SLAM and data fusion for autonomous vehicles : from classical approaches to deep learning methods

Michelle Andrade Valente da Silva

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à MINES ParisTech

SLAM and Data Fusion for Autonomous Vehicles

From classical approaches to deep learning methods

SLAM et fusion de données pour le véhicule autonome

Des approches classiques aux méthodes d'apprentissage en profondeur

Soutenue par

Michelle VALENTE

Le 18 decembre 2019

École doctorale n°621

Ingénierie des Systèmes, Matériaux, Mécanique, Énergétique

Spécialité

Informatique temps réel, robotique et automatique

Composition du jury :

Ching-Yao Chan UC Berkeley	*Rapporteur*
Vincent Fremont École Centrale de Nantes	*Rapporteur*
Emilie Wirbel Valeo	*Examinatrice*
David Filliat ENSTA	*Président du jury*
Cyril Joly MINES Paristech	*Examinateur*
Arnaud de La Fortelle MINES Paristech	*Directeur de thèse*

MINES ParisTech | PSL

Abstract

Self-driving cars have the potential to provoke a mobility transformation that will impact our everyday lives. They offer a novel mobility system that could provide more road safety, efficiency and accessibility to the users. In order to reach this goal, the vehicles need to perform autonomously three main tasks: perception, planning and control. When it comes to urban environments, perception becomes a challenging task that needs to be reliable for the safety of the driver and the others. It is extremely important to have a good understanding of the environment and its obstacles, along with a precise localization, so that the other tasks are well performed.

This thesis explores from classical approaches to Deep Learning techniques to perform mapping and localization for autonomous vehicles in urban environments. We focus on vehicles equipped with low-cost sensors with the goal to maintain a reasonable price for the future autonomous vehicles. Considering this, we use in the proposed methods sensors such as 2D laser scanners, cameras and standard IMUs.

In the first part, we introduce model-based methods using evidential occupancy grid maps. First, we present an approach to perform sensor fusion between a stereo camera and a 2D laser scanner to improve the perception of the environment. Moreover, we add an extra layer to the grid maps to set states to the detected obstacles. This state allows to track an obstacle over time and to determine if it is static or dynamic. Sequentially, we propose a localization system that uses this new layer along with classic image registration techniques to localize the vehicle while simultaneously creating the map of the environment.

In the second part, we focus on the use of Deep Learning techniques for the localization problem. First, fwe introduce a learning-based algorithm to provide odometry estimation using only 2D laser scanner data. This method shows the potential of neural networks to analyse this type of data for the estimation of the vehicle's displacement. Sequentially, we extend the previous method by fusing the 2D laser scanner with a camera in an end-to-end learning system. The addition of camera images increases the accuracy of the odometry estimation and proves that we can perform sensor fusion without any sensor modelling using neural networks. Finally, we present a new hybrid algorithm to perform the localization of a vehicle inside a previous mapped region. This algorithm takes the advantages of the use of evidential maps in dynamic environments along with the ability of neural networks to process images.

The results obtained in this thesis allowed us to better understand the challenges of vehicles equipped with low-cost sensors in dynamic environments. By adapting our methods for these sensors and performing the fusion of their information, we improved the general perception of the environment along with the localization of the vehicle. Moreover, our approaches allowed a possible comparison between the advantages and disadvantages of learning-based techniques compared to model-based ones. Finally, we proposed a form of combining these two types of approaches in a hybrid system that led to a more robust solution.

Résumé

L'arrivée des voitures autonomes va provoquer une transformation très importante de la mobilité urbaine telle que nous la connaissons, avec un impact significatif sur notre vie quotidienne. En effet, elles proposent un nouveau système de déplacement plus efficace, plus facilement accessible et avec une meilleure sécurité routière. Pour atteindre cet objectif, les véhicules autonomes doivent effectuer en toute sécurité et de manière autonome trois tâches principales: la perception, la planification et le contrôle.

L'objectif de cette thèse est d'explorer différentes techniques pour la cartographie et la localisation des voitures autonomes en milieu urbain, en partant des approches classiques jusqu'aux algorithmes d'apprentissage profond. On s'intéresse plus spécifiquement aux véhicules équipés de capteurs bon marché avec l'idée de maintenir un prix raisonnable pour les futures voitures autonomes. Dans cette optique, nous utilisons dans les méthodes proposées des capteurs comme des scanner laser 2D, des caméras et des centrales inertielles à bas coût.

Dans la première partie, nous introduisons des méthodes classiques utilisant des grilles d'occupation évidentielles. Dans un premier temps, nous présentons une nouvelle approche pour faire de la fusion entre une caméra et un scanner laser 2D pour améliorer la perception de l'environnement. De plus, nous avons ajouté une nouvelle couche dans notre grille d'occupation afin d'affecter un état à chaque objet détecté. Cet état permet de suivre l'objet et de déterminer s'il est statique ou dynamique. Ensuite, nous proposons une méthode de localisation s'appuyant sur cette nouvelle couche ainsi que sur des techniques de superposition d'images pour localiser le véhicule tout en créant une carte de l'environnement.

Dans la seconde partie, nous nous intéressons aux algorithmes d'apprentissage profond appliqués à la localisation. D'abord, nous introduisons une méthode d'apprentissage pour l'estimation d'odométrie utilisant seulement des données issues de scanners laser 2D. Cette approche démontre l'intérêt des réseaux de neurones comme un bon moyen pour analyser ce type de données, dans l'optique d'estimer le déplacement du véhicule. Ensuite, nous étendons la méthode précédente en fusionnant le laser scanner 2D avec une caméra dans un système d'apprentissage de bout-en-bout. L'ajout de cette caméra permet d'améliorer la précision de l'estimation d'odométrie et prouve qu'il est possible de faire de la fusion de capteurs avec des réseaux de neurones. Finalement, nous présentons un nouvel algorithme hybride permettant à un véhicule de se localiser dans une région déjà cartographiée. Cet algorithme s'appuie à la fois sur une grille évidentielle prenant en compte les objets dynamiques et sur la capacité des réseaux de neurones à analyser des images.

Les résultats obtenus lors de cette thèse nous ont permis de mieux comprendre les problématiques liées à l'utilisation de capteurs bon marché dans un environnement dynamique. En adaptant nos méthodes à ces capteurs et en introduisant une fusion de leur information, nous avons amélioré la perception générale de l'environnement ainsi que la localisation du véhicule. De plus, notre approche a permis d'identifier les avantages et inconvénients entre les différentes méthodes classiques et d'apprentissage. Ainsi, nous proposons une manière de combiner ces deux types d'approches dans un système hybride afin d'obtenir une localisation plus précise et plus robuste.

