensor ended up not being detected since other sensor detected it as empty space. We believe it would be better to give more weight to the sensor that is detecting `occupied` in these situations. Indeed, it is better to have a high false positive (FP) rate than a false negative (FN) rate, since it is always more dangerous to crash into an undetected obstacle. Therefore, we propose to modify the Dempster-Shafer combination formula as follows:

$$m_{12}(O) = \min\{1, (m_1 \textcircled{\cap} m_2)(O) + \overline{K}\} \tag{A.9}$$

$$m_{12}(E) = (m_1 \textcircled{\cap} m_2)(E) \tag{A.10}$$

$$m_{12}(\Theta) = 1 - m_{12}(O) - m_{12}(E) \tag{A.11}$$

where $\overline{K}$ is the average conflict over the last N_K iterations. We select $N_K = 10$, which corresponds to a temporal window of 250 ms, slow enough to avoid outliers but fast enough to detect constant conflicts and be able to quickly detect objects before colliding with them.

Motivation

The intuition behind this modification is similar to the one applied in Yager's rule. Instead of transferring the conflict mass to the `unknown` state (Yager), we choose to transfer it to the `occupied` state, in order to increase the detection rate. This obviously makes our approach clearly **biased towards `occupied` states**, but it might be desirable: the cost of missing an obstacle is much larger than that of a false detection.

Last, it can be observed that we use the *average* conflict, $\overline{K}$, instead of the normal conflict. The $\min$ operator is required to ensure $m_{12}(O) \leq 1$. This acts as a low-pass filter and should help avoiding spurious measurements: the contribution is only effective if the contradiction between the sensors remains over time.

A.2.1 Experiments

We apply this technique to one of the experiments performed in Scenario 4. Figure A.7 shows the conflict that appears when fusing the lidar and radar grids. The relevant conflicts for us are located at $(x, y) = (20, 20), (38, 30)$ approximately, where two objects are not detected in normal conditions. Then, we evaluate the performance in terms of detection rate compared to the classical Dempster-Shafer combination formula, as shown in Figure A.8.

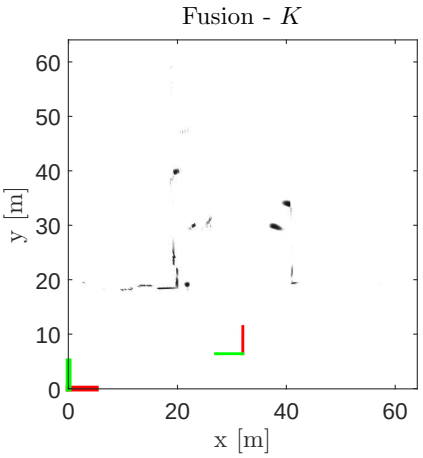
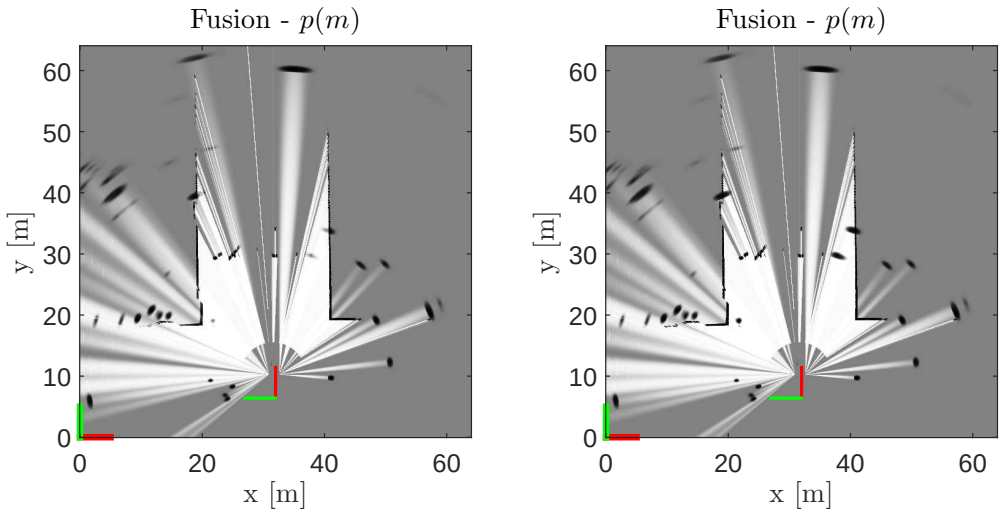


Figure A.7: Conflict when fusing radar and lidar in Scenario 4.



(a) Classical Dempster-Shafer formula.

(b) Improved formula

Figure A.8: Modified Dempster-Shafer formula for grid fusion. Two additional objects are detected after solving the conflict.

It can be clearly observed that the detection rate is greatly improved by this method. In particular, two main conflicts were solved. First, there was a post at $(x, y) = (20, 20)$ m

that was detected by the radar; however, the lidar detected the wall behind it and therefore set everything else to empty space. With this approach the cells belonging to the post have a much larger likelihood and is detected in the decision process. Furthermore, there was a corner reflector at $(x, y) = (38, 30)$ m approximately. Again, the radar detected it but the lidar detected the wall behind it. This generated a large conflict, as can be seen in Figure A.7. With the new combination formula, the resulting grid also succeeds to detect this object, hence reducing the false negative rate. The walls also are better defined, especially in the region around $(x, y) = (20, 20)$ m. This approach however does not allow the sensors to correct each other. For instance, there is a radar detection right behind the corner reflector, at around $(x, y) = (40, 35)$ m. The lidar would have refined that as empty space until hitting the wall. A similar case can be found at $(x, y) = (20, 40)$.

