DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2019

Building an Efficient Occupancy Grid Map Based on Lidar Data Fusion for Autonomous driving Applications

MARWAN SALEM

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

To the soul of my father...

Building an Efficient Occupancy Grid Map Based on Lidar Data Fusion for Autonomous driving Applications

MARWAN SALEM
marwans@kth.se

August 7, 2019

Master's Thesis Degree Project at the Autonomous Vehicle Perception group - Scania

SUPERVISOR AT KTH: John Folkesson
SUPERVISOR AT SCANIA: Henrik Felixson
EXAMINER: Mårten Björkman

Degree Project in Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science in Embedded Systems

SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
KTH ROYAL INSTITUTE OF TECHNOLOGY

Abstract

The Localization and Map building module is a core building block for designing an autonomous vehicle. It describes the vehicle ability to create an accurate model of its surroundings and maintain its position in the environment at the same time. In this thesis work, we contribute to the autonomous driving research area by providing a proof-of-concept of integrating SLAM solutions into commercial vehicles; improving the robustness of the Localization and Map building module. The proposed system applies Bayesian inference theory within the occupancy grid mapping framework and utilizes Rao-Blackwellized Particle Filter for estimating the vehicle trajectory. The work has been done at Scania CV where a heavy duty vehicle equipped with multiple-Lidar sensory architecture was used. Low level sensor fusion of the different Lidars was performed and a parallelized implementation of the algorithm was achieved using a GPU. When tested on the frequently used datasets in the community, the implemented algorithm outperformed the scan-matching technique and showed acceptable performance in comparison to another state-of-art RBPF implementation that adapts some improvements on the algorithm. The performance of the complete system was evaluated under a designed set of real scenarios. The proposed system showed a significant improvement in terms of the estimated trajectory and provided accurate occupancy representations of the vehicle surroundings. The fusion module was found to build more informative occupancy grids than the grids obtained form individual sensors.

Keywords— Autonomous Driving, Occupancy Grids, Bayesian Inference, Lidar, SLAM, Rao-Blackwellized Particle Filter, Sensor Fusion, GPU

Sammanfattning

Modulen som har hand om både lokalisering och byggandet av karta är en av huvudorganen i ett system för autonom körning. Den beskriver bilens förmåga att skapa en modell av omgivningen och att hålla en position i förhållande till omgivningen. I detta examensarbete bidrar vi till forskningen inom autonom bilkörning med ett valideringskoncept genom att integrera SLAM-lösningar i kommersiella fordon, vilket förbättrar robustheten hos lokaliserings - kartbyggarmodulen. Det föreslagna systemet använder sig utav Bayesiansk statistik applicerat i ett ramverk som har hand om att skapa en karta, som består av ett rutnät som används för att beskriva ockuperingsgraden. För att estimera den bana som fordonet kommer att färdas använder ramverket RBPF(Rao-Blackwellized particle filter). Examensarbetet har genomförts hos Scania CV, där ett tungt fordon utrustat med flera lidarsensorer har använts. En lägre nivå av sensor fusion applicerades för de olika lidarsensorerna och en parallelliserad implementation av algoritmen implementerades på GPU. När algoritmen kördes mot data som ofta används av ”allmänheten” kan vi konstatera att den implementerade algoritmen ger ett väldigt mycket bättre resultat än ”scan-matchnings”-tekniken och visar på ett acceptabelt resultat i jämförelse med en annan högpresterande RBPF-implementation, vilken tillför några förbättringar på algoritmen. Prestandan av hela systemet utvärderas med ett antal egendesignade realistiska scenarion. Det föreslagna systemet visar på en tydlig förbättring av uppskattningen av körbanan och bidrar även med en exakt representation av omgivningen. Sensor Fusionen visar på en bättre och mer informativ representation än när man endast utgår från de individuella lidarsensorerna.

Acknowledgements

I would like to express my sincere gratitude to all those who contributed in successfully completing this project. Foremost, I am extremely grateful to Henrik Felixson, my supervisor at Scania, for giving me the opportunity to carry out my thesis research at the Sensor Fusion team, for the continuous support and guidance he offered me and for his involvement, empathy, understanding and friendship. I would also like to thank John Folkesson, my supervisor at KTH, for the valuable help through the project from forming a strong research question to adapting the right methodology to be applied.

Furthermore, I would like to express my deepest appreciation to my friend Jonas Ayddan who was always encouraging and supportive when times got rough. I would also like to thank my friend, Jörgen Persson, for the stimulating discussions and sleepless nights before my deadlines.

Finally, to my family, my mother and brothers, there are not enough words that can express my gratitude for your inspiration, love and endless support...from the bottom of my heart, thank you!

Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

ADAS	Advance Driving-Assistant System
SAE	Society of Automotive Engineers
GPU	Graphical Processing Unit
SLAM	Simultaneous Localization and Mapping
RBPF	Rao-Blackwellized Particle Filter
GPS	Global Position System
DBN	Dynamic Bayes Network
EKF	Extended Kalman Filter
Sonar	Sound Navigation and Ranging
Radar	Radio Detection and Ranging
LIDAR	Light Detection and Ranging
CUDA	Compute Unified Device Architecture
MCL	Monte Carlo Localization
LOP	Liner Opinion Pool
IOP	Independent Opinion Pool
LIOP	Logarithmic Independent Opinion Pool
IMU	Inertial Measurement Unit

Chapter 1

Introduction

The development of autonomous vehicles is advancing at a very high pace and self-driving trucks on public roads will soon see the light of day. Nowadays, vehicles have been supported with advanced driver-assisting systems (ADASs) that aid the driver with performing routine tasks and make driving easier. These cutting-edge systems are tweaked year after year by adding more features and functions. Hence, the autonomy level of the vehicle has been evolving gradually and eventually ADASs will handle the whole task of driving. With the aim of enhancing the communication and collaboration within the autonomous automotive industry, the Society of Automotive Engineers (SAE) in its International Standard no. J3016 introduced a common taxonomy and a scale for the vehicle autonomy [1] as shown in Figure 1.1.

Based on the human driver's intervention with the dynamic driving tasks, the scale consists of six levels of autonomy, spanning from *level 0*, no automation, to *level 5*, full automation. *Level 0* includes systems that only provide warnings, such as Collision Warning systems, while the driver is controlling the steering and the speed of the vehicle all the time. In *level 1*, the driving assisting system can control either the steering or the speed of the vehicle. An example of that is the Adaptive Cruise Control system where the speed of the vehicle is automatically adjusted according to the distance to the detected cars in front of the vehicle. The driving assisting systems within *level 2* can control both the steering and speed of the vehicle while the driver is monitoring the environment. An Automatic Parking assisting system that automatically performs the entire parking process belongs to this level.

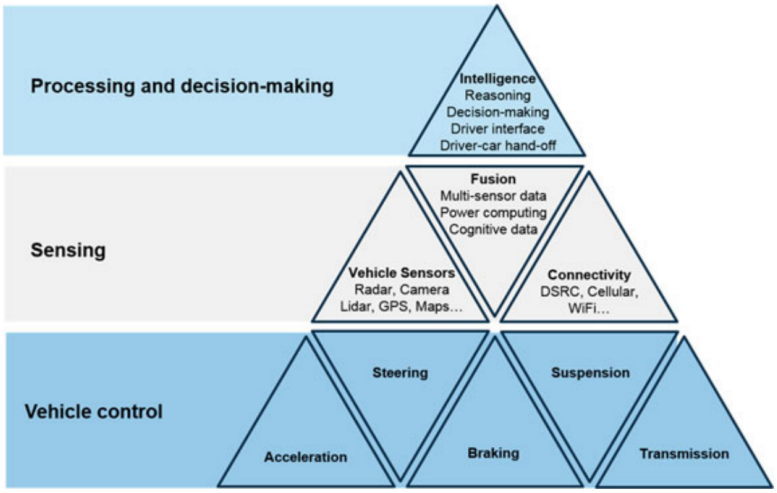
A key distinction exist between *level 2* and *level 3* where the automated driving system controls all aspects of driving including monitoring the

environment, however, the driver is expected to be ready to take over the control at all times. The Audi Traffic Jam Pilot is considered to be at automation *level 3*. On the contrary, the driver takes over the control only in certain situations in *level 4*. Vehicles with automation *level 5* can drive themselves all the time with no need for any kind of intervention by the driver.

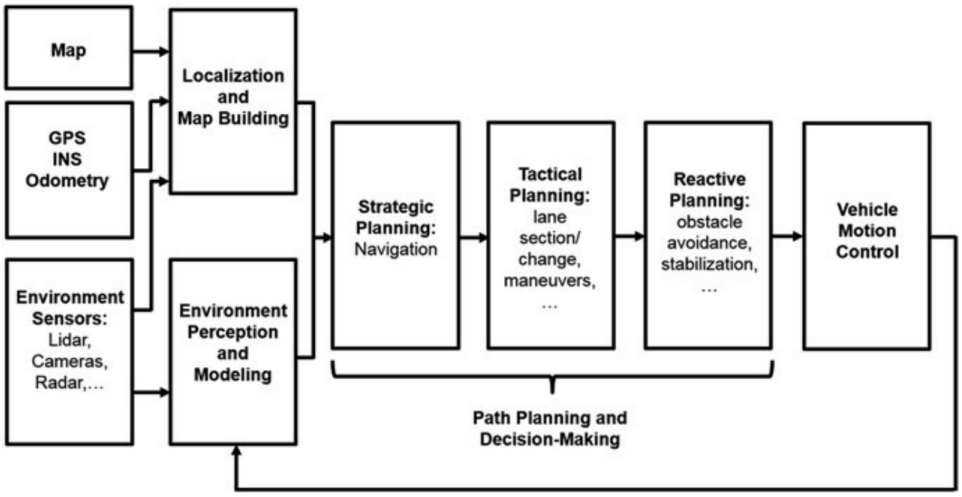
SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	**No Automation**	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	**Driver Assistance**	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	**Partial Automation**	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	**Conditional Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	**High Automation**	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	**Full Automation**	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Figure 1.1: Vehicle autonomy levels [1].

The core competencies for designing a vehicle with an autonomy level higher than *level 1* can be described in a three-layer structure [2]; vehicle control, sensing and processing and decision-making, see Figure 1.2a. The interaction between these competencies along with the vehicle's interaction with the environment are shown in Figure 1.2b. The Environment Perception and Modeling block refers to the ability of the autonomous vehicle to use the sensory system to collect information about the environment and extract relevant features from these information, developing a contextual understanding of the surroundings. The Localization and Map Building is a fundamental block that enables the autonomous driving which refers to the vehicle's ability to utilize the collected data to achieve a global map of the surroundings while maintaining its position with respect to the environment. Finally, The Path Planning and Decision-Making block refers to the process of finding the optimal path between the start location of the vehicle to the desired destination while avoiding collisions.



(a) Core competencies structure.



(b) Interaction between core competencies.

Figure 1.2: Core competencies for designing a highly automated vehicle [2].

1.1 Problem Statement

Due to the reduction in manufacturing costs, modern sensor technologies such as LIDARs are becoming very popular to be used in the automotive industry within the task of environment perception. LIDARs use an optical measurement principle to measure the distance to the surrounding objects in the space and deliver those measurements in what is known as point clouds. Occupancy grids are a location based representation of the environment

containing information about free areas and presence of obstacles in the environment. Lidar sensors are especially suited for the purpose of occupancy grid mapping due to their high angular resolution and range accuracy. However, LIDARs usually suffer from light scattering under tough weather conditions and have limited field of view. Thus, an autonomous vehicle cannot depend on only one Lidar sensor to accurately map its environment and navigate through it. On the other hand, the occupancy information provided by the different embedded Lidar sensors can be fused into one grid occupancy map where static obstacles are typically represented as occupied cells.

Autonomous commercial vehicles have seen rapid progress and they will eventually become a crucial part of our transportation system. In order to fulfill the high safety standards within the automotive industry, further development is still required such that the vehicle is able to adapt on its own in a wide variety of scenarios. This project contributes to the autonomous driving research area by investigating *how the raw readings from the different Lidar sensors can be used to build an accurate and efficient global occupancy map of the vehicle surroundings while tracking its location in the environment; increasing the situation awareness of the vehicle.*

1.2 Thesis Objectives

The project aims to solve the aforementioned problem under investigation by addressing the following:

- Building occupancy grid maps techniques.
- Designing an accurate sensor model for Lidar sensors.
- Solving the simultaneous localization and mapping problem.
- Different fusion methods for integrating data from different Lidars.
- Achieving a quantitative measure of map consistency.
- Experimental evaluation on a real vehicle.

1.3 Thesis Outline

This thesis report documents and describes in details the work that has been carried out through the project and it is organized as follows. Chapter 2 starts with describing the Simultaneous Localization and Mapping (SLAM) problem, motivating for investigating its solutions and further presenting the current state of art research related to the different aspects of the problem; including the various sensors that can be used, integrating the measurements from multiple sensors and utilizing the computational capabilities of the Graphical Processing Unit (GPU). Afterwards, in chapter 3, the method followed through the project to solve the SLAM problem is described. First, achieving an occupancy representation of the environment is addressed through applying Bayesian inference theory. Next, the mathematical sensor model is illustrated. Finally, the general algorithm which is adapted from Rao-Blackwellized Particle Filter (RBPF) and applied to localize the vehicle while maintaining a map of its surroundings is presented and processing the data from the different sensors is explained. Further, the architecture of the proposed system is demonstrated in chapter 4 while highlighting the different software and hardware building blocks.

Chapter 5 describes the evaluation procedures that were designed and performed to validate the efficiency of the proposed solution in terms of both the accuracy of the resultant occupancy grids and the computational performance. The results of the experimental evaluation process are further discussed in chapter 6 and a final conclusion has been drawn.

1.4 Ethics, Society and Sustainability

According to the world health organization, more than 1.2 million people die every year due to road accidents [3]. Fully autonomous cars have the potential to improve the road safety and save thousands of lives per year by eliminating traffic accidents in relation to driver's mistakes. Moreover, autonomous transportation will not only enhance the mobility of elderly people and those who cannot drive themselves but will also optimize the traffic flow and energy use through platooning and automated ride-sharing[4].