Contents

1 Introduction **1**
- 1.1 Motivations . . . 4
- 1.2 Objectives of the Thesis . . . 5
- 1.3 Context of the thesis . . . 7
- 1.4 Publications . . . 7
- 1.5 Structure of the document . . . 8

I Context and theoretical background **9**

2 Perception for Autonomous Vehicles **11**
- 2.1 Introduction . . . 14
- 2.2 Sensors . . . 14
  - 2.2.1 LiDAR . . . 15
  - 2.2.2 Radar . . . 16
  - 2.2.3 Vision sensors . . . 16
  - 2.2.4 Dead-Reckoning and Inertial Positioning . . . 18
  - 2.2.5 Global navigation satellite systems (GNSS) . . . 19
  - 2.2.6 Conclusion . . . 19
- 2.3 Sensor Fusion . . . 20
  - 2.3.1 Fusion architecture . . . 21
  - 2.3.2 Fusion methods . . . 21
  - 2.3.3 Conclusion . . . 22
- 2.4 Map representation . . . 23
  - 2.4.1 Metric Maps . . . 23
  - 2.4.2 Topological maps . . . 24
  - 2.4.3 Hybrid Maps . . . 24
  - 2.4.4 Conclusion . . . 25
- 2.5 Conclusion . . . 25

3 Simultaneous Localization and Mapping **27**
- 3.1 Introduction . . . 30
- 3.2 Problem Formulation . . . 31
- 3.3 State-of-the-Art for SLAM methods . . . 33
  - 3.3.1 Model-based SLAM . . . 34

3.3.2 Learning-based SLAM . . . 41
3.4 Conclusion . . . 43

II Evidential-based SLAM 45

4 Sensor Fusion in Evidential Grid Maps 47
4.1 Introduction . . . 50
4.2 Preliminaries: Occupancy Grid Maps . . . 51
4.2.1 Bayesian Framework . . . 52
4.2.2 Dempster-Shafer Framework . . . 53
4.3 Proposed Approach . . . 56
4.3.1 Laser Scanner Model . . . 56
4.3.2 Sensor model for stereo camera . . . 57
4.3.3 Temporal Grid Fusion . . . 61
4.3.4 Life-Long Grid . . . 62
4.4 Experimental Results . . . 63
4.4.1 Multi-sensor local fusion evaluation . . . 63
4.4.2 Temporal fusion evaluation . . . 65
4.4.3 Quantitative results . . . 66
4.5 Conclusion . . . 67

5 Grid Matching Localization 69
5.1 Introduction . . . 71
5.2 Proposed Solution . . . 72
5.2.1 Pre-processing . . . 74
5.2.2 Intensity-based grid matching . . . 75
5.2.3 Re-localization . . . 76
5.3 Experimental Results . . . 76
5.3.1 Odometry Drift . . . 78
5.3.2 Influence of the weight matrix . . . 78
5.3.3 Influence of the time window . . . 79
5.4 Conclusion . . . 80

III Deep Learning Localization 81

6 An LSTM Network for Real-Time Laser-based Odometry Estimation 83
6.1 Introduction . . . 85
6.2 Preliminaries: Deep Learning . . . 86
6.2.1 Machine Learning . . . 86
6.2.2 Neural Networks . . . 87
6.2.3 Recurrent Neural Networks . . . 90
6.3 Proposed Solution . . . 93
6.3.1 Data encoding . . . 93

6.3.2 Network Architecture . . . 94
6.3.3 Training . . . 96
6.4 Experimental Results . . . 98
6.5 Conclusion . . . 101

7 Deep Sensor Fusion for Odometry Estimation 103
7.1 Introduction . . . 105
7.2 Proposed Solution . . . 106
7.2.1 Data Pre-processing . . . 106
7.2.2 Network Architecture . . . 107
7.2.3 Training . . . 109
7.3 Experimental Results . . . 111
7.3.1 Ordinal Classification . . . 111
7.3.2 Fusion vs Single Sensor Network . . . 112
7.3.3 Comparison to state-of-the-art . . . 113
7.4 Conclusion . . . 115

8 Deep Learning Localization in 2D Laser Maps 117
8.1 Introduction . . . 119
8.2 Proposed Solution . . . 120
8.2.1 Data Creation . . . 121
8.2.2 Data augmentation . . . 123
8.2.3 Network and Training Configuration . . . 123
8.3 Experimental Results . . . 126
8.4 Conclusion . . . 129

IV Conclusion of the Thesis 131

9 Conclusions and Perspectives 133

Appendices 137

A Complements on Chapter 4 139
A.1 Combination rules for evidential grid maps . . . 139
A.1.1 Conjunctive combination . . . 139
A.1.2 Disjunctive combination . . . 140
A.1.3 Dempster combination . . . 140

Bibliography 142

List of Figures 153

List of Tables 157

Abbreviations

AI	Artificial Intelligence
BA	Bundle Adjustment
BPA	Basic Probability Assignment
CNN	Convolutional Neural Networks
DGPS	Differential GPS
DST	Dempster-Shafer Theory
ET	Evidence Theory
EKF	Extended Kalman Filter
FOD	Frame of Discernment
FOV	Field-of-View
GPS	Global Positioning System
GNSS	Global Navigation Satellite Systems
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
LIDAR	Light Detection And Ranging
LSTM	Long Short-Term Memory
PnP	Perspective-n-Point
RCNN	Recurrent Convolutional Neural Network
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RTK	Real-Time Kinematic
SLAM	Simultaneous Localization and Mapping
SSL	Solid State LIDAR
UKF	Unscented Kalman Filter
VO	Visual Odometry

Chapter 1

Introduction

Contents

1.1 Motivations 4
1.2 Objectives of the Thesis 5
1.3 Context of the thesis 7
1.4 Publications 7
1.5 Structure of the document 8

Résumé du Chapitre 1

Après une décennie d'investissements massifs dans le développement de véhicules autonomes, la date d'arrivée d'une voiture véritablement autonome est encore inconnue. En particulier, il n'est toujours pas possible pour un véhicule d'être complètement autonome en milieu urbain. De nombreuses grandes entreprises travaillent partout dans le monde dans ce but, cependant, malgré toutes ces recherches et tous ces investissements, des accidents impliquant des véhicules autonomes et semi-autonomes se produisent encore. La plupart des problèmes pouvant entraîner des accidents proviennent du manque de précision dans la compréhension de l'environnement. Cette thèse est motivée par les défis encore ouverts dans la perception d'une voiture autonome. Nous nous concentrons sur le cas d'une voiture autonome équipée de capteurs à faible coût pour créer des méthodes plus robustes qui permettront de faciliter les tâches de planification et de contrôle de trajectoire, les rendant plus sûres et plus efficaces. Dans ce chapitre, nous présentons en détail les motivations et les objectifs de ces travaux, ainsi que la structure du reste de ce document.

1.1 Motivations

In recent years, there has been enormous efforts in the study and development of autonomous vehicles. An autonomous vehicle, also known as self-driving car, is a vehicle that is able to guide itself without human conduction. This topic of research started on the 1970s, where the first self driving cars were created. The research progress achieved in the last years has developed enough technology to make possible autonomous vehicles in controlled or simple environments.