Discussion

This section shows another way to leverage the advantages of the Dempster-Shafer theory. The detection rate is increased due to the management of conflicts. However, we must note the implications of applying this modified Dempster-Shafer combination rule: the grid will be biased towards occupied states. Whenever two sensors are in conflict, only the belief mass associated to the `occupied` state will always be increased by adding the conflicting information. We can therefore expect a larger false positive rate as well, apart from the reduction of the false negative rate. However this is not an issue in our case since our sensors are not very noisy and quite reliable.

From a practical point of view in the context of active safety for autonomous vehicles, we believe it is beneficial to apply this new rule, even if it increases the FP rate. In this case the FP and FN rates have clearly different costs: it is much worse to crash into a non-detected obstacle than to try to avoid a non-existent object. In addition, further logic could be applied at the decision level to determine whether the obstacle should be avoided.

Appendix B

Bayes Filter: Derivation

As discussed in Chapter 4, the Bayes Filter is an efficient formulation to compute the posterior probability of the map given a set of measurements and vehicle poses:

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

We present the derivation for a single cell; the result is applicable to all of them, given that they are considered to be **independent**. We start by applying the Bayes' rule in order to filter out the latest measurement, $\mathbf{z}_t$:

$$p(m_{ij}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}) = \frac{p(\mathbf{z}_t|m_{ij}, \mathbf{x}_t, \cancel{\mathbf{z}_{1:t-1}}, \cancel{\mathbf{x}_{1:t-1}}) \cdot p(m_{ij}|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1}, \cancel{\mathbf{x}_t})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t})} \tag{B.2}$$

Some terms can be removed (crossed) from the probability distribution by taking into account the dependence relations, as shown in Figure 4.2. The factor $p(\mathbf{z}_t|m_{ij}, \mathbf{x}_t)$ is called the *forward sensor model*, and can be expanded applying Bayes' rule:

$$p(\mathbf{z}_t|m_{ij}, \mathbf{x}_t) = \frac{p(m_{ij}|\mathbf{z}_t, \mathbf{x}_t) \cdot p(\mathbf{z}_t|\mathbf{x}_t)}{p(m_{ij}|\mathbf{x}_t)} \tag{B.3}$$

The factor $p(m_{ij}|\mathbf{z}_t, \mathbf{x}_t)$ is called the *inverse sensor model* in the literature. Replacing the previous equations into Equation B.2, we obtain:

$$p(m_{ij}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}) = \frac{p(m_{ij}|\mathbf{z}_t, \mathbf{x}_t) \cdot p(\mathbf{z}_t|\mathbf{x}_t) \cdot p(m_{ij}|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t}) \cdot p(m_{ij}|\mathbf{x}_t)} \tag{B.4}$$

The probability of the `empty` state can be computed analogously:

$$p(\overline{m_{ij}}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}) = \frac{p(\overline{m_{ij}}|\mathbf{z}_t, \mathbf{x}_t) \cdot p(\mathbf{z}_t|\mathbf{x}_t) \cdot p(\overline{m_{ij}}|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t}) \cdot p(\overline{m_{ij}}|\mathbf{x}_t)} \tag{B.5}$$

Next, Equations B.4 and B.5 can be divided in order to cancel out some terms.

$$\frac{p(m_{ij}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t})}{p(\overline{m_{ij}}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t})} = \frac{p(m_{ij}|\mathbf{z}_t, \mathbf{x}_t)}{p(\overline{m_{ij}}|\mathbf{z}_t, \mathbf{x}_t)} \cdot \frac{p(m_{ij}|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1})}{p(\overline{m_{ij}}|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1})} \cdot \frac{p(\overline{m_{ij}})}{p(m_{ij})} \tag{B.6}$$

Finally, taking the logarithm of Equation B.6 we obtain the so-called *log-odds ratio*, $l(m_{ij})$. It provides the final recursive update formula of the Binary Bayes Filter, which is applied for every cell in the occupancy grid:

$$\boxed{l_t(m_{ij}) = \log \frac{p(m_{ij}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t})}{1 - p(m_{ij}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t})} = \log \frac{p(m_{ij}|\mathbf{z}_t, \mathbf{x}_t)}{1 - p(m_{ij}|\mathbf{z}_t, \mathbf{x}_t)} + l_{t-1}(m_{ij}) - \log \frac{p(m_{ij})}{1 - p(m_{ij})}} \tag{B.7}$$

This expression is used to update the individual occupancy grids with sensor data at every iteration throughout this project.