As the autonomous vehicles are expected to promote human well-being, they are expected to distribute the inevitable harm as well. Such distribution arises very challenging ethical conundrums. In scenarios where someone's death is unavoidable, a self-driving car has to decide on its own, within

fractions of the second, who should die and who should live; taking an ethical decision. The main challenge here is to define global principles that should control the ethical behaviour of autonomous vehicles. Those principles should further be adapted by the automotive industry to design moral algorithms for autonomous cars and should be used by the policymakers to regulate them. In an attempt to form a broad ethical framework for autonomous vehicles to follow in the future, a recent survey that was published in *Nature* gathered data from millions of people from all over the world through deploying an online experimental platform, the Moral Machine [5]. The survey was concluded with three strong preferences that should serve as building blocks for global machine ethics codes, namely, sparing human lives, sparing more lives and sparing young lives.

Chapter 2

Background

This chapter aims to provide the reader with a literature review of the different topics that are related to mapping and sensor fusion in the context of autonomous driving. The chapter starts with a brief motivation for utilizing Grid-based SLAM (Simultaneous Localization and Mapping) within the task of increasing the situational awareness of an autonomous vehicle. Then, a detailed description of the SLAM problem is presented while addressing the state-of-art approaches applied to solve it. The chapter proceeds with an analysis of the different sensors and data fusion approaches that are used for perceiving the environment and solving the SLAM problem. Finally, a review of the related work within the topic of using the powerful GPU (Graphical Processing Unit) while implementing mapping algorithms is shown.

2.1 Feature- and Grid-based Simultaneous Localization and Mapping

Perception of the surroundings and mapping the environment are fundamental proprieties for a vehicle to be able to drive autonomously. Building an accurate map of the environment is of high importance for an autonomous vehicle to determine the free drivable corridor, path planning and obstacle avoidance, and to eliminate the estimation error of the vehicle position with respect to some reference frame, landmark localization.

There exists two main approaches for representing the map. One popular map representation approach is *feature based mapping* such as topological maps or landmark-based maps. Whereas such approach is reliable and computationally compact, it assumes some prior knowledge about the environment

to be known in advance and relies on predefined feature extraction techniques [6]. Another alternative approach to map the environment is *occupancy grid* where the environment is divided into an evenly spaced grid of cells and each cell is considered to be either free or occupied. Occupancy grid maps are able to represent arbitrary features and provide detailed representations without the need for prior information about the environment. Although occupancy grid framework is a computationally intensive and memory consuming algorithm, one of the major advantages of the framework is that all the cells in the grid are independent of each other, which in turn makes it possible to be efficiently parallelized and executed in a GPU (Graphical Processing Units).

In order to build an accurate map of the environment, an autonomous vehicle need a good estimation of its location. The task of navigating through an unknown environment while building a map is often referred to, in the literature, as the simultaneous localization and mapping (SLAM) problem [7]. SLAM is considered a complex problem because it needs to solve two dependent problems; estimate the vehicle position relative to the map and build the map based on both the processed sensory information and the position. However, implementing SLAM solutions has recently increased within the field of the autonomous driving research due to the localization accuracy that can be achieved through SLAM, which overcomes the GPS accuracy, along with the resultant intuitive awareness of the vehicle environment.

2.2 SLAM solution

The SLAM problem is often considered as a Dynamic Bayes Network (DBN), as shown in Figure 2.1, while Markov assumption is being applied to simplify the problem and save the useful information in the current state. In this DBN structure, controls u_t and measurements z_t are the observed variables which are used to estimate the hidden variables; vehicle trajectory $x_{1:t}$ and the map m. Solving the probabilistic DBN is achieved by first estimating the vehicle position within an existing map, then incorporating new measurements and updating the map [8]. Through an intensive research during the last decade, different estimation techniques have been proposed to address and solve the SLAM problem. Those estimation techniques can be roughly classified according to the underlying model of the environment mentioned in the previous section [6]. Extended Kalman Filter (EKF) based algorithms in combination with predefined landmarks are one of the most effective approaches to reach a correlated posterior estimation about landmark

maps as well as the position. However, the Gaussian distribution assumption of the sensor model and the linear approximation of the motion model applied by EKF filters, when violated, lead the filter results to diverge [9]. Moreover, EKF filters assume uniquely identifiable landmarks which is not a typical real world case and violating it leads to inconsistent maps.

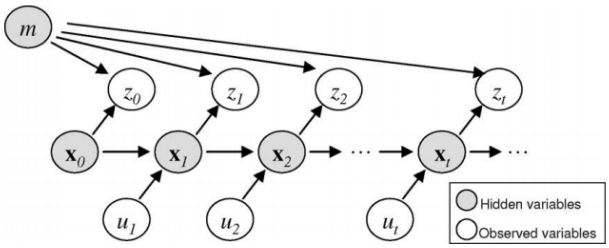


Figure 2.1: SLAM problem as a DBN structure [10].

To remedy the issues discussed with using EKF algorithms, Rao-Blackwellized particle filters (RBPF) was introduced by Doucet *et al.* [11] as an effective way to solve the SLAM problem while building accurate occupancy grid maps. RBPF approximates the posterior distribution over the potential vehicle trajectory by a proposal distribution, samples the proposal distribution using a number of particles where each particle stores a possible vehicle trajectory and a map of the environment, and assigns an importance weight for each particle, by calculating the ratio between the distributions. Further, a selection (*resampling*) step is performed where the low weighted particles are replaced by high weighted ones. The FastSLAM algorithm intoduced by Montemerlo *et al.* [12] used Rao-Blackwellized Particle filter to solve landmark based SLAM problems.

The RBPF algorithm was introduced as a generic mapping framework to solve the SLAM problem. However, it left open two major challenges to be further investigated by the literature; how the proposal distribution should be computed and when the resampling step should be performed. The first challenge implies achieving accurate proposal distribution; the more accurate the proposal distribution, the less particles required to build an accurate map, reducing the memory requirements and computation time. While the second challenge implies avoiding the *particle depletion* problem [9]; removing accurate particles when resampling. In many RBPF based SLAM applications, since the trajectory of the vehicle evolves according to the vehicle motion, the proposal distribution is chosen to be modeled as

the probabilistic odometry motion model. Hähnel *et al.* [13] presented an improved motion model that reduces the number of the required particles by combining the scan matching framework with the Rao-Blackwellized particle filtering. G. Grisetti *et al.* [14] improved the algorithm proposed in [13] by using the scan matching procedure along with the odometry information to determine the most-likely position which is further used for creating the next generation of the particles. That in turn resulted in an accurate proposal distribution with less variance which is computed on the fly instead of using a fixed distribution as in [13]. The authors of [15] reduced the risk of the the particle depletion problem by following a selective resampling technique. Instead of resampling every sampling instant, resampling was performed only when necessary based on a threshold over the weights of the particles.

2.3 Sensors for SLAM

A wide variety of modern sensory technologies are becoming very popular to be used in the automotive industry within the task of environment perception and SLAM. An analysis of these sensory technologies is described further in this section to understand their strengths and weaknesses, provide an insight of their performance and motivate the choice of Lidar sensors for this project.

In [16] Zaffar *et al.* discussed the different types of sensors used in solving the SLAM problem. Monocular camera is one of the most popular sensors used for SLAM due to its compactness and cheap cost along with the high resolution and detailed images they can provide about the environment. For example, A 6DOF real time visual SLAM system based on a monocular camera is presented in [17]. However, monocular cameras do not provide depth information which leads to complex SLAM algorithms. On the contrary, stereo cameras can calculate depth information using the disparity of two camera images looking at the same scene. An extensive experimental comparison between monocular based SLAM approach and stereovision based SLAM approach was done by the authors of [18]. Since cameras are passive sensors, they do not rely on any emission for mapping the environment and they have low power consumption. On the other hand, cameras are sensitive to changes in illumination and they do not perform well under difficult weather conditions like rain, snow and dust.

Sonar sensors locate objects by emitting a sound signal and measuring the *Time-of-Flight*: the total time taken by the sound wave from emission to reception upon bouncing off an object. Sonar sensors can easily suffer from the *Specular reflection* phenomenon where the reflected signal bounces away from the sensor [19]. Moreover, similarly to cameras, they are very sensitive to weather conditions due to the dependency of the sound waves on temperature. However, there exist SLAM applications based on sonar sensors in the literature [9]. Radar sensors are based on the same principle as sonars, but emitting radio waves instead. Radar sensors, compared to cameras and sonars, have a very robust performance in rough weather conditions and vibrations [20]. In addition to their good range resolution, they are able to measure the radial speed of the moving objects. However, the main drawback of radars is their very poor angular resolution. Degerman *et al.* addressed the ability of building an occupancy grid map using radar sensors [21].

Lidar sensors emit laser beams at fixed angular steps while rotating around a vertical axis, measure the distance to the surrounding objects using the *Time-of-Flight* principle and deliver those measurements in what is known as point clouds; generating a 3D visualization of the environment. In addition to the accurate range measurement, Lidars, on the contrary to radars, provide high angular resolution. Moreover, Lidars, by measuring the intensity of the reflected laser beam, return very accurate depth information [16], therefore compensating for the main monocular vision-based systems weakness. Due to their robust and reliable measurements, Lidars have become one of the most popular sensor options while solving the SLAM problem. An online 3D SLAM approach was recently introduced by Droeschel *et al.* in [22].

On the other hand, Lidar measurements are corrupted in environments with heavy rain and bad mist and dust conditions [20]. Thus, we cannot depend on only one Lidar sensor in those situations. In order to create a robust representation of the vehicle environment, it is therefore necessary to combine different Lidar sensory information such that they can compensate for each other's drawbacks. This approach is widely known as sensor fusion, which is further investigated in the following section.

2.4 Sensor Fusion

As can be observed from last section, an autonomous vehicle is often supported by multiple sensors to help with the environment perception and decision-making processes. Thus, a lot of effort within the autonomous driving research area has recently been dedicated towards the sensor fusion technology. Sensor fusion is the study of efficient approaches for combining the sensory information from different sources into a representation that is not feasible from individual sources and that is supportive for an automated decision-making process [23]. A comprehensive study of the state-of-art sensor fusion methodologies and challenges was introduced by Khaleghi *et al.* [24]. According to the type of the input sensory data to the system and the obtained output information, the fusion processes are defined as a three-level hierarchy [25], as shown in Figure 2.2. Whereas the *medium-level* fusion integrates information in terms of features extracted from the raw sensory data, the *high-level* sensor fusion processes sensory information in a symbolic representation or in other words; decisions.

On the other hand, the *low-level* fusion processes the raw sensor data, pixels or signals, in a central manner. Thus, one main advantage of performing sensor fusion at a low level is minimizing the information loss [26]. The Occupancy Grid framework, introduced by Elfes and Moravec [27], is one of the most popular approaches to fuse sensor information in a low-level. Occupancy grids divide the environment into an evenly spaced grid of cells and estimate the probability of each cell for being occupied based on the raw sensor measurements. The main advantage of using occupancy grid as a framework for sensor fusion is that it is easy to interpret any type of sensory measurements in cell-based occupancy information; occupancy grid fusion framework only requires a sensor model describing the probabilistic relation between the sensor measurement and the state of the cell being occupied. An algorithm for generating occupancy grid maps which relies on physical forward sensor models instead of the inverse ones was first presented by Sebastian Thrun [28].

However, the occupancy grid requires an inference theory to estimate the probability of occupancy for each cell. In [29] Ivanjkoet *et al.* did a comprehensive experimental comparison of the different occupancy grid mapping approaches, based on sonar sensor data, where maps constructed by Bayesian inference approach showed fair accuracy compared to the other approaches taking into consideration the memory resources requirements. Another common way of performing data fusion in low-level is by first fusing

all sensor data and then applying a single Bayesian estimator on a single sensor model.

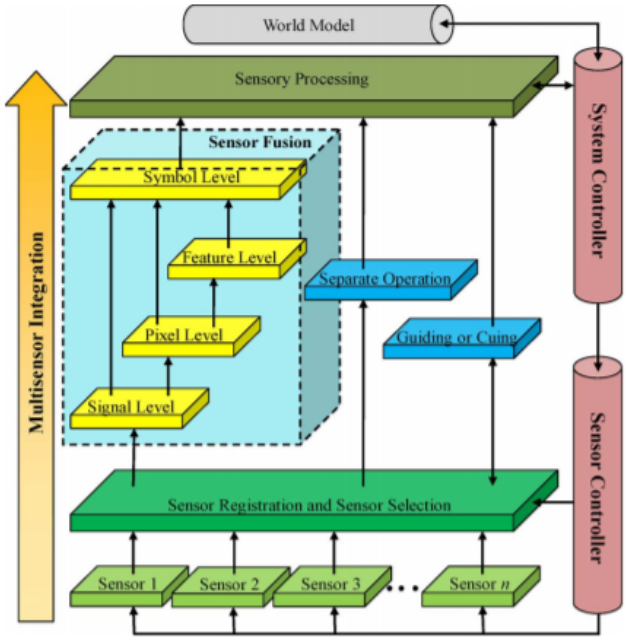


Figure 2.2: Data fusion levels [25].

Figure 2.3 shows the methodology proposed by W.Elmenreich [30] for designing an efficient multi-sensor fusion system for autonomous vehicles. As can be observed, the process of deciding the sensory data to be fused is an application-dependent process. For the task of lane detection, Homm *et al.* [31] fused the measurements from both a laser scanner and a monocular camera, showing that both sensors complement each other and form a robust and continuous lane detection system. While in [32] a Lidar along with a stereo camera were used to design an object detection system within a dynamic environment. The ultimate goal of this project is to achieve an accurate map of the static surrounding environment of a truck which can be further reliably used, through applying state-of-art computer vision technologies, for finding interesting features, path planning and obstacle avoidance. As mentioned in the previous section, Lidar as a sensor is especially suited for the purpose of localization and mapping the surroundings due to its excellent accuracy and resolution. In order to overcome its unreliable performance within bad weather conditions, a combination of a long range radar and a short range radar were used along with the Lidar by

[33] to map the environment where experimental scenarios on a Volvo truck showed good results.