What is the reason behind this increasing interest in self-driving cars? First of all, we may consider that for many people the idea of using a car that drives automatically seems fascinating in itself. However the technology innovation aspect is not the only reason why the world is so excited by this novelty. Many believe that autonomous driving is a key technology for the industry to be able to increase road safety. In USA, a National Highway Traffic Safety Administration (NHTSA) study [Singh, 2015] showed that 94% of the car accidents are caused by human errors. For this reason, the use of autonomous vehicles could be an important change to decrease the number of accidents. Besides road safety, there are several other advantages, such as transportation of elderly and people with disabilities, increase of the driver's productivity, reduction of road congestion and many more.

After a decade of massive investment in the development of autonomous vehicles, the date of arrival of a true self-driving car is still unknown. When it comes to urban environments, we are still not able to have a vehicle in operation which is completely autonomous. Several important companies are working to achieve this goal all over the world and even with all the research and investment, accidents are still happening involving autonomous and semi-autonomous vehicles [Wakabayashi, 2018][Boudette, 2018]. Most of the problems that can result in accidents come from the lack of precision in the understanding of the environment (pedestrians and other vehicles) and the prediction of their actions.

In order to perceive the environment these vehicles are equipped with several sensors, such as laser scanners, cameras and radars. The choice of the right sensors is fundamental for the safety of the driving task. However, it is important to maintain a reasonable price for the vehicle and for this reason, we can not use all the most precise and powerful sensors and computers available. Therefore, one of the main challenges is to chose the right sensor configuration along with the most fit system to process their data, fusion it and drive autonomously the vehicle.

There are three fundamental components for any autonomous robotic system to perform these tasks (Figure 1.1):

- **Perception** is related to the ability of an autonomous system to collect information from its sensors and extract relevant knowledge from the environment.
- **Planning** is the process of making decisions in order to achieve the goal of the system.
- **Control** refers to the ability to execute the planned actions that has been generated in the previous level.

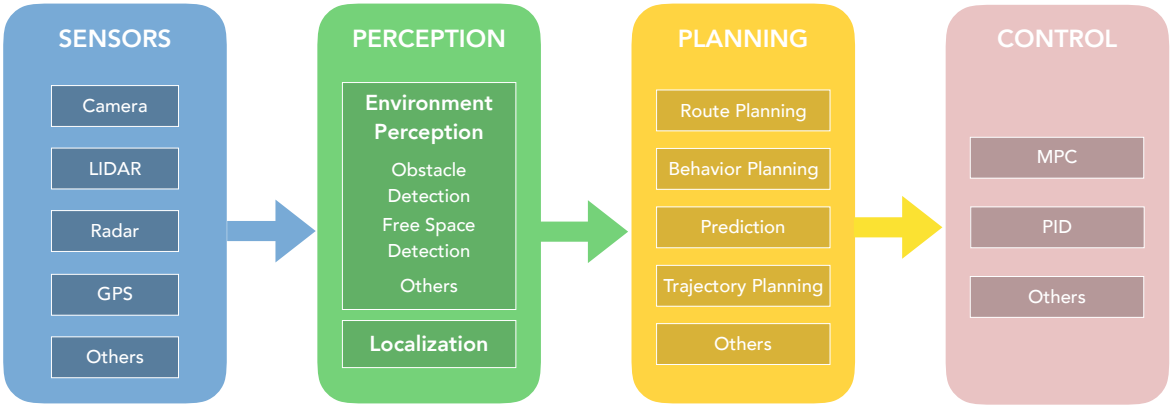


Figure 1.1: Components of an autonomous robotic system along with their most common tasks.

In Figure 1.1 we show some examples of tasks that each block is responsible to address. Each block task is highly dependent on the well performance of the tasks of a precedent block. For example, in order to perform safely the trajectory planning, it is necessary to have a reliable representation of the environment and a precise localization of the vehicle. This information is developed in the perception block and needs to be well adapted for the type of algorithms that are applied in the posterior components. These algorithms also need to be adapted for the type of sensors that a vehicle is equipped with. As mentioned before, low cost sensors could facilitate the deployment of autonomous vehicles, however these sensors provide less information with lower precision to the algorithms. Therefore, the methods developed at the perception block need to be able to deal with these difficulties.

In this thesis we are motivated by the still open challenges in the perception aspect. We focus on the case of an autonomous vehicle equipped with low cost sensors to create more robust methods that will allow to facilitate the tasks of path planning and control, making them safer and more efficient.

1.2 Objectives of the Thesis

As introduced in the previous section, the perception block is crucial for all the remaining tasks of an autonomous vehicle to be well performed. One of the main challenges is to find the best representation of the environment that will give enough information for the vehicle to drive autonomously. This information usually is given in the format of maps, which represents the obstacles around the vehicle and can provide even more complex information, such as the type of obstacles (pedestrians, vehicles, buildings) and their states (static, moving). The more precise and descriptive is the information defined in this block, the better the remaining tasks can be executed. For example, one of the open challenges for autonomous driving is the prediction of the other vehicles behavior. This is only possible if in the perception layer, we are able to detect those vehicles and track them overtime.

Considering this, one of the objectives of this thesis is to provide a suitable representation of the environment taking into consideration our constraints: low-cost sensors, that will maintain a reasonable price for the future self-driving cars, and the complexity of an urban environment. The use of low-cost sensors increase the necessity of sensor fusion in order to have a reliable understanding of the surroundings. While driving in an urban environment makes necessary that our methods deal with highly dynamic environments and with the lack of precise localization data. Taking this into account, in this work we chose to focus our mapping process using two sensors: 2D laser scanners and cameras.

The mapping process is often tightly correlated with the localization of the vehicle. At first glance, localization of autonomous vehicles seems significantly easy thanks to the use of GPS. However, the precision and availability of GPS data cannot be reliable for any situation. Therefore, it is necessary to create a system where the vehicle does not depend only on GPS information for having a precise localization. This is possible when we perform the localization based on the mapping of the environment. This problem is known as Simultaneous Localization and Mapping (SLAM), which has been one of the main research topics in the robotics field. In viewing of this, our objective is to create and apply SLAM algorithms to allow the vehicle to perceive the environment and to have a precise localization within it, considering the previously mentioned constraints.

Most of the existing SLAM techniques explicitly models the sensors characteristics, robot motion and the environment based on geometry. These classical methods are also referred as model-based SLAM. The state-of-the-art for model-based SLAM has become very popular in the robotics field in the last decades. However, they still face many challenging issues, specially in large-scale urban environments where there are a large amount of information to be processed and several dynamic obstacles. In the meanwhile, deep learning techniques have received a lot of attention in the computer vision field and researchers have been exploring how it could be applied to the SLAM problem. The use of learning-based methods could potentially deal with the difficulties found in model-based algorithm, facilitating not only the SLAM problem, but also the fusion of different sensors.