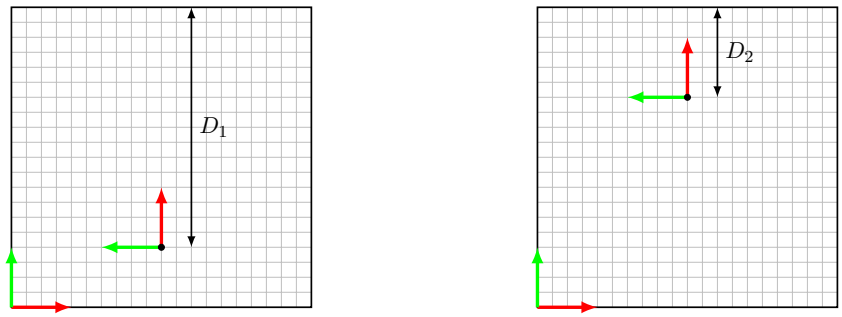
Appendix C

Grid Transformations

In this appendix we present some relevant implementation details regarding the occupancy grids. First, we will show how the grid is transformed to compensate for the ego-vehicle motion. Second, we describe how to transform the inverse sensor models from `sensor` polar coordinates (as described in Chapter 5) to `world` Cartesian coordinates.

C.1 Ego-Vehicle Motion Compensation

The vehicle is moving over time, so the local vehicle pose within the grid is updated accordingly at every iteration. Eventually, the vehicle will be located close to the borders of the occupancy grid, and therefore there will not be enough space forward to integrate measurements corresponding to obstacles that are far away, as illustrated in Figure C.1.



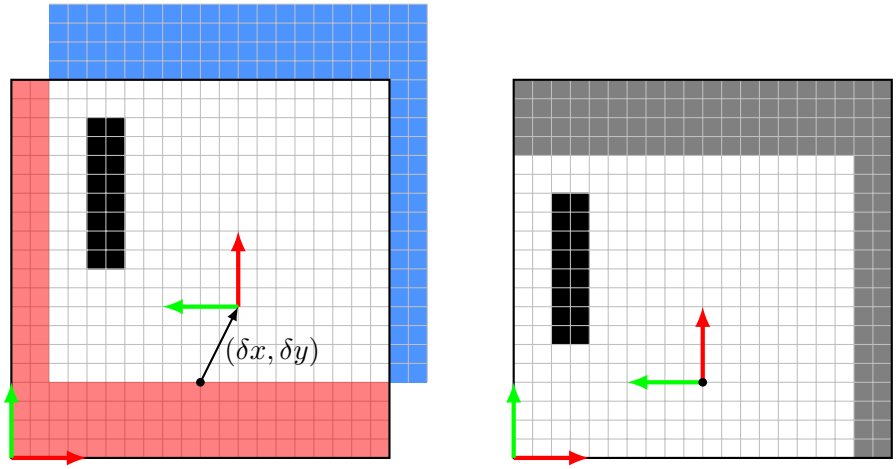
(a) (b)

Figure C.1: Ego-vehicle motion compensation: introduction to the problem. In C.1a, there is a large distance D_1 forward to detect obstacles. As the vehicle moves forward, it reaches the top of the grid (C.1b), so obstacles further away than a distance D_2 will not appear on the grid.

The solution is to transform the whole grid to account for the vehicle motion. Afterwards, the local vehicle pose is restored to its initial value. Two transformations are implemented: translation and rotation, which are described in detail in the following sections.

Translation

To account for the vehicle motion, the grid is **shifted** in the opposite direction. This is implemented efficiently in MATLAB by simply adding and removing rows and columns from the grid, as shown in Figure C.2:



(a) Before grid translation (b) After grid translation

Figure C.2: Grid translation. In C.2a, the vehicle moves a distance $(\delta x, \delta y)$. New cells are added from the top-right corner (blue), and old cells are removed from the bottom-left corner (red). In C.2b, the new cells have an unknown state (gray). The vehicle pose is reset to its original value.

The shift in X and Y direction, expressed in number of cells, is computed as follows:

$$N_x = -\left\lfloor \frac{x_v - x_v^0}{\Delta} \right\rfloor \quad ; \quad N_y = -\left\lfloor \frac{y_v - y_v^0}{\Delta} \right\rfloor \tag{C.1}$$

where $\mathbf{x}_v^0 = \begin{pmatrix} x_v^0 & y_v^0 & \theta_v^0 \end{pmatrix}^\mathsf{T}$ is the initial vehicle pose, and $\Delta = 0.1$ m is the grid resolution. The initial pose is arbitrarily chosen as:

$$\mathbf{x}_v^0 = \begin{pmatrix} \frac{W}{2} & \frac{H}{10} & \frac{\pi}{2} \end{pmatrix}^\mathsf{T} \tag{C.2}$$

where $H = W = 64$ m are the height and width of the grid, respectively. This allows the vehicle to have a large distance forward (around 50 m) to detect obstacles.

The translation operation is only performed whenever $|x_v - x_v^0| > D_{th}$ or $|y_v - y_v^0| > D_{th}$, where D_{th} is a parameter that determines how much the vehicle is allowed to move before updating the grid. A value of $D_{th} = 5$ m is chosen for the project, ensuring that the vehicle will not move too much, which would cause the measurements to lie outside the grid. A too small value is not desirable either, since then the grid would be shifted too often, reducing the computational performance. The smallest value that could be used would be $D_{th,\min} = \Delta = 0.1$ m, the grid resolution.