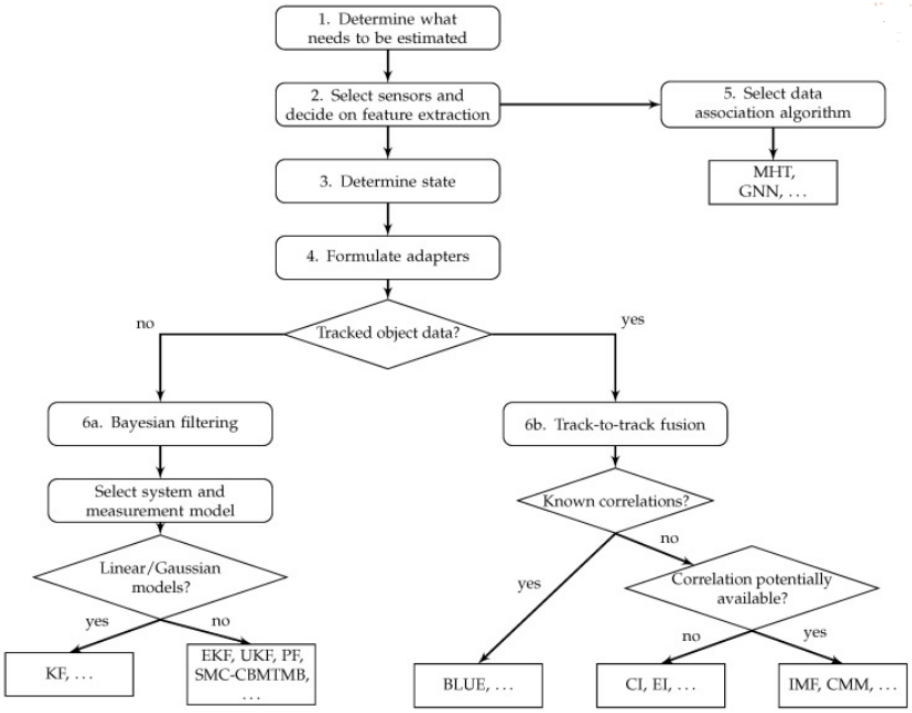


Figure 2.3: Different steps within the process of designing a multi-sensor fusion system [30].

In this project, we will investigate building an accurate occupancy grid map of the vehicle surrounding using a sensor setup that consists of multi-lidar configuration; compensating for each other's drawbacks within rough weather conditions. The fusion of the different Lidar data will be performed in a low-level manner following the two aforementioned techniques: occupancy grid fusion, constructing a sensor model for each Lidar and then fusing the occupancy information provided by the different embedded Lidar sensors into one fused occupancy grid map, and fusing the Lidar measurements into one sensor model before interpreting these measurements into an occupancy grid representation. Furthermore, a comparison between the two approaches will be held by evaluating the accuracy of the occupancy information resultant from each approach within different scenarios.

2.5 SLAM and GPU

The main drawback of low-level sensor fusion is the huge amount of data that needs to be processed. The project aims to make use of the computational capabilities of the Graphical Processing Unit (GPU) to deal with the computational heavy processes involved in the grid mapping algorithm. However, due to the time limitations of the project timeline, addressing the issue of the most efficient GPU implementation is not part of the work objectives. Florian *et al.* [34] addressed utilizing the GPU to overcome the limitations of classical occupancy grid computation in automotive environments and introduced a method for fast computation of occupancy grid maps with laser range-finders and radar sensors. While Diego *et al.* [35] worked on the applicability of GPU computing within the area of robotic mapping with laser rangefinders with a novel grid matching method. In [36] the different methods to accelerate the particle filter based SLAM algorithms were explored using NVIDIA's general purpose GPU computing platform - CUDA - where the most time consuming step, particle weight calculation, was imported into CUDA. The authors of [37] addressed the problem of switching coordinate systems by comparing different types of mapping algorithms: the exact algorithm, the sampling approach and the texture mapping approach based upon a GPU computation.

2.6 SLAM Benchmark

As can be seen from section 2.2, different algorithms and estimation techniques have been addressed toward solving the SLAM problem. However, there is no standard approach for comparing the performance of those different algorithms. For instance, in Feature-based SLAM algorithms, the Mahalanobis distance between the estimated and true location of the landmarks are often used as a measure of accuracy. While in Grid-based SLAM algorithms, the resultant maps are compared based on visual inspection. Thus, there is a huge need, in the SLAM research community, for a common approach to construct meaningful comparisons between the different algorithms.

In [38], Kümmerle *et al.* proposed a metric for measuring the performance of a SLAM algorithm by obtaining the relative geometric relations, displacements, between the output poses of the algorithm along the trajectory of the vehicle and calculating the deformation energy needed to transfer the estimated relations to the true ones (ε):

$$\varepsilon(\delta) = \frac{1}{N}\sum_{i,j} trans(\delta_{i,j} \ominus \delta^*_{i,j})^2 + rot(\delta_{i,j} \ominus \delta^*_{i,j})^2 \tag{2.1}$$

Where $\delta_{i,j} = x_i \ominus x_j$ is the relative transformation from estimated pose x_i to estimated pose x_j, $\delta^*_{i,j} = x^*_i \ominus x^*_j$ is the corresponding true transformation, N is the number of relations, $trans(.)$ and $rot(.)$ are the translational and rotational components of the relations. Since this metric is not based on comparing the resultant maps but the poses instead, it allows for comparing SLAM algorithms which are based on different map representations, for instance Feature- and Grid-based algorithms, and that are applying different sensor models.

On the other hand, the main disadvantage of this method is that it includes intensive manual work; a human operator that is aware of the topology of the environment where the datasets have been recorded has to inspect the estimated SLAM poses, visualize the sensors observations and construct the true relations. The exact procedure for obtaining the reference relations in indoor environments is described in [38]. For large outdoor environments, Kümmerle *et al.* adapted Monte Carlo Localization (MCL) framework to process aerial images in order to extract the reference relations.

It is important to note that the metric in equation 2.1 does not specify the relative relations that should be included in the summation; the selection of the relative displacements depends on the application at hand. The authors of [38] argue that by considering only the relative displacements between nearby poses, the local consistency of the resultant map is assessed which in turn is sufficient for evaluating the performance of localization algorithms. In addition to the proposed metric, Kümmerle *et al.* provided the relative relations that were obtained manually for the frequently used public datasets within the SLAM research community, thus forming a benchmark for evaluating the different SLAM algorithms. In this project, we will make use of the benchmark to validate the implemented RBPF SLAM algorithm as further described in chapter 5.

Chapter 3

Methodology

In this chapter, we present the methodology adapted and applied through the project. At the beginning, we explain in details the occupancy grid mapping as a framework to represent the vehicle surroundings and the application of Bayesian inference theory to achieve such representation while addressing the mathematical model of the Lidar sensor. Then we describe the Rao-Blackwellized grid-based SLAM algorithm implemented in this project. Finally, we discuss the different fusion approaches, analyze their performance and introduce the most suitable ones that have been chosen and applied.

3.1 Occupancy Grid Mapping

As an alternative framework to extracting geometric models directly from sensor readings while mapping the environment, Elfes *et al.* [39] were the first to introduce the *occupancy grid mapping* framework. The main idea behind occupancy grid maps is to divide the environment into an evenly spaced grid with fixed dimensions and fixed resolution. A random binary variable m_{ij} is assigned to each cell within the map m where this random variable refers to the occupancy state of the cell; *Occupied* or $Free$. Thus:

- For a cell with a total confidence that it is *Occupied*: $p(m_{ij}) = 1$
- For a cell with a total confidence that it is *Free*: $p(m_{ij}) = 0$
- For a cell with *Unknown* occupancy state: $p(m_{ij}) = .5$

When denoting the vacancy state of the cell ($p(m_{ij}) = 0$) as $\bar{m}_{ij}$, that in turn implies the following constraint:

$$p(m_{ij}) + p(\bar{m}_{ij}) = 1 \tag{3.1}$$

Applying the aforementioned conventions to all the cells within the map, we can efficiently describe the map m as a two dimensional grid as follows:

$$m = \{m_{ij} : 1 \leq i \leq W, 1 \leq j \leq H\} \tag{3.2}$$

Where W is the number of cells in the width dimension and H is the number of cells in the height dimension.

This kind of occupancy representation of the vehicle's surroundings can be achieved by mirroring the presence of objects in the environment; parts of the environment where obstacles exist are mapped as occupied cells. On the other hand, obstacles are detected by processing the sensor readings while maintaining the vehicle position in the map. Thus, our problem can be formulated as computing the posterior distribution over the map based on the sensor measurements $z_{1:t}$ and given the vehicle trajectory $x_{1:t}$:

$$p(m_{1:t}|z_{1:t}, x_{1:t}) \tag{3.3}$$

3.1.1 Bayesian Inference Theory

As we are interested in estimating only the current state of the map, m_t, we can apply Bayesian inference theory and estimate m_t in a predict-update procedure, while assuming that the current state of the map depends only on its previous state, as follows:

- **Predict**:

$$p(m_t|z_{1:t-1}, x_{1:t-1}) = \int p(m_t|m_{t-1}) p(m_{t-1}|z_{1:t-1}, x_{1:t-1}) dm_{t-1} \tag{3.4}$$

- **Update**:

$$p(m_t|z_{1:t}, x_{1:t}) = \frac{p(z_t|m_t, z_{1:t-1}, x_{1:t}) p(m_t|z_{1:t-1}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})} \tag{3.5}$$

3.1.2 Practical Approximations

The problem formulation in equation 3.3 refers to the general case where the map is dynamic and evolves over time. However, the evolution of the map is always unpredictable. That in turn results in the term $p(m_t|m_{t-1})$, in the prediction step, being hard to compute. Thus, a common approximation is to assume a **Static map** $p(m|z_{1:t},x_{1:t})$; only static obstacles are mapped while dynamic behaviours within the map are neglected. Taking this assumption into consideration, we can apply what is called *Static State Bayes filer* or *Binary Bayes filer* where only the update step is applied and the sensor readings are directly processed and used to update the probability distribution of the map.

Moreover, in order to reach a further simplified formulation of the problem, another common approximation in the literature is to assume that **the cells in the map are independent from each others**. That means there is no occupancy relation or dependency between a specific cell and its neighbouring cells given the occupancy state of this cell. Although this assumption is not true for all real situations, it has been shown as a fair assumption that allows for simple computations. Taking this assumption into consideration, our problem formulation is reduced from estimating the occupancy state of the whole map to the product of the occupancy state probability of the individual cells:

$$p(m|z_{1:t},x_{1:t}) = \prod_{i=1}^{i=W}\prod_{j=1}^{j=H} p(m_{ij}|z_{1:t},x_{1:t}) \tag{3.6}$$

3.1.3 Mathematical Derivation

Applying the Static State Bayes filter to estimate the occupancy probability of each cell:

$$p(m_{ij}|z_{1:t},x_{1:t}) = \frac{p(z_t|m_{ij},x_t,z_{1:t-1},x_{1:t-1}).p(m_{ij}|z_{1:t-1},x_{1:t-1},x_t)}{p(z_t|z_{1:t-1},x_{1:t})} \tag{3.7}$$

Applying Markov assumption where the current state of the sensor measurements z_t and the vehicle pose x_t are assumed to sum the past, the previous states $z_{1:t-1}$ and $x_{1:t-1}$, equation 3.7 shall be simplified to:

$$p(m_{ij}|z_{1:t},x_{1:t}) = \frac{p(z_t|m_{ij},x_t).p(m_{ij}|z_{1:t-1},x_{1:t-1})}{p(z_t|z_{1:t-1},x_{1:t})} \tag{3.8}$$

Where

$$p(z_t|m_{ij},x_t) = \frac{p(m_{ij}|z_t,x_t).p(z_t|x_t)}{p(m_{ij}|x_t)} = \frac{p(m_{ij}|z_t,x_t).p(z_t|x_t)}{p(m_{ij})} \tag{3.9}$$

Substituting in equation 3.8, we obtain:

$$p(m_{ij}|z_{1:t},x_{1:t}) = \frac{p(m_{ij}|z_t,x_t).p(z_t|x_t).p(m_{ij}|z_{1:t-1},x_{1:t-1})}{p(m_{ij}).p(z_t|z_{1:t-1},x_{1:t})} \tag{3.10}$$

By analogy, the vacancy state of the cell can be calculated as:

$$p(\bar{m}_{ij}|z_{1:t},x_{1:t}) = \frac{p(\bar{m}_{ij}|z_t,x_t).p(z_t|x_t).p(\bar{m}_{ij}|z_{1:t-1},x_{1:t-1})}{p(\bar{m}_{ij}).p(z_t|z_{1:t-1},x_{1:t})} \tag{3.11}$$

Dividing equation 3.10 by equation 3.11 yields to:

$$\frac{p(m_{ij}|z_{1:t},x_{1:t})}{p(\bar{m}_{ij}|z_{1:t},x_{1:t})} = \frac{p(m_{ij}|z_t,x_t)}{p(\bar{m}_{ij}|z_t,x_t)} \frac{p(m_{ij}|z_{1:t-1},x_{1:t-1})}{p(\bar{m}_{ij}|z_{1:t-1},x_{1:t-1})} \frac{p(\bar{m}_{ij})}{p(m_{ij})}$$

$$\frac{p(m_{ij}|z_{1:t},x_{1:t})}{1-p(m_{ij}|z_{1:t},x_{1:t})} = \frac{p(m_{ij}|z_t,x_t)}{1-p(m_{ij}|z_t,x_t)} \frac{p(m_{ij}|z_{1:t-1},x_{1:t-1})}{1-p(m_{ij}|z_{1:t-1},x_{1:t-1})} \frac{1-p(m_{ij})}{p(m_{ij})} \tag{3.12}$$

Taking the logarithm for the previous expression allows us to simplify the computations by moving to summation instead of multiplication:

$$log\frac{p(m_{ij}|z_{1:t},x_{1:t})}{1-p(m_{ij}|z_{1:t},x_{1:t})} = log\frac{p(m_{ij}|z_t,x_t)}{1-p(m_{ij}|z_t,x_t)} + log\frac{p(m_{ij}|z_{1:t-1},x_{1:t-1})}{1-p(m_{ij}|z_{1:t-1},x_{1:t-1})} - log\frac{p(m_{ij})}{1-p(m_{ij})} \tag{3.13}$$

Finally, denoting the log odd ratio as $l(m_{ij}|z_{1:t}, x_{1:t})$, we obtain:

$$\boxed{l(m_{ij}|z_{1:t}, x_{1:t}) = l(m_{ij}|z_{1:t-1}, x_{1:t-1}) + l(m_{ij}|z_t, x_t) - l(m_{ij})} \tag{3.14}$$

or in short:

$$\boxed{l_{t,ij} = l_{t-1,ij} + l(m_{ij}|z_t, x_t) - l(m_{ij})} \tag{3.15}$$

Equation 3.15 is the final expression used at each iteration for incorporating the new senor readings in updating the occupancy state of each cell that lies in the sensor field of view through the occupancy grid mapping algorithm as shown in Algorithm 3.1. For those cells that do not fall within the sensor cone, the occupancy probability remains unchanged, keeping the last obtained value $l_{t-1,ij}$. In the following two sections, 3.2 and 3.3, we provide further explanation of the terms involved in equation 3.15.