The research in this thesis aims to explore how the recent advances in Deep Learning techniques can be combined with SLAM to address the challenges of model-based only SLAM. Learning-based methods are already well adapted to receive as input camera images, however the use of laser scanners is not yet commonly explored by these techniques. By using neural networks, we could eliminate the need of sensor and environment modelling, making these approaches free to discover such representations as it sees fit. Our objective is to not only research how these techniques can be applied to the SLAM problem using 2D laser scanners and cameras, but also to be able to compare them to classical approaches, and even to explore the creation of a hybrid method (model and learning based techniques together).

1.3 Context of the thesis

This thesis was conducted at the Center for Robotics (CAOR) of MINES Paristech. The research was supervised by Dr. Cyril Joly and Dr. Arnaud de La Fortelle, and supported by the chair Drive for All, an international effort on the future of ground autonomous driving. The chair is sponsored by the industrial partners PSA Peugeot, Valeo and Safran, and it brings together four universities: MINES Paristech, EPFL, UC Berkeley and Shanghai Jiao Tong. The aim of this group is to gather international knowledge and apply it on the vehicles provided by the industrial partners. This group addresses the main challenges currently in the development of autonomous vehicle, such as path planning, vehicle control and perception.

1.4 Publications

The results presented in this thesis were published as conference or journal articles. The list bellow presents these publications and the corresponding chapters in this thesis.

1. Valente, Michelle, Cyril Joly, and Arnaud de La Fortelle. "Fusing Laser Scanner and Stereo Camera in Evidential Grid Maps." 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV). IEEE, 2018. (chapter 4)

2. Valente, Michelle, Cyril Joly, and Arnaud De La Fortelle. "Grid Matching Localization on Evidential SLAM." 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV). IEEE, 2018. (chapter 5)

3. Valente, Michelle, Cyril Joly, and Arnaud de La Fortelle. "Evidential SLAM fusing 2D Laser Scanner and Stereo Camera." Unmanned Systems (2019).(chapter 4 and chapter 5)

4. Valente, Michelle, Cyril Joly, and Arnaud de La Fortelle. "An LSTM Network for Real-Time Odometry Estimation." Intelligent Vehicles Symposium (IV). IEEE, 2019. (chapter 6)

5. Valente, Michelle, Cyril Joly, and Arnaud de La Fortelle. "Deep Sensor Fusion for Real-Time Odometry Estimation." International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019. (chapter 7)

6. Valente, Michelle, Cyril Joly, and Arnaud de La Fortelle. "Deep Learning Localization in 2D Laser Maps". Currently under review. (chapter 8)

1.5 Structure of the document

The remainder of the manuscript is organized in four parts as follow:
Part I introduces to the context of this work and it is divided into two chapters:

- In **chapter 2** the automotive context is introduced by presenting the main concepts of the perception of an autonomous vehicle, such as its most common sensors, how to perform the fusion between them and how to store their data.
- In **chapter 3** the theory of Simultaneous Localization and Mapping (SLAM) problem is explained. We present the state-of-the-art solutions to this problem from classical approaches to deep learning methods, using two main types of sensors: cameras and laser scanners.

Part II describes mapping and localization methods based on the Evidential theory. It is composed of two chapters:

- In **chapter 4** we first introduce the theory of occupancy grid maps and the Evidential model that will be used as base for our proposed approach. Sequentially, we propose a fusion method that uses evidential grid maps to improve the environment representation detected by a stereo camera and a 2D laser scanner. Moreover, we introduce a new life-long grid map layer that allows to distinguish between static and dynamic obstacles.
- In **chapter 5** we use the life-long grid map introduced in the previous chapter to propose a new localization solution based on image registration, where the differentiation between static and dynamic obstacles can increase the localization accuracy.

Part III explores the use of Deep Learning techniques to address the localization problem of autonomous vehicles. It is made up of three chapters:

- In **chapter 6** we first present the important concepts of Deep Learning to better understand the remaining chapters. Sequentially, we propose an end-to-end deep learning approach for real-time odometry estimation based on 2D laser scanners.
- In **chapter 7** we extend the method presented in the previous chapter, by creating a network that is able to fuse the input of a 2D laser scanner with the images of a monocamera.
- In **chapter 8** we propose an approach for localization that mixes the classic occupancy grid SLAM with deep learning techniques. The proposed method is able to relocalize a vehicle in a previous mapped region by estimating the odometry and the drift related to the map using convolutional neural networks.

Part IV concludes this thesis and it is composed of one chapter:

- In **chapter 9** the results of this thesis are discussed and summarized along with hints for future research.

Part I

Context and theoretical background

This part introduces to the context of this work. Details on the background allow to refine the topic of the thesis that touch both the state of the automotive industry and current research carried in the field of localisation. First, the main aspects of the perception of a self-driving vehicle are presented. In this section we expose the most common sensors along with their characteristics, how to fusion the information coming from different sources and how to store the data into maps. Sequentially, the theoretical background of the SLAM problem is presented along with the state-of-the-art from classical approaches to deep learning methods.

Chapter 2

Perception for Autonomous Vehicles

Contents

2.1 Introduction 14

2.2 Sensors 14

2.2.1 LiDAR 15

2.2.2 Radar 16

2.2.3 Vision sensors 16

2.2.4 Dead-Reckoning and Inertial Positioning 18

2.2.5 Global navigation satellite systems (GNSS) 19

2.2.6 Conclusion 19

2.3 Sensor Fusion 20

2.3.1 Fusion architecture 21

2.3.2 Fusion methods 21

2.3.3 Conclusion 22

2.4 Map representation 23

2.4.1 Metric Maps 23

2.4.2 Topological maps 24

2.4.3 Hybrid Maps 24

2.4.4 Conclusion 25

2.5 Conclusion 25

Résumé du chapitre 2

Les concepts clés permettant de donner la capacité de perception de l'environnement au véhicule autonome sont présentés dans ce chapitre. Tout d'abord, nous passons en revue les capteurs les plus courants qui permettent à un véhicule autonome de conduire et de percevoir le monde qui l'entoure. Ensuite, nous montrons comment les informations provenant de ces capteurs peuvent être fusionnées pour améliorer la compréhension de l'environnement. Enfin, nous présentons comment les informations recueillies à partir de l'ensemble des capteurs peuvent être représentées sous forme de cartes, qui sont un outil essentiel pour effectuer différentes tâches telles que la localisation et la planification de chemin.

2.1 Introduction

As presented in the previous chapter, the main objective of this work is to create mapping and localization solutions for an autonomous vehicle. To achieve this goal, we need to define important aspects of the perception of our robotic system. In this chapter we present these key concepts that will allow us to define later the constrains of our solutions.

Humans constantly interact with the world while simultaneously acquiring information from it. We hear sounds, see objects and touch them. However, it is our perception that gives meaning to these actions and allows us to decipher the information we get from the world. Moreover, only after we perceive the information that we acquire, we can perform the right action for it. For example, if we hear a sound, understand that it is a fire alarm, we know that it is necessary to evacuate from a building. However, without the perception and knowledge that we have to understand that it was a fire alarm, only the sound information would not be useful.