After performing this operation, the local vehicle pose is consequently shifted by the

same amount in order to return to its original position:

$$\begin{pmatrix} x_v \\ y_v \\ \theta_v \end{pmatrix} \leftarrow \begin{pmatrix} x_v \\ y_v \\ \theta_v \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \\ 0 \end{pmatrix} = \begin{pmatrix} x_v \\ y_v \\ \theta_v \end{pmatrix} + \begin{pmatrix} N_x \cdot \Delta \\ N_y \cdot \Delta \\ 0 \end{pmatrix} \tag{C.3}$$

It is clear that $\mathbf{x}_v \neq \mathbf{x}_v^0$ after this operation, since the displacement is a multiple of the grid resolution Δ. However this is not an issue: this operation only ensures that the vehicle pose is in a good location in the grid for storing measurements at a long distance from the vehicle.

Rotation

In addition to a simple shift, the grid also needs to be rotated when the vehicle is not pointing forward on the grid. Otherwise, it is more likely that measurements far away from the vehicle lie outside the grid. For instance, consider the extreme scenario where $\theta_v = 180^\text{o}$ and the vehicle position is at the bottom of the grid. Then it would only have a few meters forward to detect obstacles. The grid rotation is implemented as follows:

1. A set P is created, containing all the points $\{x_i, y_i\}$ in the grid. In addition, a set V contains the probability values associated with the points in P.

2. A rotation matrix R is defined:

$$R = \begin{pmatrix} \cos(\Delta\theta_v) & \sin(\Delta\theta_v) \\ -\sin(\Delta\theta_v) & \cos(\Delta\theta_v) \end{pmatrix} \quad ; \quad \Delta\theta_v = (\theta_v - \theta_v^0) \tag{C.4}$$

   The rotation is performed with an angle $-\Delta\theta_v$, compensating for the vehicle rotation.

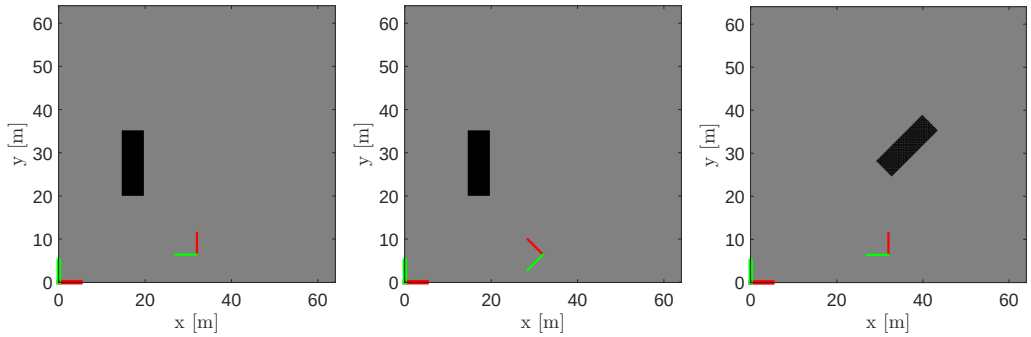
3. Every point $p = \begin{pmatrix} x & y \end{pmatrix}^\mathsf{T} \in P$ is rotated around the vehicle position, according to:

$$p' = R \cdot \left(p - \begin{pmatrix} x_v \\ y_v \end{pmatrix} \right) + \begin{pmatrix} x_v \\ y_v \end{pmatrix} \tag{C.5}$$

4. The stored values V are transferred to their new position in the grid, P'. This is performed using **nearest-neighbour association** in order to have a small computational load. This introduces errors due to discretization and rounding, which can be mitigated by more complex interpolation methods, such as bilinear interpolation.

5. Finally, the vehicle orientation is updated: $\theta_v \leftarrow \theta_v^0$.

An example of the rotation algorithm is presented in Figure C.3. It can be observed that the rotation generates additional error in form of small artifacts, especially inside the black areas, with gray dots. This is due to the continuous values returned by the sine and cosine functions, and the finite resolution of the grid. To avoid this issue, it is preferable to perform **rotations of 90º only**, for which the sine and cosine functions return ± 1 or 0. Another solution would be to have a double-sized grid and place the vehicle in the center: this way it would never be necessary to rotate the grid since the vehicle would always have enough space forward to place the measurements. Nevertheless, this comes at a cost of four times as much memory for every grid and higher computational time, especially when performing grid fusion, so we discard this option.