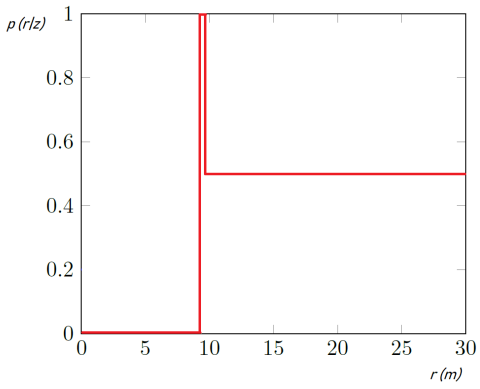
Algorithm 3.1 Occupancy Grid Mapping algorithm

1: **Algorithm occupancy_grid_mapping ({**$l_{t-1,i}$**}**, x_t**,** z_t**):**
2: **for** `all cells` m_i **do**
3: **if** m_i `in perceptual field of` z_t **then**
4: $l_{t,i} = l_{t-1,i}$ + **inverse_sensor_model(**m_i**,** x_t**,** z_t**)** - l_0
5: **else**
6: $l_{t,i} = l_{t-1,i}$
7: **end if**
8: **end for**
9: return $\{l_{t,i}\}$

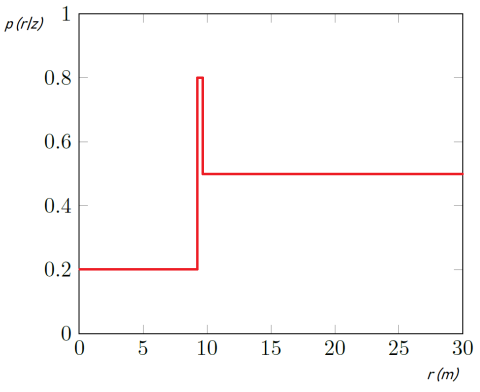
3.2 Sensor Modeling

In equation 3.15, the occupancy value of each cell within the sensor cone is updated by adding the term $l(m_{ij}|z_t, x_t)$. This term is responsible for incorporating the new sensor measurements by calculating the probability of the cell occupancy given only the current measurement z_t and the current vehicle pose x_t. This probability is often called the *inverse sensor model* as it interprets the causes into effects; provides information about the map given a sensor reading caused by an object in this map. As previously discussed in section 2.3, Lidars provide high angular resolution readings. Thus, its angular uncertainty is always neglected and only the uncertainty within the range readings is taken into consideration. That in turn results in Lidars having a 1D inverse sensor model in the range dimension. Figure 3.1a shows a Lidar with an ideal inverse sensor model that complies with the probability conventions stated at the start of this chapter:

- The region before the detection of the object is *free* region; occupancy probability = 0
- The region exactly where the detection happens, where the object is placed, is *occupied* region; occupancy probability = 1
- The occupancy state of the region beyond the detection is *unknown*; occupancy probability = .5



(a) Ideal inverse sensor model.



(b) Ideal inverse sensor model with limited probability.

Figure 3.1: Inverse sensor model for a Lidar detecting an object placed at range = 10 m.

In order for the upcoming Lidar measurements to be able to contribute to the update of the occupancy state of the cell, the probability of occupancy should be limited such that it does not reach neither 0 nor 1. Through this project, the maximum probability of occupancy P_{max} is chosen to be 0.8 whereas the minimum probability P_{min} is chosen to be 0.2. Thus, we can describe the modified ideal inverse sensor model as:

$$P(r|z) = \begin{cases} P_{min} & : 0 \leq r \leq z - \frac{\Delta r}{2} \\ P_{max} & : z - \frac{\Delta r}{2} \leq r \leq z + \frac{\Delta r}{2} \\ 0.5 & : r > z + \frac{\Delta r}{2} \end{cases} \tag{3.16}$$

Where z is the actual measurement obtained from the Lidar and r is the set of all possible values of the range measurement. Furthermore, we account for the noise that Lidars experience in reality by convolving the ideal sensor model with a Gaussian distribution with a zero mean and a variance value that simulates the sensor noise. In that way, we obtain a more accurate sensor model as shown in Figure 3.2 and which can be approximated as:

$$P(r|z) = \begin{cases} 0.2 & : r < z - \frac{\Delta r}{2} \\ \eta \mathcal{N}(r, z, \sigma_z) & : r \in [z - \frac{\Delta r}{2}, z] \\ 0.5 & : r > z \end{cases} \tag{3.17}$$

where:

$$\mathcal{N}(r, z, \sigma_z) = \frac{1}{\sqrt{2\pi}\sigma_z} . exp(-\frac{(r-z)^2}{2\sigma_z^2}) \tag{3.18}$$

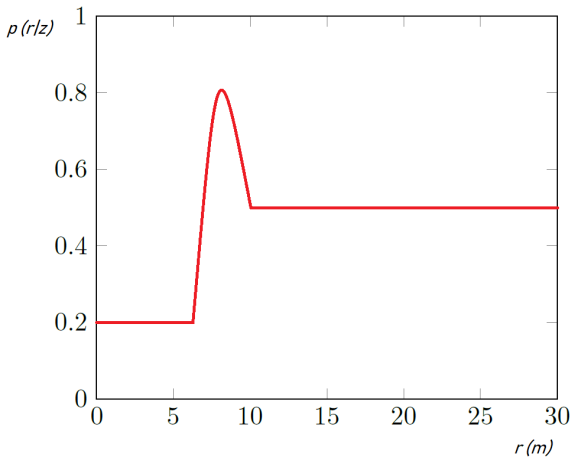


Figure 3.2: Gaussian inverse sensor model for a Lidar detecting an object placed at range = 10 m.

3.3 Map Initialization

In equation 3.15, each occupancy state update iteration involves the subtraction of the term $l(m_{ij})$. This term represents the prior probability, in log odds form, of the occupancy state of the cell. Since we usually have no prior knowledge about the map, each cell is assumed to have an unknown occupancy state at the beginning. Thus we initialize all the cells in the grid with occupancy probability equals .5.

$$p(m_{ij}) = .5 \qquad \forall i, j \tag{3.19}$$

3.4 Rao-Blackwellized Grid-Based SLAM

In the occupancy grid mapping algorithm shown in Algorthm 3.1, known and accurate positions of the vehicle are assumed to be given in order to obtain accurate representation of the vehicle surroundings. However, in real world, there is a large error within the raw odometry measurements that yields to inaccurate information about the vehicle positions in the environment. Feeding such noisy measurements into the grid mapping algorithm results in building inconsistent maps of the environment. Therefore, the raw odometry error cannot be neglected and in order to account for it, the problem shall be extended from predicting the posterior over only the map $p(m|z_{1:t}, x_{1:t})$ to predicting the posterior over both the map and the trajectory of the vehicle, defining the SLAM problem as:

$$p(x_{1:t}, m|z_{1:t}, u_{0:t}) \tag{3.20}$$

The main idea behind Rao-Blackwellized Grid-Based SLAM is to make use of both Monte Carlo Localization (MCL) and occupancy grid mapping to solve the problem. It divides the estimation of the joint posterior into first estimating the vehicle trajectory which is then used to estimate the map:

$$p(x_{1:t}, m|z_{1:t}, u_{0:t}) = p(x_{1:t}|z_{1:t}, u_{0:t}).p(m|z_{1:t}, x_{1:t}) \tag{3.21}$$

The Rao-Blackwellized Grid-Based SlAM samples the posterior of the trajectory by a number of particles where each particle represents a potential vehicle trajectory and is associated with building a particular map, based on the represented trajectory $x_{1:t}$ along with the observations $z_{1:t}$. The Rao-Blackwellized Grid-Based SlAM algorithm used and implemented through this project to solve the SLAM problem is shown in Algorithm 3.2 and can be summarized by the following steps:

1. **Sampling**: The new generation of particles $\{x_t\}$ is obtained by sampling from a proposal distribution $\pi(x_t|z_{1:t}, u_{0:t})$ which in most cases is chosen to be the probabilistic motion model distribution that accounts for the odometry sensor noise. Thus, in other words, in this step we obtain the new generation of particles $\{x_t\}$ by applying the motion model $p(x_t|x_{t-1}, u_{t-1})$ on the last generation $\{x_{t-1}\}$.

2. **Weighting**: In order to account for the difference between the proposal distribution and the true, or *target*, distribution of the vehicle trajectory, we use the latest scan readings received by the Lidar to calculate an importance weight w_t for each particle:

$$w_t = \frac{p(x_t|z_{1:t}, u_{0:t})}{\pi(x_t|z_{1:t}, u_{0:t})} \propto p(z_t|m, x_t) \tag{3.22}$$

   Mathematically, in this step, each particle is weighted based on the likelihood of the current observation in the maximum likelihood map; the map built so far using the vehicle trajectory proposed by this particular particle. In order to compute such observation likelihood, a scan-to-map registration routine is implemented through this project. For each range reading within the current scan batch received by the Lidar, we find the closest corresponding occupied cell in the map built so far. The larger the distance between the scan reading and its corresponding occupied cell, the lower the likelihood this scan reading has.

3. **Mapping**: The map estimate $p(m|z_{1:t}, x_{1:t})$ is computed for each particle given the vehicle trajectory, $x_{1:t}$, represented by this particle along with the observations, $z_{1:t}$, from the Lidar sensor.

4. **Re-sampling**: Due to the limited number of particles, particles with low weights are replaced by ones with high weights.

Algorithm 3.2 Grid-based RBPF SLAM (X_{t-1}, u_t, z_t)

$X'_t = X_t = \emptyset$
for k = 1 to M **do**
  $x_t^{[k]}$ = **sample_motion_model** (u_t, $x_{t-1}^{[k]}$)
  $w_t^{[k]}$ = **measurement_model** (z_t, $x_t^{[k]}$, $m_{t-1}^{[k]}$)
  $m_t^{[k]}$ = **update_occupancy_grid** (z_t, $x_t^{[k]}$, $m_{t-1}^{[k]}$)
  $X'_t = X'_t + \langle x_t^{[k]}, w_t^{[k]}, m_t^{[k]} \rangle$
end for
for k=1 to M **do**
  draw i with probability $\propto w_t^{[i]}$
  add $\langle x_t^{[i]}, m_t^{[i]} \rangle$ to X_t
end for
return X_t

3.5 Mapping with Multiple Lidars

As previously mentioned, Lidar sensor is one of the most suitable sensors to be used while solving the SLAM problem due to its high angular resolution and reliable readings. However, a Lidar sensor can suffer from data corruption under tough weather conditions. Using such corrupted measurements for estimating the map leads, in turn, to corrupted representation of the environment. Thus, it is with a great advantage to equip the truck with multiple Lidars such that a more accurate representation is feasible by integrating information from the different Lidars.

Here, an interesting question to ask is how to best integrate data from the different Lidars. As can be seen by Figure 3.3, there are two main approaches to incorporate the different Lidars measurements. The first approach is to build a sensor model for each Lidar, maintain an occupancy grid out of each sensor readings individually and further fuse those separate grids into one single grid; *Grid Fusion*. An alternative approach is to integrate the raw readings from the different Lidars into one point cloud, one sensor model, that is further processed through the SLAM algorithm and used to build the map; *Raw Measurement Fusion*.

In order to argue for the choice of *Raw Measurement Fusion* as the fusion method used through this project, we will first discuss the most common methods within the *Grid Fusion* approach and discuss why they are not suitable to be followed in comparison to the *Raw Measurement Fusion*.

(a) Grid Fusion

(b) Fusing raw Lidar data

Figure 3.3: Multi-Lidar data fusion techniques.

3.5.1 Grid Fusion

Let us assume S number of sensors, the posterior over a cell m_{ij} in the map generated from the s^{th} sensor readings is:

$$p_s(m_{ij}) = p(m_{ij}|z^s_{1:t}, x_{1:t}) \qquad 1 \leq s \leq S \tag{3.23}$$

Fusing the single grids from all the S sensors into one single grid, the posterior over the cell in the fused grid is defined as:

$$p(m_{ij}) = p(m_{ij}|z^1_{1:t}, .., z^s_{1:t}, x_{1:t}) \qquad 1 \leq s \leq S \tag{3.24}$$

The most common approaches to be applied while performing sensor fusion are:

- **Linear Opinion Pool** (LOP): An average of the probabilities provided by the individual sensors is performed while assigning a positive weight w_s to each sensor, where $\Sigma_s w_s = 1$, to determine the contribution of each sensor.

$$p(m_{ij}) = \Sigma_s w_s p_s(m_{ij}) \tag{3.25}$$

- **Independent Opinion Pool** (IOP): Based on the assumption that the sensors are independent, the IOP finds the geometric mean of the individual probabilities provided by each sensor.

$$p(m_{ij}) = K \prod_s p_s(m_{ij}) \tag{3.26}$$

  where K is the normalizing constant. The **Logarithmic Independent Opinion Pool** (LIOP) is a frequently generalized form of the IOP where a positive weight w_s is assigned to each sensor; $p(m_{ij}) = K \prod_s p_s(m_{ij})^{w_s}$

There are a lot of measures proposed in the literature to evaluate the performance of the fusion method. From our point of view, the absolute criteria to select the most suitable fusion technique can be described by the following properties:

- **Mitigation in Conflicting Situations**: When the measurements provided by the sensors are indicating contradictory information about the occupancy state of a cell, the property of mitigation is the ability of the fusion system to increase the uncertainty about the occupancy state of this cell. For instance, if one sensor perceives a cell as most likely to be occupied and another sensor perceives the same cell as most likely to be empty, the fusion output should make the probability of occupancy of this cell tend to .5 (unknown).

- **Reinforcement in Supporting Situations**: When the measurements provided by the sensors are indicating similar information about the occupancy state of a cell, the property of reinforcement is the ability of the fusion system to increase its belief about the occupancy state of this cell. For instance, when two sensors perceive the same cell as most likely to be occupied, the fusion output should make the probability of occupancy of this cell tend to 1 (occupied). Similarly, it should make the probability tend to 0 (empty) when both sensors perceive the cell as free.

- **Neglecting Non-Informative Information**: The final belief of the fusion system about the occupancy state of the cell should not be affected by non-informative sensors, reporting occupancy probability .5 (unknown state), in comparison to the informative ones.