Now imagine how that works for an autonomous vehicle. The vehicle drives in a road acquiring a constant stream of information coming from its sensors, such as images, laser scanner pointclouds, radar distances and GPS positions. All this information can be then translated into pedestrians behavior, other vehicles localization, street signs and much more. The understanding of the meaning behind the different type of data is crucial for the vehicle to take the right actions and to perform fast decisions that can maintain the safety of its passengers. Just as perception allows humans to make associations and act on them, the ability to perceive the environment and understand it is a fundamental task for an autonomous vehicle to drive safely.

The key concepts in order to give this ability to perceive the environment to the system of an autonomous vehicle are presented in this chapter. First, we overview the most common sensors that allows an autonomous vehicle to drive and perceive the world. Sequentially, we show how the information from these sensors can be fused to increase the understanding of the environment. Finally, we present how the information gathered from the different sensors can be stored into maps, which are an essential tool to perform different tasks, such as localization and path planning.

2.2 Sensors

In this section we will highlight some of the most important characteristics of the sensors used on an autonomous vehicle. An example configuration with the most common automotive sensors is presented in Figure 2.1. Cameras, laser scanners and radars are the most commonly used sensors in autonomous vehicles to detect the obstacles and map the environment. Along with these sensors usually we can find a GNSS receiver, commonly known as GPS, for global positioning and for high-frequency positioning, sensors such as IMUs and wheel encoders.

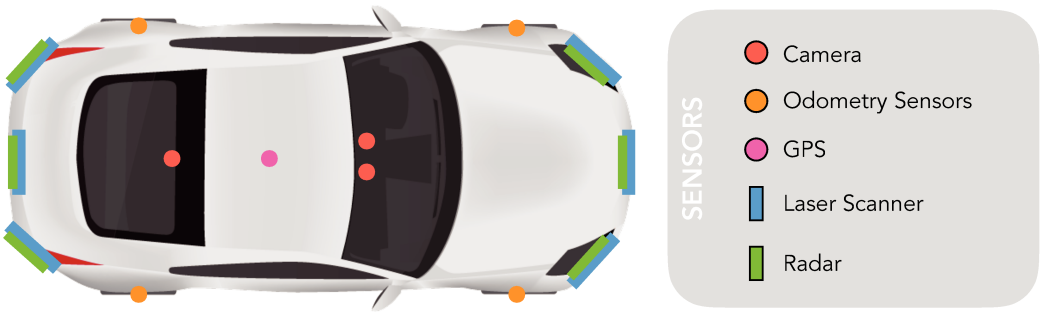


Figure 2.1: An example of sensor configuration for an autonomous vehicle

2.2.1 LiDAR

Light Detection And Ranging (LiDAR) is an active sensor based on the emission of laser beams at fixed angular steps. It measures the reflected pulses once the beam hits an object and analyzes it to calculate the distance to the object. This type of sensor typically operate in the near-infrared spectrum and some can work outdoors at ranges from a few meters out to over a hundred meters.

The most common and less expensive type of LiDAR sensors used in robotics applications are the 2D LiDARs. This means, they are able to see a planar slice of the world around them, but not above or below. There are also another type of LiDARs that can detect the world in 3D. This can be done either when multiple laser sensors are used, or when the laser sensor is rotated to take multiple scans. The first case is used by the Velodyne[1] line of 3D LiDAR sensors, while the second one is projected by researchers because of the lower cost [Surmann et al., 2001][Ricaud et al., 2017].

Almost all the current autonomous vehicles are equipped with a LiDAR. This popularity is due to the high accuracy of this kind of sensors. However, there are still some drawbacks such as the sensitivity to dust, rain, and snow. Rasshofer et al. [2011] provide an overview on the fundamentals of LiDAR systems and show how they can be influenced by weather phenomena as well. Besides the weather, LiDAR sensors can be sensitive to vibrations and impacts, which can influence the perception of autonomous vehicles.

The major limitations for 3D laser scanners in the automotive industry is the high price. Recently, the industry has been developing a more affordable 3D LIDAR [Ackerman, 2016a,b]. The goal is to replace mechanical scanning LiDAR, that physically rotate the laser and receiver assembly to collect data over an area that spans up to 360° with Solid State LiDAR (SSL) that will have no moving parts. This new technology can make this type of sensors suitable for automotive industry. There are also more affordable options to 3D laser scanners when the height field-of-view (FOV) is limited by a fewer number of laser scanner layers, these options usually help avoiding the problem of skidding produced by elevated or irregular ground.

[1] http://velodynelidar.com/

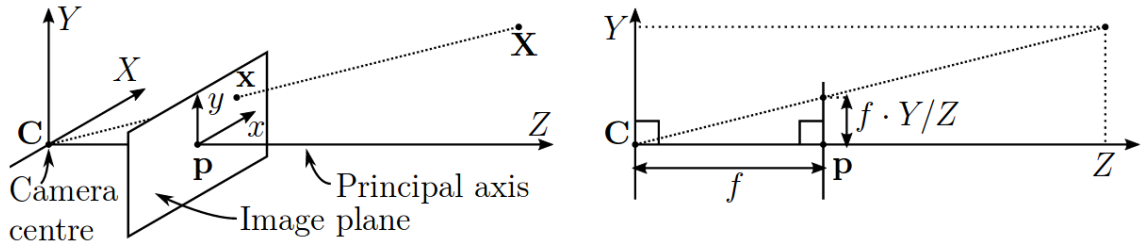


Figure 2.2: Pinhole camera model: a 3D point X is mapped to a point x on the image plane by the ray connecting the point and the center of projection C.

2.2.2 Radar

Radar stands for Radio Detetion and Ranging and it is a popular active sensing technology for vehicles. It can be used for different purposes such as adaptive cruise control, collision avoidance and obstacles detection. Just like laser based sensors, the radar emit strong radio waves and the receiver collects the reflected signals back. The range of obstacle is calculated by the time-of-flight information. Another advantage is that the velocity of the object can be calculated directly from the frequency shift between the emitted signal and Doppler echo.

Most of the radars operate at 24 Ghz and they are able to detect short and medium range obstacles. However lately it has become more common the use of long-range radars at 77 GHz, it has a low resolution but can detect obstacles up to 200m away. They are a popular choice of sensor because they are robust mechanically and operate effectively under a wide range of environmental conditions. They can provide range and azimuth measurements as well as range rates. Moreover, they generate less data, which can reduce the computational power.

2.2.3 Vision sensors

Vision sensors are classified as passive sensors, once they do not emit any ray and perceive the environment based on the different wavelength spectra. The drawbacks using this sensors usually are related to environment aspects, like lack of light or weather conditions (rain, snow, dust). However, the cameras allow you to detect environment close what a human eye can see. Therefore, the main advantage for this kind of sensor is the available methods for distinguish objects that are detected based on their physical properties, such as texture, color and contrast. This ability allows autonomous vehicles to perform several important tasks, like detection of traffic lights, pedestrians, other vehicles and lane markings. The availability and price make them applicable for automotive applications in large scale.