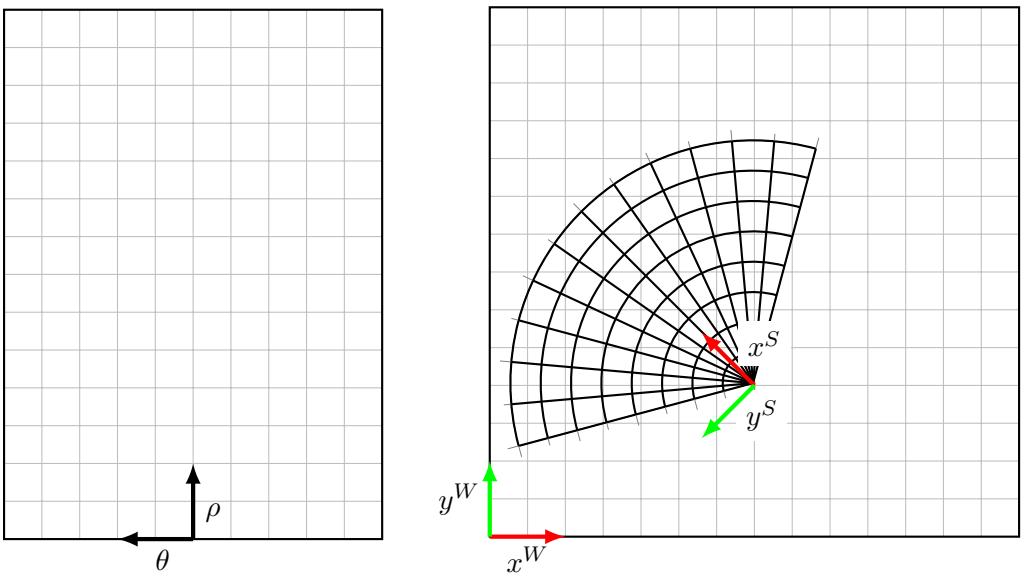


(a) Initial grid (b) Updated vehicle pose (c) Transformed grid

Figure C.3: Grid rotation: example. The vehicle is rotated 45° with respect to the original pose.

C.2 Polar to Cartesian Transformation

In Chapter 5, we developed inverse sensor models for both the lidar and radar sensors in local **polar** coordinates, i.e. $p(\rho, \theta|\mathbf{z})$. The goal is however to obtain $p(m_{ij}|\mathbf{z})$ = $p(x^W, y^W|\mathbf{z})$: the inverse sensor model in Cartesian coordinates, with respect to the `world` reference frame. The situation is illustrated in Figure C.4. In addition, the specifications of the polar and Cartesian grids are included in Table C.1 and C.2.



(a) Local polar sensor grid (b) Transformation into global cartesian grid

Figure C.4: The sensor grid is computed in a local polar coordinate frame (`sensor`). This information needs to be transfered into the global Cartesian coordinate frame (`world`).

Parameter	Value
$\rho_{\min}$	0 m
$\rho_{\max}$	60 m
$\Delta\rho$	0.025 m
$\theta_{\min}$	-80°
$\theta_{\max}$	80°
$\Delta\theta$	0.25°

Table C.1: Local polar sensor grid: specifications.

Parameter	Value
H	64 m
W	64 m
Δ	0.1 m

Table C.2: Global Cartesian grid: specifications.

The coordinate frame transformation is solved according to the following steps:

1. Given a set of points $S_p = \{\rho_i, \theta_i, v_i\}$ in polar coordinates, we transform them into Cartesian coordinates relative to the `sensor` coordinate frame to create the set S_c. In this case, v_i is the probability value associated with the cell ρ_i, θ_i in polar coordinates.

$$\begin{cases} x_i^S = \rho_i \cos\theta_i \\ y_i^S = \rho_i \sin\theta_i \end{cases} \implies S_c^S = \{x_i^S, y_i^S, v_i\} \tag{C.6}$$

2. The Cartesian points in the `sensor` coordinate frame are transformed into the `world` coordinate frame. For simplicity, we perform this operation in homogeneous coordinates:

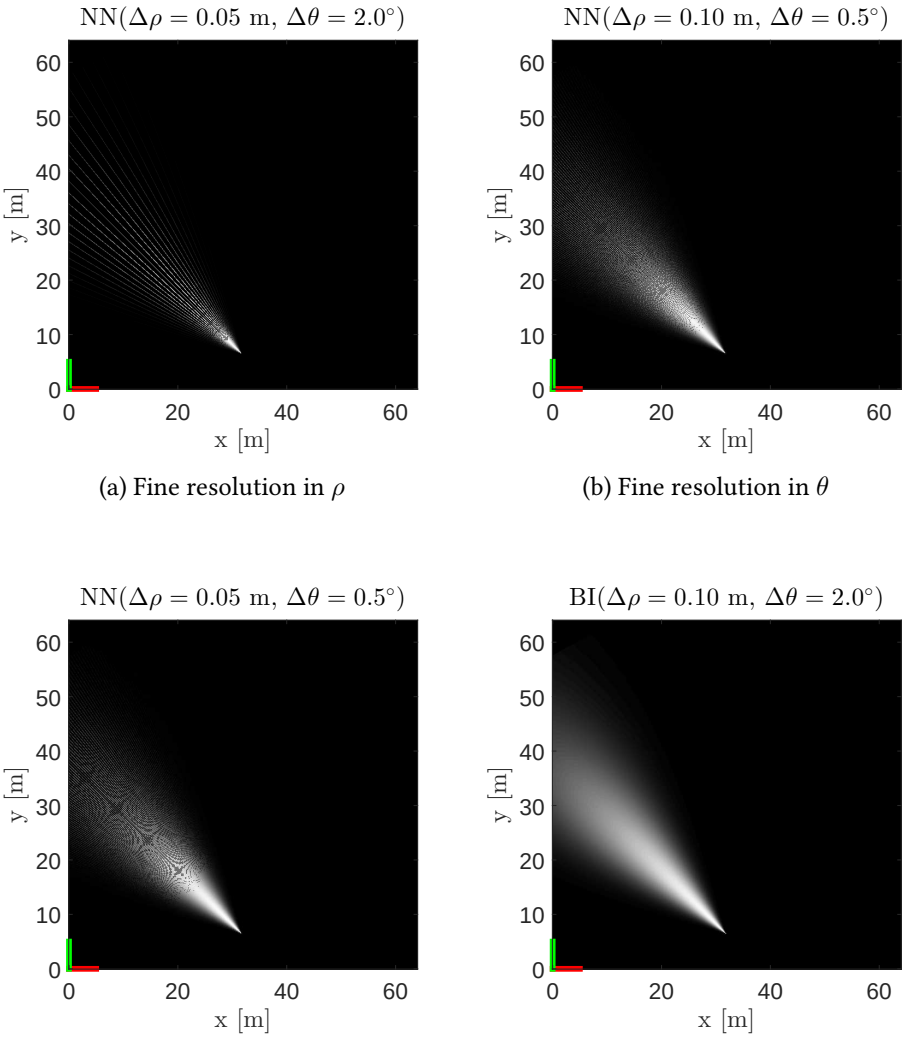
$$\begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}^W = T_w^s \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}^S = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}^S \tag{C.7}$$