Evaluating the LOP and IOP grid fusion methods against the aforementioned properties, the IOP, by calculating the geometric average of the individual probabilities, allows for reinforcement in the supporting information cases and totally neglects the non-informative sensors. However, the IOP is too extreme in its conclusion in the conflicting information situation; disabling the mitigation characteristic. To illustrate that, let us consider the case of three sensors reporting occupancy probabilities about a particular cell as $p_s(m_{ij}) = .9,.9,.9$ respectively. Applying the IOP formula in equation 3.26, the output probability is $p(m_{ij}) = .999$, concluding that the cell is high likely to be occupied. Similarly, when the probabilities reported by the three sensors are $p_s(m_{ij}) = .9,.5,.5$ respectively, the output probability is $p(m_{ij}) = .9$, neglecting the sensors reporting unknown state. when the reported probabilities by the sensors are $p_s(m_{ij}) = .9,.1,.1$ respectively, while an increase of the uncertainty is desirable in such case, the output probability is $p(m_{ij}) = .1$.

On the other hand, the LOP, by calculating the weighted average of the individual probabilities, is always conservative in its estimate; disabling the reinforcement characteristic in the supporting information cases and enabling the non-informative sensors to contribute to the final belief. To explain that, let us assume two sensors with equal weights reporting occupancy probabilities of $p_s(m_{ij}) = .9,.9$ respectively about a particular cell. While an increase in the belief about the occupancy state of this cell is desirable, applying LOP formula in equation 3.25 leads to output probability of $p(m_{ij}) = .9$. In another case when the sensors report probabilities of $p_s(m_{ij}) = .9,.5$ respectively, the output probability is $p(m_{ij}) = .7$; allowing the non-informative sensor to affect the final belief. To sum up, the LOP in most cases results in less informative grid than the ones being fused. Hence, we will rule out the LOP and decide the IOP as a more suitable grid fusion method for our application.

3.5.2 Raw Measurement Fusion

Since we are using a sensor setup that consists of multi-lidars to build an accurate occupancy grid map, we can take advantage of the similarity in the sensor models and the frequency between the Lidars by directly merging the data from the different Lidars into one grid as the simplest fusion method. The point cloud provided from each sensor is first transformed from the sensor coordinate frame to the vehicle coordinate frame; putting the different point clouds into a common frame. The fused cloud is further constructed by taking fixed angular steps where the readings from the different clouds are compared at each step and the closest reading is considered. This fused point cloud is finally used to build a single occupancy grid map.

Now let us analyse performing raw measurement fusion against IOP grid fusion. Firstly, in the raw measurement scale, sensors are only informative. At a specific angle, a sensor either provide a range measurement, corresponding to an occupied area, or not, corresponding to a free area. Secondly, although raw measurement fusion does not feature reinforcement in supportive information, it never, theoretically, leads to false negatives. Supportive sensors in law measurement level are sensors that provide the same range measurement at a specific angle and thus only one of them is considered in the fused cloud and processed through the sensor model to be interpreted as an occupancy probability. Finally, the main drawback of the raw measurement fusion is that it is always biased to false positives in conflicting information situations. A typical situation of such case would be a sensor reporting a range measurement at a specific angle while all other sensors do not for the

same angle. Hence, when constructing the fused point cloud, only the sensor providing a range measurement will be considered, in comparison to the other sensors. Nevertheless, such disadvantage can be redeemed by filtration of false measurements. Moreover, since all the Lidars have the same sensor model, it is unlikely that sensors provide conflicting information.

To conclude, an adequate occupancy representation of the vehicle surroundings is feasible through combining raw measurements from the different Lidars, in comparison to performing IOP fusion. Since performing independent opinion pool fusion is computationally expensive due to the fact that we have to maintain a map for each sensor before fusing them into one grid, combining raw measurements is the most suitable fusion method to our application where efficient and accurate maps are our objectives.

Chapter 4

Implementation

This chapter is dedicated to provide the reader with a better understanding of the project by presenting an overview of the developed system and addressing the issues associated with the implementation of the methodology described in the previous chapter.

4.1 Hardware Architecture

The project made use of Scania test truck which was equipped with an inertial measurement unit (IMU) sensor that was used to estimate the ego-vehicle motion and further update the position of the truck, an RTK GPS that was used for creating a ground truth about the vehicle position along with three Scala Lidar sensors mounted as shown in Figure 4.1 with the following specs:

Table 4.1: Lidar sensor specs

Parameter	value
Number of Vertical Layers	4
Horizontal field of view	145 °
Vertical field of view	$\pm$3.2 °
Angular resolution	.25 °
Range resolution	$\leq .1m$
Maximum range	$150m$
Data Refresh Time	40/80 ms

One of the main aims of this project is to make use of the computational capabilities of graphical processing units to handle the heavy processes

involved within both occupancy grid mapping and particle filter algorithms. Thus, an NVIDIA Jetson TX2 board that has a GPU, with 256 NVIDIA CUDA cores, was chosen as an embedded platform for running the implemented SLAM algorithm and evaluate the performance against a CPU platform.

(a) Side right Lidar (b) Front forward Lidar (c) Side left Lidar

Figure 4.1: Lidar sensors configuratin on the test truck.

4.2 Coordinate Frames Transformation

In this project, the perception of the vehicle's environment is achieved using a variety of sensors that are distributed in different spots on the truck. Thus, data received from the sensors are expressed in different coordinate frames, see Figure 4.2. Transformation between those different frames to a common reference frame, the vehicle coordinate frame; *base link*, is a fundamental step while processing the sensor data. Finding such transformation between each sensor frame and the vehicle frame is often referred as the *extrinsic calibration* of the sensor. Sensors calibration for the used test truck was previously done by Scania where an object was accurately positioned in relation to the vehicle and the transformation matrices were then defined by matching the sensors measurements to the actual position of the object. So, through the project, once the readings are received by the Lidars and the IMU, they are conveyed to the *base link* frame by applying the fixed predefined transformation matrices.

Moreover, in order to estimate the vehicle position while maintaining a map of its environment, the following coordinate frames transformations were maintained:

- **odom** to **base link**: The IMU measurements, after being transformed to the *base link* frame, are used to estimate the ego-vehicle motion between two time instants, t_1 and t_2, and further update the position of the truck relative to its initial position, *odom* frame.

- **map** to **odom**: This transformation corresponds to expressing the vehicle position in relation to the discretized map of its environment. Due to the error within the odometry information provided by the IMU, the vehicle position relative to the *map* frame can significantly drift over time. The localization component of the proposed SLAM algorithm accounts for such odometric errors and corrects the vehicle position in the map.

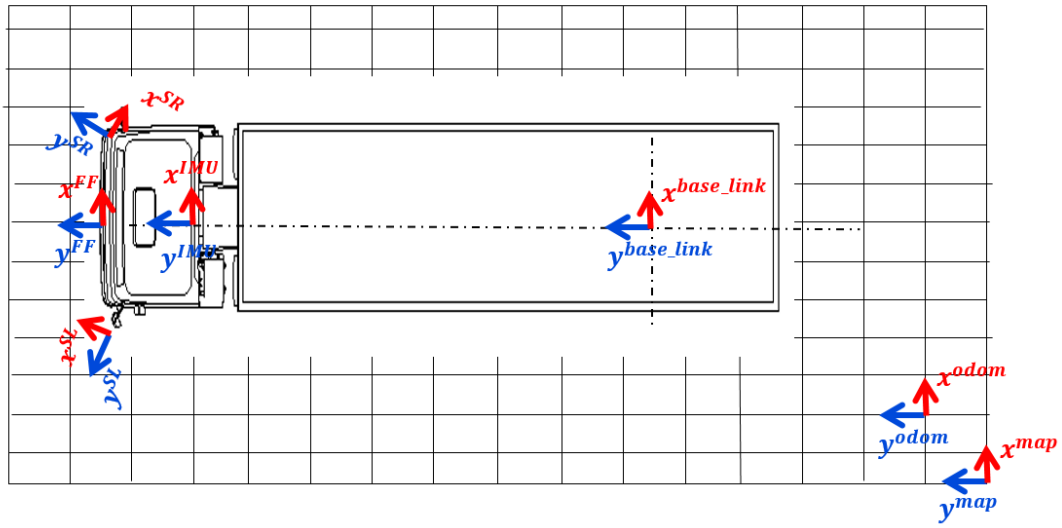


Figure 4.2: Coordinate frames conventions. *SR, FF,* and *SL*; side right, front forward and side left Lidars frames. *IMU*; inertial measurement unit frame. *odom*; the inertial odometric frame of the vehicle. *map*; the frame where the vehicle is moving.

4.3 Software Architecture

Figure 4.3 shows an overview of the software architecture developed through the project as an implementation of the chosen methodology.

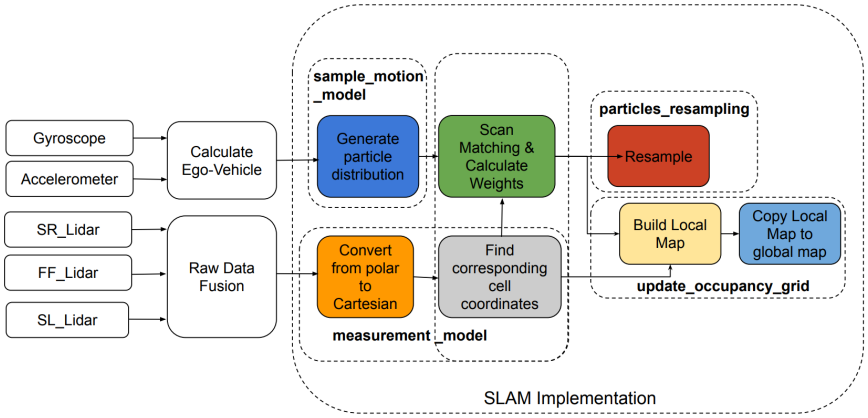


Figure 4.3: Software architecture overview.

We can sum up the workflow within the software architecture as follows:

1. Data are collected from all the sensors, Lidars and IMU, and transformed to the *base link* frame.

2. The information provided by the IMU are used to calculate the ego-vehicle motion and provide the odometry information for the SLAM algorithm.

3. The data from the different Lidars are fused into one point cloud that is further provided to the mapping component of the SLAM algorithm.

4. The Rao-Blackwellized Grid-Based SLAM algorithm shown previously in Algprithm 3.2 is applied on the provided scan and odometry data:

    - Once an odometry update happens, the motion model is applied and the new particles generation is obtained.
    - Once a new scan point cloud is received, the measurement model is applied for each particle; the readings are converted from polar to Cartesian, Bresenham's line algorithm is applied on each reading to determine the coordinates of the occupied cells, scan matching with the last obtained map is performed and finally the weight is calculated for this particle.

4.4 Practical Aspects

The different components of the software architecture were written in C++ programming language using Qt Creator toolkit. The system was implemented on the Kinetic distribution of the Robot Operating System (ROS) and run on Ubuntu 16.04 LTS. The data from the different Lidars were represented by synchronized Point Cloud ROS messages format and recorded in rosbag format. Moreover, the project made use of the ROS transformation TF package to maintain the relationship between the different coordinate frames.

Using 1000 particles and a grid resolution of .1 m, the building blocks of the SLAM algorithm were implemented as 7 kernels GPU, namely:

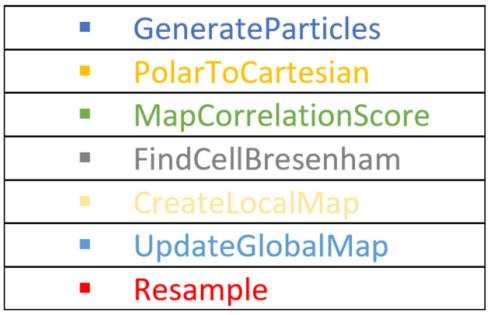


Figure 4.4: Implemented kernels on the GPU.

The same building blocks were implemented as functions, with the same naming criteria in C++ on ROS, Robotics Operating Systems, environment on the CPU. Finally, a comparison of the performance in terms of the execution time was constructed between the GPU and the CPU which is further shown and discussed in the next section.

Chapter 5

Evaluation and Results

The evaluation process through this project has been applied in two main steps. First, in order to validate the implemented Rao-Blackwellized particle filter SLAM algorithm, a quantitative analysis was performed using the SLAM benchmark described and discussed in section 2.6. Second, a set of experiments was designed and performed where the performance of the proposed system was qualitatively assessed. In this section, we thoroughly describe the evaluation procedure while presenting and discussing the results.

5.1 RBPF SLAM Benchmarking

The authors of [38] managed to create an objective benchmark by selecting datasets that represent the various types of indoor environments among the most frequently used datasets in the SLAM community, namely, MIT Killian Court and ACES building at the University of Texas datasets as an example of a topologically challenging corridor-environment with nested loops, and Building 079 at the University of Freiburg, Intel Research Lab and CSAIL at MIT datasets as examples of a cluttered office environment. They also addressed outdoor environments by recording new dataset at the University Hospital in Friburg. Moreover, Kümmerle *et al.* manually extracted relative displacements between the nearby poses for each dataset.

Further, they processed the aforementioned datasets through three common mapping techniques including scan-matching and Rao-Blackwellized particle filter SLAM. Finally, the absolute and squared errors out of each technique was computed according to the proposed metric in equation 2.1. The translational components of the errors for each dataset when processed through scan-matching and RBPF are presented in the first and second columns of table 5.1 respectively. While the rotational components are

presented in the first two columns of table 5.2. The datasets, reference relative relations, map images and the error metric evaluator source files can be found online at: *http://ais.informatik.uni-freiburg.de/slamevaluation/*

Processing the mentioned datasets through our implementation of RBPF SLAM algorithm and using the provided metric along with the reference relations, the absolute and squared errors have been calculated and registered in the third columns of tables 5.1 and 5.2. Thus, we can validate the performance of our implemented RBPF algorithm by comparing the computed errors for each dataset against the errors obtained from the scan-matching technique and RBPF used in [38]. Here, we used 50 particles such that a fair comparison to the results obtained in [38] can be held. In Figure 5.1, we provide a graphical analysis of comparing the translational errors of the three alorithms, while Figure 5.2 presents a graphical analysis of the rotational ones.