Most commercial cameras can be described as pinhole cameras, which are modeled by a perspective projection, shown in Figure 2.2. It defines the mathematical projection from 3D world coordinates to 2D image plane coordinates. This transformation consists in first a projection from 3D world coordinates to 3D camera coordinates, and then the projection from 3D camera coordinates to 2D image coordinates.

Figure 2.3: Perspective, diotropic and catadioptric images. Image from [FLÓREZ and Stiller, 2011].

However a real camera is not perfect and sustain a variety of different characteristics. For geometric measurements, the main concern is camera distortion. In computer vision geometric measurements is essential to have an accurate knowledge of the image projection parameters. Those parameters can be found by performing a camera calibration process. It consists in recovering the following parameters:

1. The intrinsic camera parameters, i.e. the inner transformations of the camera, including focal length, position of the principal point, sensor scale and skew factor.

2. The non-linear lens distortion parameters.

3. The external transformation parameters for each of the views of the camera in the calibration process.

There are several strategies to camera calibration, since correctly calibrated cameras are required for many applications in perception. Some approaches make use of a special, calibrated 3D setup, where the position of all 3D points and the camera center are known [Heikkila and Silven, 1997]. Other approaches, more utilized by researchers for its simplicity, use multiple views of a 3D pattern of known structure but unknown position and orientation in space [Zhang, 1999]. Finally, there are some methods that make no assumptions about the 3D structure of the scene, known as self-calibration methods [Faugeras et al., 1992].

The combination of two projective cameras with overlapping field-of-views provides a stereo imaging system able to give 3D range measurements. This is done by finding the disparity between the simultaneous images captured by the left and right cameras, which consists in the number of pixels a particular point has moved in the right camera image compared to the left camera image.

A perspective camera covers typically a 45 horizontal field-of-view (Figure 2.3(a)). Some automotive applications, however, require to cover larger zones. To this end, a perspective camera can be modified by the use of a wide-lens, creating a so-called fish-eye system (i.e. dioptric camera) [Miyamoto, 1964]. Fish-eye cameras can cover up to 180 horizontal field-of-view (Figure 2.3(b)), but radial lens distortions cause a nonlinear pixel mapping of the image plane. This can add complexity to image processing algorithms. Kannala and Brandt [2006] presented a generic geometric model

and a calibration method based on a planar calibration pattern that can be used for both fish-eye lenses and conventional cameras.

Another interesting vision sensor is the catadioptric camera [Nayar, 1997] (also called omnidirectional camera). It consists of a perspective camera with a convex mirror and it provides a 360 field of view in a single image (Figure 2.3(c)). It is worth mentioning that images obtained with the use of catadioptric cameras are characterized by a low resolution and a central blind spot. Mei and Rives [2006] propose a model based on the exact theoretical projection function and with the addition of parameters to model real-world errors, in order to calibrate omnidirectional single viewpoint sensors from planar grids.

A different approach to have a wider field-of-view is to use a cluster of cameras. This group of cameras are known as omnidirectional multi-camera system (OMS) or polycameras. The series Ladybug by PointGrey[2] is an example of this kind of system. Taking the Ladybug3 as an example, it comprises six individual cameras and provides a 360 field-of-view.

2.2.4 Dead-Reckoning and Inertial Positioning

Dead-Reckoning (DK) is the process of estimating the position and orientation of a robot based on the previous position measurements. The most simple way of providing this estimation is using encoders, which are rotatory sensors usually fixed to the wheels of the robot. However, it is not possible to estimate lateral movements or quantify slippages using only encoders. For this reason, the odometry measurements usually is completemented with Inertial Measurements units.

Inertial Measurement Unit (IMU) [Morrison, 1987] is a device that measures linear and angular motion with a combination of gyroscopes and accelerometers. Once this sensor is connected to the vehicle, it provides a continuous stream of data related to the linear acceleration of the car on three principal axes, combined with angular velocity values.

Their use in autonomous vehicles is to provide an independent source for computing the position and orientation of the vehicle relative to some initial pose. This sensor may be sometimes the only means of navigation when other sensors, like GPS, are unavailable.

The main advantage of these sensors is that they are not subject to external factors. However, their precision is not high and after some measurements it can contribute to generate an integration drift, which means that the estimation of the pose of the sensor deteriorates over time. Considering this, IMU is a common sensor used as complement for high-frequency local positioning and it is very often used with other sensors to perform data fusion.

[2]https://www.ptgrey.com/360-degree-spherical-camera-systems

2.2.5 Global navigation satellite systems (GNSS)

The main idea of a GNSS is to use receivers to measure the time of arrival of satellite signals and compare it to the transmission time to calculate the signals propagation time. This time is used to estimate distances from the GNSS receiver and the satellites. From these distances, GNSS receivers calculate the position by means of multilateration which relies on multiple satellite measurements to produce a position fix. At this time there are several examples of GNSS in operation: the USA's Global Positioning System (GPS), the Russian GLONASS, the Europe's Galileo positioning system and China's BDS. GPS is the most popular and that is why a GNSS system is commonly called by GPS.

Due to errors such as Ionospheric and Atmospheric delays, and signal blocking, inexpensive commercial GPS receivers can have not a reliable accuracy. However, additional hardware and infrastructure can be added to reduce these errors. Differential GPS (DGPS) [Parkinson and Enge, 1996] is an enhancement to GPS that provides improved position and timing performance. It uses one or a group of fixed, ground-based reference stations whose positions are accurately known. Each station is equipped with at least one GPS, and it broadcasts the differences between its GPS observation and internally computed observations. The most commonly DGPS technique used is the RTK-GPS (Real-Time Kinematics GPS) [Langley, 1998] created in the mid-1990s. It requires the installation of a reference station at a fixed and known location closed to the mobile GPS receiver. Through a radio link the reference station transmits data to the mobile GPS. The mobile GPS receives the data from the station and the data coming directly for its own GPS unit and processes it. Since the two GPS are closed to each other and one remains at a fixed location, errors and position ambiguities can be significantly reduced and provide an accuracy of 1 to 5 cm [El-Rabbany, 2002]. However, to use this kind of technology, it is necessary to have several reference stations and afford the access to them, these reasons increase the cost and make it not a viable commercial solution.

2.2.6 Conclusion

In this section, we presented the most common sensors that can be found in an intelligent vehicle. In Table 2.1 we can observe the different characteristics for cameras, radars and LiDARs. It is easy to notice the complementary aspects of these sensors and how their fusion could increase the perception of the environment for an autonomous vehicle. There are different parameters that can be used to evaluate the sensors, such as accuracy and robustness. Accuracy refers to how close the measurements are to the true values, while robustness is used to evaluate the quality of being reliable and unlikely to fail in challenging conditions. In this work, we focus on solutions based on low-cost sensors, which usually comes along with the drawback of lower accuracy and robustness. Therefore, in order to perform robust and safe driving in these conditions, it is even more necessary to fuse the information coming from a variety sensors. Considering this, in the next section we will present an overview of the different ways sensor fusion can be performed.