   where T_w^s is the 3×3 transformation matrix from `world` to `sensor` coordinates. It can be easily computed as $T_w^s = T_w^v \cdot T_v^s$, where T_w^s is the local vehicle pose (see Figure 4.6) and T_v^s is the transformation from `vehicle` to `sensor` coordinate frame, which is given.

3. Even though the resulting points $S_c^W = \{x_i^W, y_i^W\}$ are expressed in `world` coordinates, they cannot be directly transferred into the occupancy grid, since the latter contains only a discrete set of positions. In a similar fashion as in the previous section, we perform nearest-neighbour association to transfer the probability values associated to the points in S_c^W into the global occupancy grid.

Discussion

The main drawback of the nearest-neighbour association method is that, for a short distance from the vehicle, a cell may have several values from the polar grid associated to it, whereas for long range, some cells may not even have any association at all, causing gaps in the final result. An example to illustrate this situation is presented in Figure C.5, where a two-dimensional Gaussian is transformed from polar to Cartesian coordinates.



(a) Fine resolution in ρ

(b) Fine resolution in θ

(c) Fine resolution in ρ and θ

(d) Coarse resolution + interpolation

Figure C.5: Comparison of the resulting grid for a simulated radar beam using bilinear interpolation (BI) and nearest-neighbour associarion (NN), for different resolutions of the polar grid, $(\Delta\rho, \Delta\theta)$. A much finer polar grid is required to achieve similar accuracy as bilinear interpolation, but at a larger computational cost. Furthermore, the Moiré effect appears at large distances if no interpolation is performed.

It can be seen that a fine resolution $(\Delta\rho, \Delta\theta)$ in the polar grid is required to avoid artifacts. When performing **bilinear interpolation** instead, the result is good even with a coarser polar grid resolution. The computational time is however around one order of magnitude higher.

Most of the authors in the literature use bilinear interpolation and GPU computing obtaining more accurate, real-time results [105][110]. Since we do not have access to these resources, we do not follow this approach in this project. Our solution is still acceptable with a fine enough resolution in angle ($\Delta\theta$) and/or range ($\Delta\rho$), and the artifacts do not affect the general obstacle detection module.

Implementation improvements

It is possible to obtain a more accurate result similar to bilinear interpolation but with lower computational cost, although it is less intuitive from a conceptual and programming point of view. It basically performs the same operations but in reverse order:

1. A set $S_c^W = \{x_i^W, y_i^W\}$ of all the points in the `world` Cartesian coordinate frame is created.

2. These points are then transformed into the `sensor` Cartesian coordinate frame using T_s^w to create the set S_c^S.

3. The previous points are further transformed into polar coordinates, to create the set S_p^S.

4. Next, the Gaussians are computed in polar coordinates over the previous set. Only the 3σ region of the Gaussian is considered, for higher computational efficiency.

5. Finally, the resulting values are transferred to the corresponding positions in the original global Cartesian grid, since the correspondences between the sets S_c^W and S_p^S are known.

In this project, this procedure is applied only to the case of the **radar** sensor, since the number of measurements is small and therefore it is computationally affordable. For the lidar sensor, it is much faster to integrate all laser beams in a vectorized way into the polar grid and then perform the polar to Cartesian transformation described previously. Besides, practically no artifacts are produced in this case, so there is no need for a more exact method.

Appendix D

Hardware Specifications

D.1 Velodyne Lidar

Parameter	Value
Laser wavelength	905 nm
Number of layers	64
Field of view (azimuth)	360°
Field of view (elevation)	26.8° (-2° to 24.8°)
Angular resolution (azimuth)	0.08°
Angular resolution (elevation)	0.4°
Range accuracy	< 0.02 m
Range max.	120 m
Update rate	15 Hz

Table D.1: Velodyne specifications.

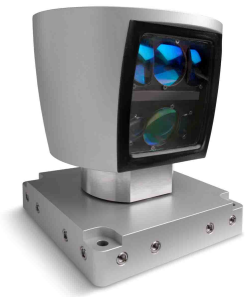


Figure D.1: Velodyne picture.

D.2 Lidar

Parameter	Value
Laser wavelength	905 nm
Number of layers	4
Field of view (azimuth)	145°
Field of view (elevation)	± 1.6°
Angular resolution (azimuth)	0.25°
Angular resolution (elevation)	0.8°
Range accuracy	< 0.1 m
Range max.	150 m
Update rate	25 Hz

Table D.2: Lidar specifications.

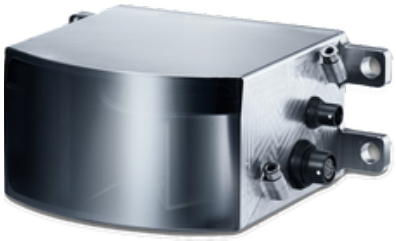


Figure D.2: Lidar picture.