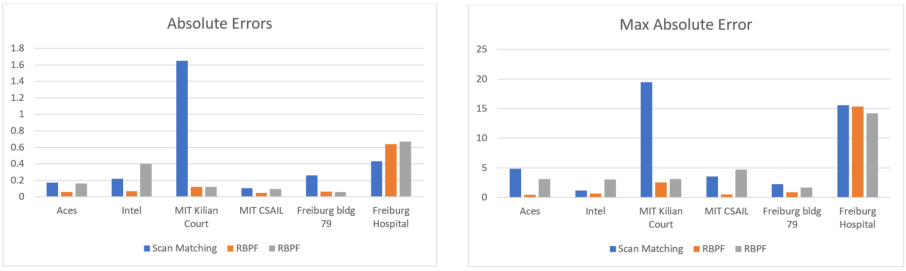
Based on the error scores in the tables and the error plots, we can see that the implemented RBPF in our project, like the one used in [38], outperforms the scan matching approach in most cases. This has to do with the fact that scan matching introduces high errors at loop closures. Moreover, we can see that the results of the RBPF implemented within our system are always comparable to the ones of the RBPF implementation used by Kümmerle *et al.*. Since the later led, on average, to building consistent maps when processing the different datasets, we can then conclude that our implementation has been validated to result in accurate representation of the environment. However, we can argue that the difference in performance between the two implementations is due to the fact that the RBPF used in [38] applies an improved motion model by using scan matching along with the odometry information and uses selective re-sampling technique. That in turn proves the positive effects of the improvements proposed in [14] and [15] on the performance of the Rao-Blackwellized Particle Filter algorithm.

Translational error m (abs) / m^2 (sqr)	Scan Matching [38]	RBPF [38]	RBPF
	Aces		
Absolute error	0.173 ± 0.614	0.060 ± 0.049	0.16 ± 0.37
Squared error	0.407 ± 2.726	0.006 ± 0.011	0.16 ± 0.83
Maximum absolute error	4.869	0.433	3.119
	Intel		
Absolute error	0.220 ± 0.269	0.070 ± 0.083	0.40 ± 0.63
Squared error	0.136 ± 2.277	0.011 ± 0.034	0.56 ± 1.45
Maximum absolute error	1.168	0.698	3.07
	MIT Kilian Court		
Absolute error	1.651 ± 4.138	0.122 ± 0.386	0.12 ± 0.35
Squared error	19.85 ± 59.84	0.164 ± 0.814	0.13 ± 0.78
Maximum absolute error	19.467	2.513	3.128
	MIT CSAIL		
Absolute error	0.106 ± 0.325	0.049 ± 0.049	0.093 ± 0.33
Squared error	0.117 ± 0.728	0.005 ± 0.013	0.17 ± 0.49
Maximum absolute error	3.570	0.508	4.72
	Freiburg bldg 79		
Absolute error	0.258 ± 0.427	0.061 ± 0.044	0.057 ± 0.027
Squared error	0.249 ± 0.687	0.006 ± 0.020	0.004 ± 0.05
Maximum absolute error	2.280	0.856	1.69
	Freiburg Hospital		
Absolute error	0.434 ± 1.615	0.637 ± 2.638	0.67 ± 3.05
Squared error	2.79 ± 18.19	7.367 ± 38.496	2.76 ± 23.52
Maximum absolute error	15.584	15.343	14.2109

Table 5.1: Translational components of the average and squared errors resultant from processing the different datasets through the three mapping algorithms along with the corresponding standard deviation and the maximum error. The errors out of our implemented RBPF are presented in the right column while the errors in the left and middle columns were obtained by Kümmerle *et al.* in [38].

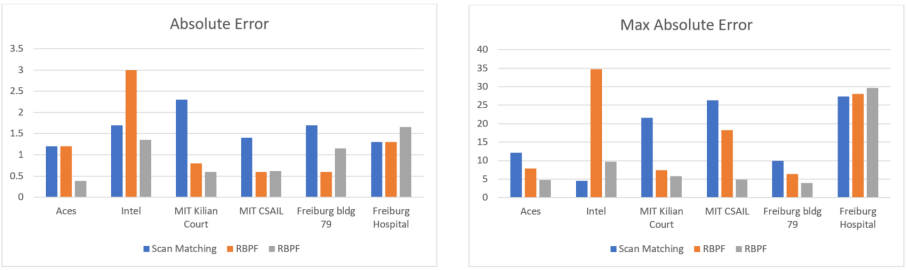
Rotational error deg (abs) / deg^2 (sqr)	Scan Matching [38]	RBPF [38]	RBPF
	Aces		
Absolute error	1.2 ± 1.5	1.2 ± 1.3	0.39 ± 0.43
Squared error	3.7 ± 10.7	3.1 ± 7.0	0.34 ± 1.49
Maximum absolute error	12.1	7.9	4.8
	Intel		
Absolute error	1.7 ± 4.8	3.0 ± 5.3	1.35 ± 2.28
Squared error	25.8 ± 170.9	36.7 ± 187.7	14.35 ± 160.0
Maximum absolute error	4.5	34.7	9.75
	MIT Kilian Court		
Absolute error	2.3 ± 4.5	0.8 ± 0.8	0.60 ± 0.41
Squared error	25.4 ± 65.0	0.9 ± 1.7	0.54 ± 1.66
Maximum absolute error	21.6	7.4	5.838
	MIT CSAIL		
Absolute error	1.4 ± 4.5	0.6 ± 1.2	0.62 ± 0.65
Squared error	22.3 ± 111.3	1.9 ± 17.3	0.8 ± 2.29
Maximum absolute error	26.3	18.2	4.85
	Freiburg bldg 79		
Absolute error	1.7 ± 2.1	0.6 ± 0.6	1.15 ± 0.99
Squared error	7.3 ± 14.5	0.7 ± 2.0	2.23 ± 3.34
Maximum absolute error	9.9	6.4	3.94
	Freiburg Hospital		
Absolute error	1.3 ± 3.0	1.3 ± 2.3	$1.66 \pm .66$
Squared error	10.9 ± 50.4	7.1 ± 42.2	3.19 ± 44.71
Maximum absolute error	27.4	28.0	29.7

Table 5.2: Rotational components of the average and squared errors resultant from processing the different datasets through the three mapping algorithms along with the corresponding standard deviation and the maximum error. The errors out of our implemented RBPF are presented in the right column while the errors in the left and middle columns were obtained by Kümmerle *et al.* in [38].



(a) Average absolute error (b) Maximum absolute error

Figure 5.1: Graphical analysis of the translational errors of the different mapping approaches. The errors of our implemented RBPF are shown in grey, while the errors of the RBPF used in [38] are presented in orange.



(a) Average absolute error (b) Maximum absolute error

Figure 5.2: Graphical analysis of the rotational error of the different mapping approaches. The errors of our implemented RBPF are shown in grey, while the errors of the RBPF used in [38] are presented in orange.

5.2 Experimental Evaluation

5.2.1 Experiment I

Environment Setup
The experimental evaluation was carried out at Scania test track where the test truck described previously in section 4.1 was stopped at some point in the middle of the road. Three objects were placed in front of the vehicle, in a triangular shape, that were formed using six moving boxes; each two boxes were put on top of each other acting as a large enough object to be detected by the Lidars. Behind the scene, another truck was parking beyond the left box with a spare object placed in front of it. Moreover, part of the experimental team were standing in two groups to the left side of the test truck.

Ground Truth
In order to reach a rough quantitative measure that support the qualitative analysis applied while evaluating the accuracy of the occupancy grids generated by the proposed algorithm, the left object was placed on an x-sign which was accurately positioned on the test track by Scania and the other two objects were delicately placed relative to the left one as shown in Figure 5.3.

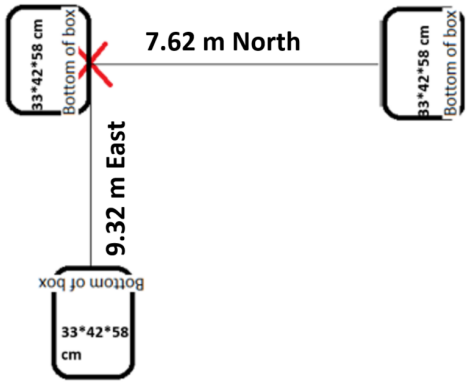


Figure 5.3: True relative distances between the objects.

5.2.2 Experiment I - Test Scenario 1

Objectives
This test case was designed with the sake of achieving the following aims:

- Verify the detection capabilities of each Lidar sensor.

- Evaluate the performance of the occupancy grid mapping component of the proposed SLAM system.
- Validate the ability of the implemented mapping module to handle the dynamic changes in the environment.

Description

While the vehicle is being static, the data from all the sensors were being logged for almost 1 minute. During recording the data, one person from the two standing groups moved toward the test vehicle and went back to the other group; causing referenced dynamic changes within the static scene. The collected logs were further processed offline; the data from each Lidar were passed individually to the mapping algorithm and an occupancy grid with .1 *m* cell resolution was maintained for each of the sensors .

Results and Discussion

While Figure 5.4 shows an online simulation of the detection from the different Lidars, the occupancy grid obtained from each Lidar readings can be seen in Figure 5.5.

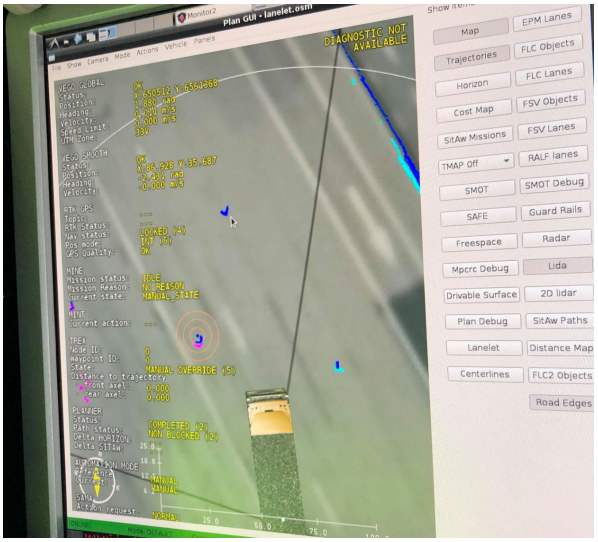
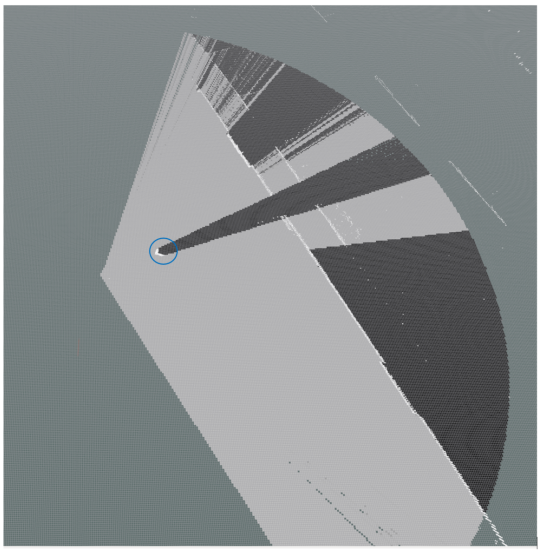


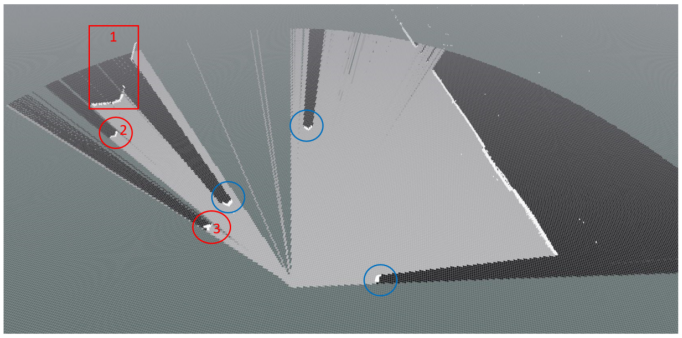
Figure 5.4: Online simulation of the Lidars detection. *Light blue*; side right Lidar, *dark blue*; front forward Lidar, *pink*; side left Lidar.

As can be seen in Figure 5.4, while the front forward Lidar has detected the three objects, each of the right and left Lidars has detected only one; right and left objects respectively. The objects were correctly mapped in the occupancy

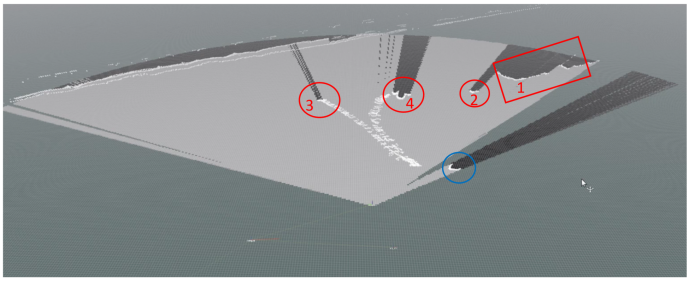
grid of the corresponding detecting Lidar; marked in blue in Figures 5.5a, 5.5b and 5.5c. The recorded log showed packet loss of 22.4 % in the front forward Lidar measurements which explains the huge missing part of the occupancy map generated from the front forward Lidar shown in Figure 5.5b. Nevertheless, we can see that not only the three objects are mapped, but also the truck behind the scene along with the spare object; referenced by numbers one and two respectively in the figure. We can extrapolate the overlap in the field of view between the front forward and side left Lidars by the left object, the parking truck and the spare object being mapped again in the occupancy grid of the side left Lidar shown in Figure 5.5c beside the two groups of people; referenced by numbers three and four in the figure. More importantly, we can see in Figure 5.5c that the dynamic movements were detected by the left Lidar but not interpreted into occupied cells; proving the ability of the algorithm to handle the dynamic changes while mapping a static environment.



(a) Occupancy grid of the **side right** Lidar readings.



(b) Occupancy grid of the **front forward** Lidar readings.

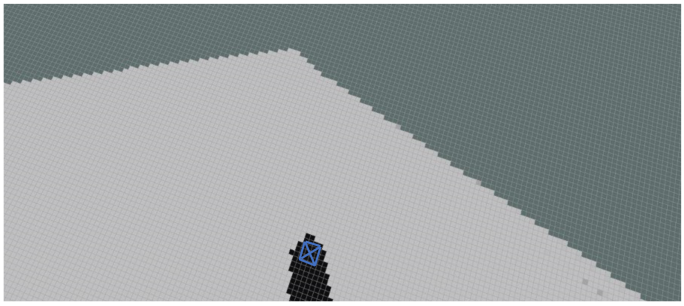


(c) Occupancy grid of the **side left** Lidar readings.