	Camera	**Radar**	**LiDAR**
Object Detection	Medium	High	High
Object Classification	High	Low	Low
Distance Estimation	Medium	High	High
Velocity Estimation	Low	High	Medium
Lane Detection	High	Low	Low
Functionality in poor lighting	Low	High	High
Functionality in bad weather	Low	High	Medium
Cost	Medium	Low	High

Table 2.1: Each of the three main perception sensors has their own advantages and disadvantages. In this table we can observe that radar, LiDAR and cameras are more complementary than competitive, making their fusion necessary for autonomous vehicles.

2.3 Sensor Fusion

In the previous chapter we analyzed the different sensors of an autonomous vehicle and how they have complementary advantages and disadvantages (Table 2.1), making data fusion an important task to create a robust and safe perception system. The vehicle perception can be improved in many aspects by merging data from different sensors, such as having a more rich representation of the environment, more precise measurements and a better management of the uncertainty of each sensor. To provide this, fusion approaches need to take into account several aspects for each sensor: the nature of the data, the field of view, synchronization times, and frequency.

Although combining information from different senses is a quite natural and effortless process for human beings, imitate the same process in robotic systems is an extremely challenging task [Chavez-Garcia, 2014]. By fusing redundant information coming from different sensors we obtain a more precise and trustful output that can generate better decisions. There are three main challenges to perform data fusion. First, a precise data association process need to be done, therefore a calibration method is necessary in order to be able to correspond the data coming from different sources. Second, it is necessary to create a design architecture of the fusion approach, where it will be defined in which stage the fusion will be performed. And third, define how the uncertain and imprecise data is managed. These steps are different depending on the application and the type of sensors. In this section we will first overview the fusion architectures that can be applied and sequentially the types of methods that can manage the uncertain and imprecise data to perform the fusion.

2.3.1 Fusion architecture

In order to merge data coming from two different sources we need to define in which step of the process we will associate them. Intelligently combining information from the sensors will give a more complete view of the world and will improve the perception of a robotic system. Obtaining a classification of the different fusion architectures is a difficult task due to multidisplinarity and the large number of case studies reported in the literature. However, for autonomous vehicles we can try to separate the different algorithms into two main groups:

1. **Low-level**: the raw sensor data coming from each sensor are fed to a fusion algorithm. For example, in the mapping task context, we can simply transform all sensor data into metric information, and sequentially perform the fusion [Baig and Aycard, 2010][Valente et al., 2018a]. This can be done directly since all the data will be in the same format.

2. **High-level**: the raw sensor data is processed and passed through an algorithm, which depends on the task, before performing the fusion. For example, in the context of an obstacle detection method, the data from different sensors can be passed to different algorithms that perform obstacle detection, and then the output of the different obstacles detected can be fused to have a complete view of the environment [Chavez-Garcia and Aycard, 2015][Wei et al., 2018].

This classification is directly related to the level of abstraction of the sensor data. We could also consider a third classification, a hybrid fusion, where a method could take the raw data of a sensor and fusion with the processed data of another sensor. Each of these fusion architectures has certain advantages and disadvantages depending on the application and the types of sensors applied to the task.

2.3.2 Fusion methods

Several methods have been proposed to combine information coming from different sensors. Chavez-Garcia [2014] separates the most commonly used approaches to multi-sensor data fusion into three categories: probabilistic fusion, evidential belief reasoning and fuzzy logic. Here we summarize the main characteristics of these methods.

1. **Probabilistic Fusion**

   Bayesian [Bernardo and Smith, 2009] methods or probabilistic methods, rely on the probability distribution to represent the uncertainty of the sensor data. The probabilistic data fusion is generally based on the Bayes theorem for combining information. In practical, the fusion can be implemented in a number of ways, such as using Kalman or extended Kalman filters, through sequential Model Carlo methods, or through the use of functional density estimates [Durrant-Whyte and Henderson, 2008].

2. **Evidential Belief Reasoning**

   Evidence Theory (ET), also known as Dempster-Shafer theory of evidence, [Wu et al., 2002] adds the notion of assigning beliefs and plausibilities to possible measurement hypotheses along with the required combination rule to fuse them. This type of approaches aim at quantifying different degrees of belief. This is interesting because it allows each sensor to contribute information in different levels of details. It can also be interesting to model dynamic environments, where there are a lot of uncertain or imprecise information.

3. **Fuzzy Logic**

   Due to the powerful theory to represent vague data, fuzzy set theory is particularly useful to represent and fuse sensor information [Russo and Ramponi, 1994]. Fuzzy logic allows the uncertainty in multisensor fusion to be directly represented in the inference process by allowing each proposition, as well as the actual implication operator, to be assigned a real number from 0.0 to 1.0 to indicate its degree of truth. This normalization process allows efficient fuzzy data fusion when incomplete or vague data is used.

4. **Deep Learning**

   In Chavez-Garcia [2014] the authors did not mention the use of deep learning methods for sensor fusion, however in the last years the interest of using this type of techniques for this purpose has been increasing. The interest of using this type of approach is that a lot of constraints and difficulties of sensor can be facilitated. For example, sensor calibration is not necessary, since these parameters can be learned from the network directly. Moreover, we do not need to define a common model for the fusion, since most of the time the fusion can be performed by just concatenating the information from the sensors. This information can be passed through a specialized network before or not.

   Most of the existing deep sensor fusion solutions are based on object detection, like in [Xu et al., 2018] where the authors fuse 3D laser scanners and camera images to predict object's bounding boxes. There are also methods that use sensor fusion for end-to-end learning [Patel et al., 2017], where the input is the data of different sensors and the output is directly steering commands.

2.3.3 Conclusion

In this section, we gave a brief overview of how data fusion can be performed to increase the perception of the environment. In this work, the sensor fusion will be explored at different times in order to increase the information detected in the environment. The improvement in perception will be necessary not only for mapping purpose (chapter 4), but also to increase the accuracy in localization methods (chapter 7).

In the next section, we will present how the information detected by different sensors can be stored into maps. The use of maps can be necessary for different tasks of an autonomous vehicles and can be a common data format to perform sensor fusion.

2.4 Map representation

After choosing the sensors and how to perform their fusion, a very important step for the perception of autonomous vehicles is to choose the right map representation in order to store the information coming from the sensors. Depending on the type of sensor, application and environment one map representation can be more suitable than others.

The main representations of maps are classified into three main categories:

1. **Metric** maps represent the geometric properties of the environment, such as the distance to the obstacles in the environment.

2. **Topological** maps use a high level of abstraction, where the environment is represented by a purely symbolic description.

3. **Hybrids** maps are based on the mix between topological and metric characteristics.