D.3 Radar

Parameter	Value
Frequency	76-77 GHz
Waveform	STAR-PD
Field of view (azimuth)	150°
Field of view (elevation)	9°
Angular resolution (azimuth)	±1°
Range accuracy	0.1 m
Range-rate accuracy	0.07 m/s
Range max.	85 m
Max. number of detections	64
Update rate	20 Hz

Table D.3: Radar specifications.

Figure D.3: Radar picture.

Appendix E

Experimental Setup

In this appendix some pictures of the test scenarios are presented. In the images, red circles indicate the presence of corner reflectors, for a clearer visualization. In addition, objects are numbered for easier reference throughout the report.

E.1 Scenario 1

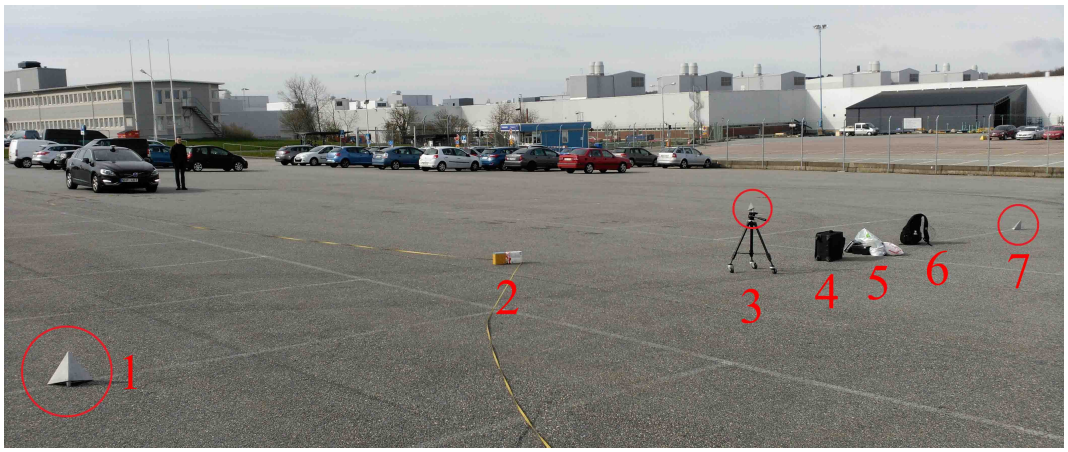
(a) Lateral view

(b) Front view

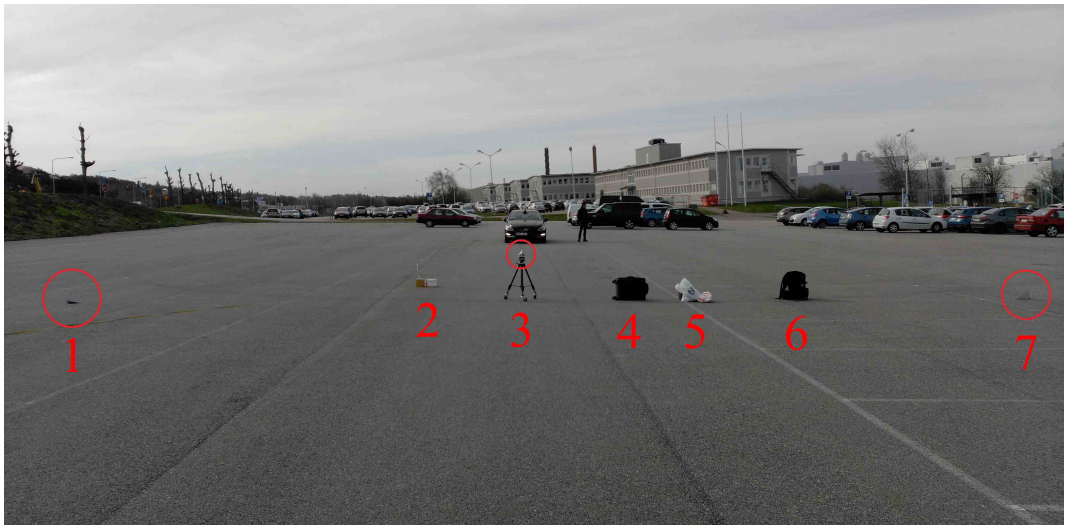
(c) View from vehicle, beyond the corner reflectors

Figure E.1: Scenario 1: environment.

E.2 Scenario 2



(a) Lateral view



(b) Front view

Figure E.2: Scenario 2: environment.