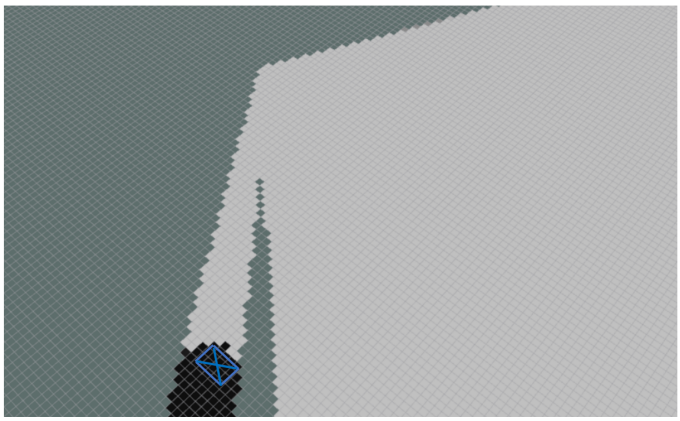
Figure 5.5: Resultant occupancy grids out of processing each Lidar point cloud individually through the mapping algorithm. Grids are visualized in each Lidar frame. In *blue* circles; the referenced objects. In *red* circles; the different objects behind the scene. The *white* points are the accumulated measurements of each Lidar.

The parts of the grids generated by the side right and side left Lidars readings where the objects have been mapped are highlighted in Figures 5.6a and 5.6b respectively, with the objects extracted manually. Although sharp occupied cells directly beneath the two dimensions of the object detected by each Lidar were expected, the figures show some occupied cells before the two dimensions. This has to do with the fact that the windy weather during the test along with the improper attachment between the two boxes forming the object have led to parts of the top box to be detected by the Lidar before the bottom one. However, neglecting this fact, for the simplicity of the evaluation, we can see that those excessive cells do not exceed two rows in width. Thus, we can conclude that the deviation between the ground truth and the mapping algorithm output is lower than .2 m.

The part in Figure 5.5b where the objects are mapped is zoomed in Figure 5.7 while the objects are manually extracted and the relative distances between the objects are measured. A comparison between those distances and the true ones is constructed in table 5.3.



(a) The **right object** mapped by **side right** Lidar readings.



(b) The **left objec** mapped by **side left** Lidar readings.

Figure 5.6: Extracted Objects from the occupancy grids.

Table 5.3: Relative distances between the objects in scenario 1

Distance	True value (m)	Measured value (m)	error (m)
Left box to Middle box	7.62	8.0	.38
Left box to Right box	9.32	9.5	.18

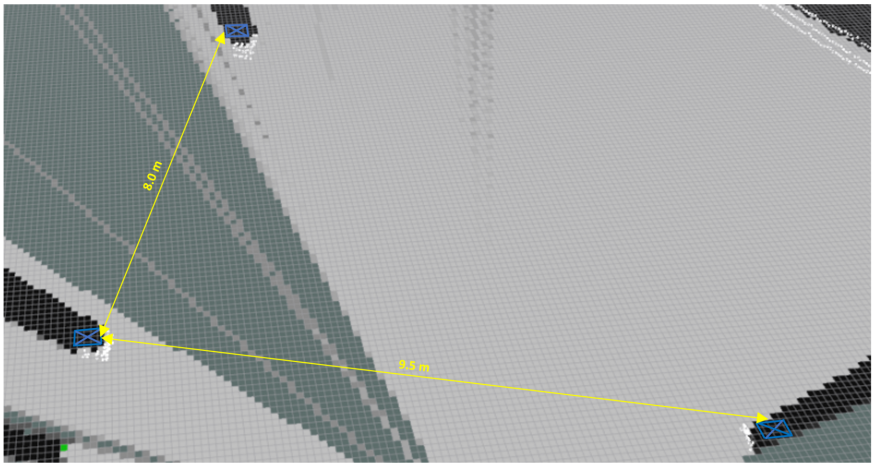


Figure 5.7: Extracted objects from the front forward Lidar occupancy grid.

5.2.3 Experiment I - Test Scenario 2

Objectives
The aim of this test scenario is to evaluate the performance of the mapping component of the proposed SLAM system while paying attention to the effect of the low-level sensor fusion process.

Description
While the vehicle being static, the side left Lidar was completely blinded and the data from all sensors were being logged for almost 1 minute. The collected log was further processed offline; the data from the different Lidars were first fused into one point cloud which was then passed to the mapping algorithm and a general occupancy grid with .1 *m* cell resolution was constructed.

Results and Discussion
Figure 5.8 shows the projected point cloud received from the different Lidars along with the point cloud resultant from the raw measurement fusion process. The side left Lidar does not provide any information about the environment; being totally blinded. It is clear that the fused point cloud, white points in the figure, perfectly combines the readings provided by the two Lidars; side right and front forward. The occupancy grid map built out of processing the fused point cloud is presented in Figure 5.9. Again, the missing parts of the grid corresponds to the large packet loss in the front forward Lidar measurements. However, we can see that the fused grid provide more information about the vehicle's surroundings compared to the grid constructed from the side right Lidar point cloud individually in the first scenario. Moreover, applying low-level fusion is a great boost to the SLAM system in case of sensor failure, simulated in our scenario by camouflaging the side left Lidar, given that the Lidars are complementing each other in terms of their field of view. Figure 5.10 magnifies the part of the grid where the objects are mapped. Extracting the objects on the map and measuring the relative distances between them, we can compare those distances to the ground truth as illustrated in table 5.4. We can notice that by applying the low-level fusion, the error has reduced compared to the case when only the front forward Lidar readings were used for mapping the environment.

Table 5.4: Relative distances between the objects in scenario 2

Distance	True value (m)	Measured value (m)	error (m)
Left box to Middle box	7.62	7.9	.28
Left box to Right box	9.32	9.2	.12

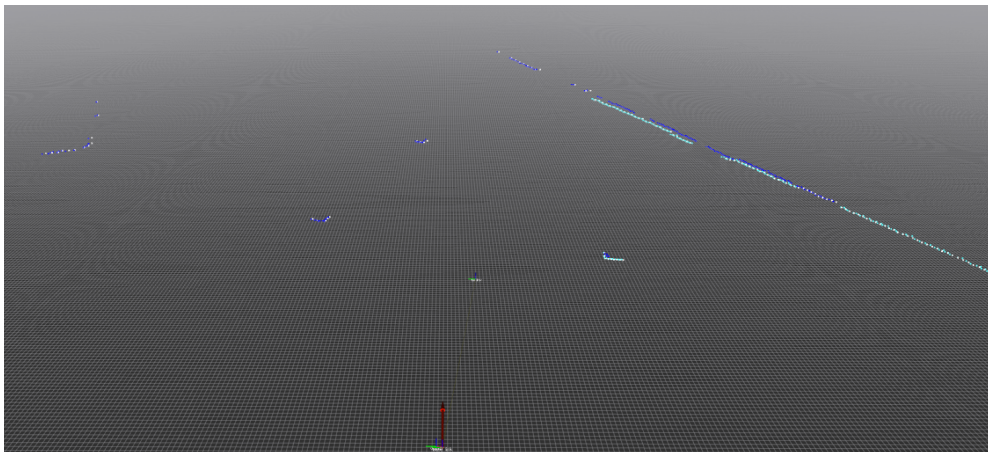


Figure 5.8: Accumulated Laser scans received from the Lidars; In *light blue*, side right laser scans and in *dark blue*, front forward laser scans. In *white*, the fused laser scans when applying low-level fusion.

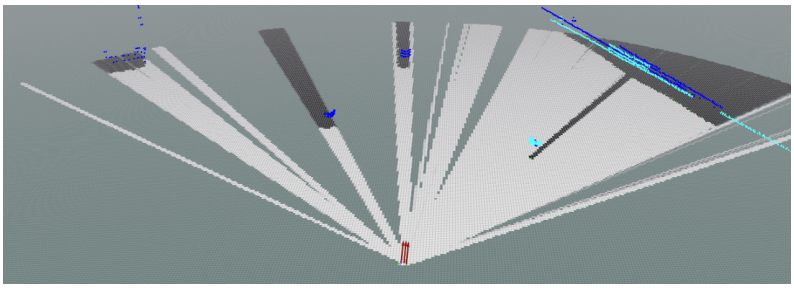


Figure 5.9: Occupancy grid map built out of the fused point cloud and visualized in the base link frame.

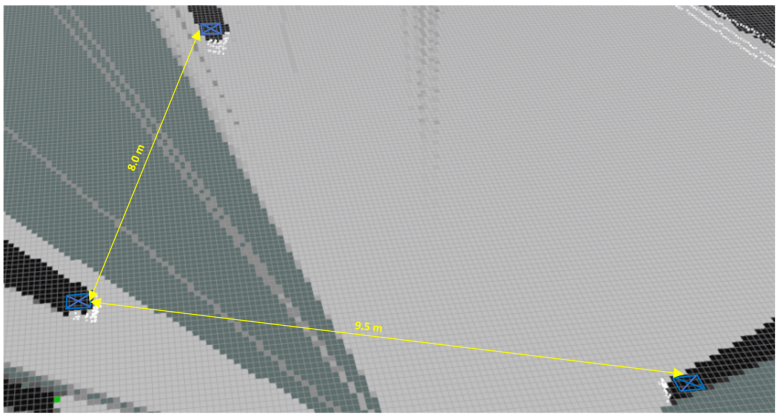


Figure 5.10: Extracted objects from the fused grid map.

5.2.4 Experiment II

Description

In this experiment, we evaluate the performance of the full proposed SLAM system; both localization and mapping. With the same environment setup as the first experiment, the truck was parked far from the objects where none of the objects was detected by the front forward Lidar. The truck started moving slowly from its initial position approaching the boxes while performing two stops on the way to make sure that there is enough time to map the boxes. While the truck was moving, the data from all the sensors; Lidars, IMU and RTK GPS were being logged. The collected log was further processed offline; the data from the different Lidars were first fused into one point cloud which was then passed to the SLAM algorithm and a general occupancy grid with .5 m cell resolution was maintained.

Results and Discussions

In Figure 5.11, we present the RBPF SLAM trajectory in comparison to the odometry information obtained from the IMU and the ground truth represented by the RTK GPS measurements. We can see that the SLAM output was struggling at the beginning where no detection was available for localization. However, the SLAM trajectory was further improved when objects were detected and mapped, resulting in the estimated positions being close to the ground truth ones. Moreover, it is obvious that the estimated trajectory by the RBPF SLAM, marked in yellow in the figure, is better than its corresponding obtained form the IMU readings, shown in red. That is proven as well by the SLAM results showing lower max and mean absolute errors compared to the IMU odometry, see table 5.5.

Table 5.5: Maximum and Mean absolute error

Estimation Approach	MAE (m)	Max error (m)
RBPF SLAM	0.6939	2.3471
Imu Odometry	2.3471	4.6782

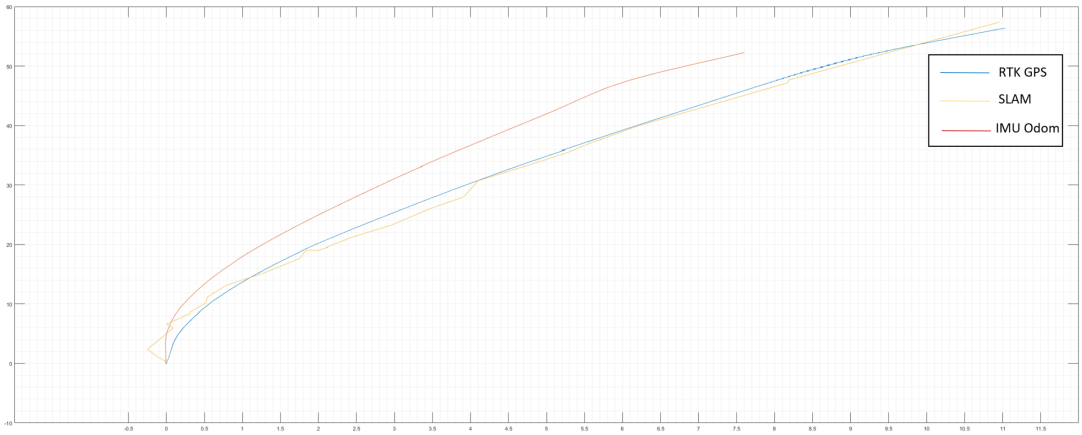


Figure 5.11: The different estimated trajectories of the truck during the experiment.

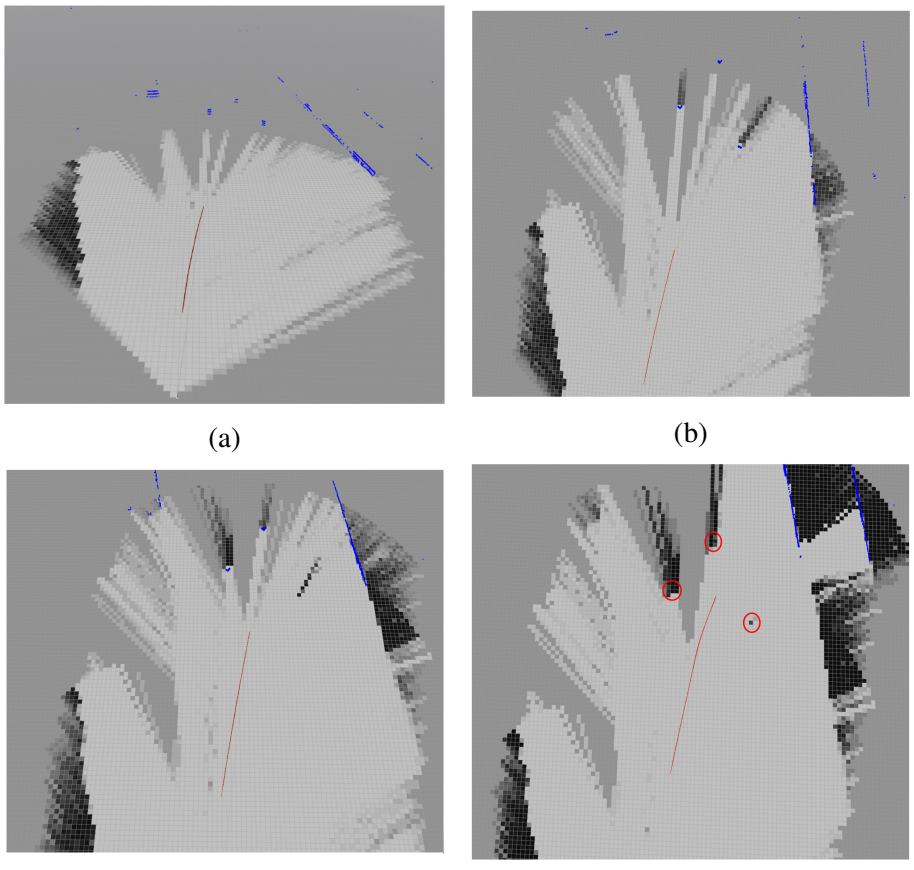


Figure 5.12: Maintaining an occupancy grid map using RBPF SLAM.

Figure 5.12 illustrates the process of maintaining an occupancy grid map of the environment while keeping track of the vehicle trajectory. In Figure 5.12-a, we can see that no object has been detected yet. While in Figure 5.12-b, the right and left objects have been detected and are being mapped. Afterwords, in Figure 5.12-c, while the middle object has been detected, the right one is not in the field of view of any of the Lidars anymore and the occupancy state of its surrounding area is being updated. The final obtained occupancy grid is presented in Figure 5.12-d where the objects are marked in red circles. The right object has been interpreted into one occupied cell which perfectly mirrors the area of the box. However, this is not the case for the middle and left objects as the truck stopped and did not pass the objects. Moreover, logging the data was directly terminated at this point which resulted in not enough time for updating the occupancy.

5.2.5 Computational Performance

In this section, we discuss the efficiency of the proposed SLAM system in terms of its computational performance. As mentioned in section 4.4, the building blocks of the SLAM algorithm were first implemented as 7 kernels on NVIDIA Jetson TX2 GPU where most parts of the algorithm have been parallelized. The same implementation was further adjusted and run on the CPU with 7 functions processing the same tasks as the kernels. The execution time of each building block on the CPU and the GPU is shown in Table 5.6. We notice a dramatic drop in the total execution time of the algorithm from 70.5 *ms* on the CPU to 22.816 *ms* on the GPU, see Figure 5.13. That in turn indicates that the proposed SLAM algorithm is 2.09 times faster compared to when it is running on the CPU.

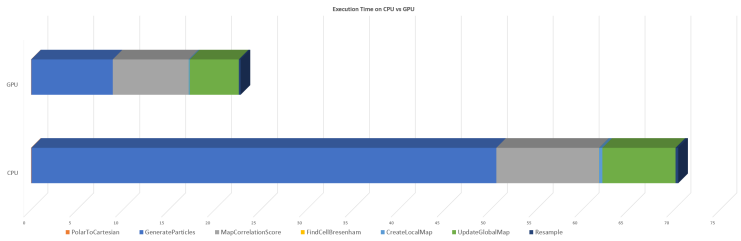


Figure 5.13: Total execution times.

Table 5.6: Execution Time of the different blocks

Kernel/Function	CPU (*ms*)	GPU (*ms*)
PolarToCartesian	0.042	0.028
GenerateParticles	50.61	8.87
MapCorrelationScore	11.155	8.243
FindCellBresenham	0.014	0.009
CreateLocalMap	0.378	0.128
UpdateGlobalMap	7.96	5.34
Resample	0.298	0.198

Chapter 6

Conclusion and Future Work

We finalize the report by summarizing the work that has been done within the implementation phase of this project, highlighting the results out of the evaluation process and proposing potential improvements in the form of future work.

6.1 General Conclusion

The Localization and Mapping module is one of the core competencies for designing vehicles with autonomy level higher than *level 1*. Due to the reduction in production costs of advanced sensory technologies, the autonomous driving research community has recently become interested in integrating SLAM, Simultaneous Localization and Mapping, solutions into commercial vehicles; increasing the intuitive awareness of the vehicles. The goal of this project was to propose a complete SLAM system that can be used by self-driving vehicles to efficiently map its surroundings while maintaining its location in the environment. Our approach to achieve such system was to utilize Bayesian inference theory to reach a detailed occupancy representation of the environment. The state-of-art Rao-Blackwellized particle filter was deployed with the grid representation of the environment to delicately estimate the location of the vehicle while mapping its surroundings.

The algorithm was efficiently implemented on Jetson TX2 board overcoming the complex computations associated with occupancy grid and particle filter frameworks. The implementation was validated by testing it on the most frequent used datasets in the SLAM community and benchmarking it against another mapping approach, scan-matching, and against one of the most popular grid-based RBPF implementations in the community, *GMapping*. The results out of our implementation not only outperformed the scan-

matching technique results but also was comparable to the results obtained from the *GMapping* RBPF implementation. To provide an insight of potential real time solution that could be achieved by adapting and/or improving our GPU implementation, a similar implementation was done on the CPU and a comparison in terms of the execution time was constructed between the two embedded platforms where the GPU implementation was found to be almost three times faster than its corresponding on the CPU.

While most of the research is done on scaled vehicles and in virtual environments, this project aimed to address the challenging SLAM problem within an industrial context and to provide a proof-of-concept of the feasibility of integrating SLAM solutions into commercial vehicles. With the Lidar being one of the most suitable sensors for accurately mapping the environment, we used Scania heavy truck that was equipped with multiple Lidars sensory architecture. An inverse sensor model was defined for the Lidars to update the occupancy state of the grid cells. Moreover, raw measurement fusion was performed to integrate the Lidars readings; compensating for each other drawbacks. Further, a set of real experiments was designed and performed at Scania test track where the performance of the full system has been evaluated in terms of the quality of the resulting occupancy map and the accuracy of the estimated trajectory of the truck compared to the ground truth.

In the different designed scenarios, the proposed system was able to build adequate occupancy maps of the truck surroundings. Moreover, in the second scenario of Experiment 1, the raw measurement fusion applied between the Lidars helped to skip the sensor synchronization step and was proven to result in more informative grid than the grid that each Lidar can individually provide. Furthermore, the localization component showed more accurate estimation of the truck trajectory compared to the one obtained from the raw odometry information. With the truck being able to map its static environment and localize itself in a a global reference frame, through applying state-of-art computer vision technologies, it can further plan it path, determine the drivable corridor and avoid collisions.

6.2 Discussion and Future Work

Due to the constraints on the available resources and timeline of the project, the functionality of the developed system within this research project was limited. Thus, we here discuss some of those limitations and propose directions for further research:

- **Mapping Dynamic Environments:** The mapping within this projects was limited to environments containing static objects due to the assumptions applied by the occupancy grid mapping framework. Extending the developed system to be able to detect and map the dynamic changes within the environment would be with a great assist for an autonomous vehicle. One approach to do so is to apply the Bayesian Occupancy Filter, BOF, introduced in [40]. In BOF, a four-dimensional grid is maintained where two dimensions hold the (x, y) coordinates of the cell and the other two dimensions represent the estimated velocity of the cell in x and y directions.

- **Raw Measurement and Grid Fusion:** As discussed in section 3.5, we fused the raw readings provided from the different Lidars and we argued for using such fusion technique against the grid fusion method. Although the results showed that the applied raw measurement fusion leads to more informative grid than the grids obtained from each Lidar individually, such fusion approach is limited and does not allow for including more sensors with different physical working principles than the Lidar. Whereas more sensors can be easily added within the grid fusion framework by maintaining an additional grid map out of the new sensor measurements and fusing it with the others, seeking such general scalable fusion module always leads to suboptimal utilization of the different sensor readings. Thus, an alternative direction can be to propose different sensory system architectures, find the most optimal fusion approach for the system in hand and compare the performance of the different systems under the same set of designed experiments.

- **Exploitation of the RBPF improvements proposed in [14] and [15]:** In section 5.1, the RBPF implementation used in [38] showed better performance against our implementation; proving the positive effects of the improvements proposed in [14] and [15] in terms of the resultant maps consistency. However, no detailed information about the execution time of such implementation are provided. An interesting point would be to parallelize the implementation using a GPU and evaluate its computational performance as well.

- **Different Objects:** Due to the limited available resources of the project, we had the chance to test the proposed system only on objects

formed from carton boxes. The evaluation process should be further extended to determine the Lidar detection capabilities with different types of objects and under different simulated weather conditions while paying attention to the performance of the fusion module within each scenario.

Bibliography

[1] SAE International. In: *Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems J3016*. 2014.

[2] Daniel Watzenig and Martin Horn. *Automated Driving: Safer and More Efficient Future Driving*. 1st. 2016.

[3] World Health Organization. In: *Global status report on road safety : time for action*. 2009. http://www.who.int/iris/handle/10665/44122.

[4] Jaagup Ainsalu et al. "State of the Art of Automated Buses". In: *Sustainability* 10.9 (Aug. 2018), pp. 1–34.

[5] Edmond Awad et al. "The Moral Machine Experiment". In: *Nature* 563 (Nov. 2018). DOI: 10.1038/s41586-018-0637-6.

[6] Kai M. Wurm, Cyrill Stachniss, and Giorgio Grisetti. "Bridging the gap between feature- and grid-based SLAM". In: *Robotics and Autonomous Systems* 58.2 (2010), pp. 140–148.

[7] Cesar Cadena et al. "Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age". In: *arXiv e-prints* (June 2016), arXiv:1606.05830.

[8] Christopher Weyers and Gilbert Peterson. "Improving occupancy grid FastSLAM by integrating navigation sensors". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2011), pp. 859–864.

[9] Karl Berntorp and Jerker Nordh. "Rao-Blackwellized Particle Smoothing for Occupancy-Grid Based SLAM Using Low-Cost Sensors". In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 10174–10181.

[10] Jose Luis Blanco, J.-A Fernández-Madrigal, and Javier González-Jiménez. "Toward a Unified Bayesian Approach to Hybrid Metric–Topological SLAM". In: *Robotics, IEEE Transactions on* 24 (May 2008), pp. 259–270.

[11] Arnaud Doucet et al. “Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks”. In: *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*. UAI ’00. 2000, pp. 176–183. ISBN: 1-55860-709-9.

[12] Michael Montemerlo et al. “FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem”. In: *In Proceedings of the AAAI National Conference on Artificial Intelligence*. AAAI, 2002, pp. 593–598.

[13] D Hahnel et al. “An Efficient FastSLAM Algorithm for Generating Maps of Large-Scale Cyclic Environments from Raw Laser Range Measurements”. In: vol. 1. Nov. 2003, 206–211 vol.1.

[14] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. “Improving Grid-based SLAM with Rao-Blackwellized Particle Filters By Adaptive Proposals and Selective Resampling”. In: Jan. 2005, pp. 2432–2437.

[15] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. “Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters”. In: *IEEE Transactions on Robotics* 23 (2007), pp. 34–46.

[16] Mubariz Zaffar et al. “Sensors, SLAM and Long-term Autonomy: A Review”. In: *arXiv e-prints* (July 2018), arXiv:1807.01605.

[17] Andres Diaz et al. “A real time 6DOF visual SLAM system using a monocular camera”. In: Oct. 2012, pp. 45–50. ISBN: 978-1-4673-4650-4.

[18] Thomas Lemaire et al. “Vision-Based SLAM: Stereo and Monocular Approaches”. In: *International Journal of Computer Vision* 74 (Sept. 2007), pp. 343–364.

[19] T.J. Chong et al. “Sensor Technologies and Simultaneous Localization and Mapping (SLAM)”. In: *Procedia Computer Science* 76 (2015), pp. 174–179.

[20] Julian Ryde and Nick Hillier. “Performance of laser and radar ranging devices in adverse environmental conditions”. In: *J. Field Robotics* 26 (2009), pp. 712–727.

[21] Johan Degerman, Thomas Pernstal, and Klas Alenljung. “3D occupancy grid mapping using statistical radar models”. In: *2016 IEEE Intelligent Vehicles Symposium (IV)* (2016), pp. 902–908.

[22] David Droeschel and Sven Behnke. “Efficient Continuous-Time SLAM for 3D Lidar-Based Online Mapping”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (2018), pp. 1–9.

[23] Ren C. Luo, Chih C. Chang, and Chun Chi Lai. "Multisensor Fusion and Integration: Theories, Applications, and its Perspectives". In: *IEEE Sensors Journal* 11 (2011), pp. 3122–3138.

[24] Bahador Khaleghi et al. "Multisensor data fusion: A review of the state-of-the-art". In: *Information Fusion* 14 (2013), pp. 28–44.

[25] R.C. Luo and Michael Kay. "A tutorial on multisensor integration and fusion". In: Dec. 1990, 707–722 vol.1.

[26] Jos Elfring et al. "Effective World Modeling: Multisensor Data Fusion Methodology for Automated Driving". In: *Sensors*. 2016.

[27] Alberto Elfes and Larry Matthies. "Sensor integration for robot navigation: Combining sonar and stereo range data in a grid-based representataion". In: Jan. 1988, pp. 1802–1807.

[28] Sebastian Thrun. "Learning Occupancy Grids With Forward Sensor Models". In: 2002.

[29] Edouard Ivanjko, Ivan Petrovic, and Misel Brezak. "Experimental Comparison of Sonar Based Occupancy Grid Mapping Methods". In: 50 (Dec. 2009), pp. 65–79.

[30] Wilfried Elmenreich. *An Introduction to Sensor Fusion*. Research Report 47/2001. Technische Universität Wien, Institut für Technische Informatik, 2001.

[31] Florian Homm, Nico Kaempchen, and Darius Burschka. "Fusion of laserscannner and video based lanemarking detection for robust lateral vehicle control and lane change maneuvers". In: *2011 IEEE Intelligent Vehicles Symposium (IV)* (2011), pp. 969–974.

[32] Igor Paromtchik, Mathias Perrollaz, and Christian Laugier. "Fusion of Telemetric and Visual Data from Road Scenes with a Lexus Experimental Platform". In: *IEEE International Symposium on Intelligent Vehicles*. June 2011.

[33] Ruben Garcia et al. "High level sensor data fusion for automotive applications using occupancy grids". In: *2008 10th International Conference on Control, Automation, Robotics and Vision* (2008), pp. 530–535.

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

[35] D. Rodriguez-Losada et al. "GPU-Mapping: Robotic Map Building with Graphical Multiprocessors". In: *Robotics Automation Magazine, IEEE* 20.2 (June 2013), pp. 40–51.

[36] Haiyang Zhang and Fred Martin. "CUDA accelerated robot localization and mapping". In: *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)* (2013), pp. 1–6.

[37] Manuel Yguel, Olivier Aycard, and Christian Laugier. "Efficient GPU-based construction of occupancy grids using several laser range-finders". In: 6 (Jan. 2008), pp. 48–83.

[38] Rainer Kümmerle et al. "On measuring the accuracy of SLAM algorithms". In: *Auton. Robots* 27 (2009), pp. 387–407.

[39] A. Elfes and L. Matthies. "Sensor integration for robot navigation: Combining sonar and stereo range data in a grid-based representataion". In: *26th IEEE Conference on Decision and Control*. Vol. 26. Dec. 1987, pp. 1802–1807.

[40] Christophe 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.

TRITA -EECS-EX-2019:540