E.3 Scenario 3

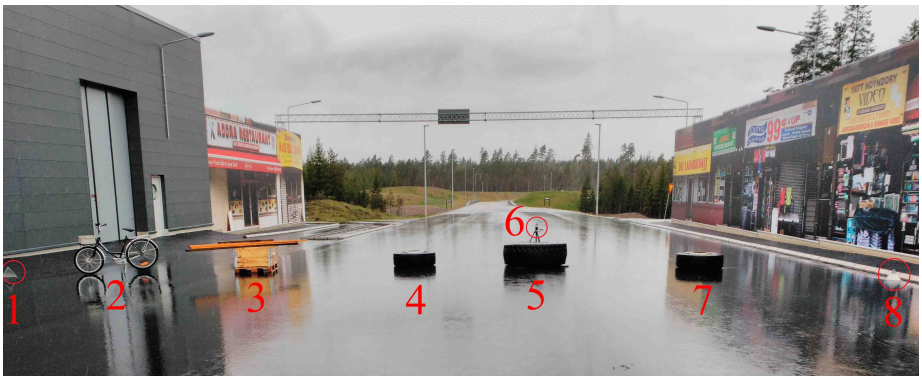
(a) Lateral view

(b) Front view

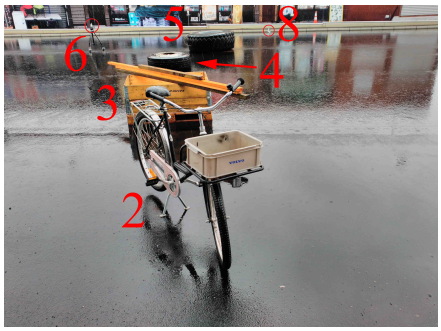
Figure E.3: Scenario 3: environment.

E.4 Scenario 4

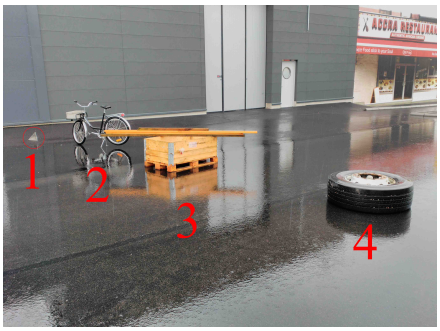
(a) Front view 1



(b) Front view 2



(c) Lateral view 1



(d) Lateral view 2

Figure E.4: Scenario 4: environment.

E.5 Other Tests

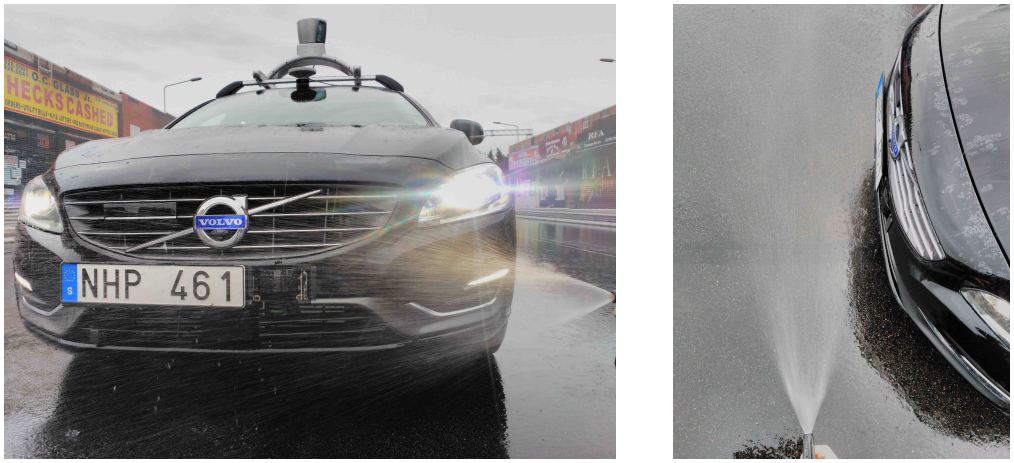
E.5.1 Radar Weighting

(a) Corner reflectors (b) Car position

Figure E.5: Multi-echo reflections with radar: experiment.

E.5.2 Lidar and Artificial Rain

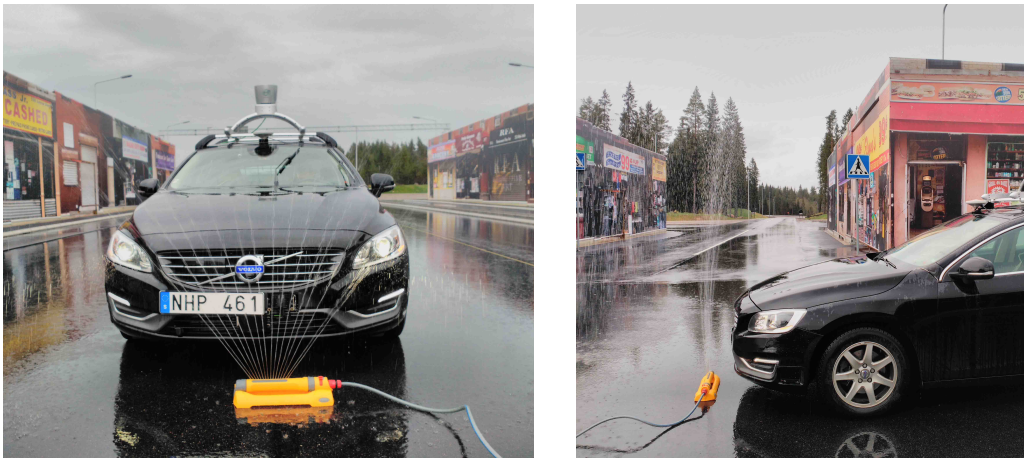
Test 1



(a) Front view (b) Side view

Figure E.6: Lidar performance with artificial rain. Test 1.

Test 2



(a) Front view (b) Side view

Figure E.7: Lidar performance with artificial rain. Test 2.