2.4.1 Metric Maps

In metric maps, the information gathered from the sensors are stored representing the metric information of the obstacles detected. This type of map is robust to map large-scale environments and it is popular to use to Simultaneous Localization and Mapping (SLAM) approaches, where the metric information helps to estimate the localization of the mobile robot. In the literature we can find two main types of metric maps: feature-based and grid-based maps.

2.4.1.1 Feature-based maps

For this representation, the raw data is processed to identify and extract features which will be used to build the map. The main goal of this kind of strategy is to provide a more compact representation of the environment. A common approach is to identify geometric primitives like points, lines and circles. However, it is a hard task to do in irregular scenarios like in outdoor environments. This approach is also common in visual SLAM where features of the environments are extracted using image features [Pink, 2008].

In general this type of map can be more compact than grid-based maps if the mobile robot is in a well structured environment. Moreover, this type of map are closer to the kind of perception of environment that humans have, which can provide accurate results if the geometric primitives are chosen correctly. The main drawback of this kind of map is that only environments containing the basic geometric primitives can be correctly represented.

2.4.1.2 Grid-Based

Grid-based maps were first introduced in [Elfes, 1991] in the format of an Occupancy Grid and has been one of the most common representations for popular SLAM solutions. The map represents the environment in regular cells and the occupancy state of the cell indicates the probability of the cell to be occupied by an obstacle or not. There are different strategies to update the state of the cell once a new data arrives, such as Bayesian filtering, Dempster-Shafer and Fuzzy logic.

Bayesian filtering [Coué et al., 2006] is the most common background used, since it is easy to cope with errors and uncertainty coming from the sensors. Moras et al. [2011] introduced a variation of the occupancy grid called Credibilist Occupancy Grid, also known as Evidential Grids, that is based on Dempster-Shafer theory. This representation offers an interesting solution to differentiate between unknown (no information) and doubt caused by conflicting information in the matching process. The extra information provided by this map representation is a good approach for dynamic environments. Another popular grid-based map are the Octomaps [Wurm et al., 2010], which is an extension of an occupancy grid map to a 3D space model. It consists of an octree data structure that is represented by a tree with nodes, where each parent node splits into eight equal-sized voxels.

2.4.2 Topological maps

This representation has a high level of abstraction, where the environment is represented by a purely symbolic description. The topological map can be viewed as a graph of places, where at each node, the information concerning the place and the way to each other places connected to it, is stored. Usually this kind of map are generated on top of a grid-based or feature-based map by breaking the map into coherent regions [Chang et al., 2007]. The fundamental weakness of this map resides in the lack of metric information, and the difficulty of using this map for dynamic environments. Topological mapping has a long tradition in mobile robotics. In one of the first articles on topological mapping [Simmons and Koenig, 1995], a Partially Observable Markov Decision Process (POMDP) model is used to estimate the position of a robot as a probability distribution. More recently, Bernuy and Ruiz-del Solar [2018] uses this type of maps along with semantic information obtained from neural networks to perform a localization algorithm which uses a Particle Filter for obtaining vehicle's pose.

2.4.3 Hybrid Maps

The characteristics of metric and topological maps are complementary, which made researchers explore the use of both in hybrid maps. In the context of autonomous vehicles, the use of hybrid maps is explored when common road objects are used to improve the use of metric maps. For example, in Choi [2014] the authors propose a hybrid map-based SLAM. Their method describes the environment traversed by a vehicle using a grid map and a feature map together, where tall objects such as street lamps and trees are added.

2.4.4 Conclusion

In this section, we gave an overview of the different types of maps that can be used to perceive the environment from a robotic system. The use of maps is fundamental to perform a variety of necessary tasks of any autonomous system. In our work, the maps are used for different purposes, such as to perform data fusion, to differentiate the obstacles detected in the environment and for our localization methods. In chapter 4 grid-based maps will be presented in more details, more specifically probabilistic occupancy grid maps and evidential grid maps.

2.5 Conclusion

In this chapter, we presented the main aspects of the perception of an autonomous vehicle. We showed the different types of sensors that can be applied to autonomous vehicles. It is possible to observe that each sensor has their own characteristics, which makes them more reliable for certain applications than others. For this reason, it is necessary to use a variety of sensors when it comes to a complex system like as an autonomous vehicle. Considering this, we introduced the main concepts of sensor fusion that are necessary for the perception of this system. Finally, we presented the different ways to store the information gathered from the sensors in the form of maps.

Perception is only the first stage in the pipeline for the functioning of an intelligent vehicle. Once the vehicle is able to extract relevant data from the surrounding environment, it can use it localize itself, to plan the path ahead and to actually actuate, all without human intervention. Each stage has its own challenges in order to create an autonomous vehicle, however all of them rely on the fact that we need real time processing and even more important, they need to be a robust and reliable systems.

In the next chapter, we will present how we can use this information to localize the vehicle while simultaneously mapping the environment. This task is a popular problem in the robotic community known as SLAM, and it is necessary for an autonomous vehicle to map its surroundings and to have a precise localization that is not dependent on a GNSS device.

Chapter 3

Simultaneous Localization and Mapping

Contents

3.1 Introduction **30**

3.2 Problem Formulation **31**

3.3 State-of-the-Art for SLAM methods **33**

3.3.1 Model-based SLAM 34

3.3.2 Learning-based SLAM 41

3.4 Conclusion **43**

Résumé du chapitre 3

Dans ce chapitre, nous présentons plus en détail la problématique du SLAM, ainsi que sa notation mathématique et les méthodes de pointe pour le résoudre. Nous séparons les méthodes présentées en deux catégories principales : SLAM basé sur un modèle existant et SLAM basé sur l'apprentissage. La première catégorie est constituée des approches SLAM classiques, tandis que la seconde regroupe de nouvelles méthodes qui ont émergé avec la popularité des techniques de Deep Learning. Nous concluons enfin cet état de l'art en présentant l'objectif de cette thèse en regard de ce qui a été présenté jusqu'ici.

3.1 Introduction

We presented in the previous chapter the key aspects of the perception for an autonomous vehicle. After gathering the information from different sensors and storing it, the next step is to create a map of the environment during its trajectory while simultaneous localizing the vehicle in this map. In this chapter, we present the theoretical background and the state-of-the-art for this challenging task.

Simultaneous Localization and Mapping (SLAM) is the technique used for mobile robots to build a map of the environment and at the same time use this map to determine the robot's location. The SLAM problem was first introduced by Hugh Durrant-Whyte and John J. Leonard [Leonard and Durrant-Whyte, 1991], which was based on the work of Smith, Self and Cheeseman [Smith et al., 1990]. This problem is known as a chicken egg problem, since an accurate map is necessary for reliable location and an accurate location is essential for building a consistent map.

The most simple form of localizing a robot inside an environment is using its odometry. For example, in a terrestrial mobile robot one can use the displacements measured by encoders in their wheels to estimate the current position and orientation of the robot. However, error is accumulated when the robot moves and eventually is so large that we cannot have a good estimation of the robot position. The main challenge is to provide a way of correcting the estimation of the robot state (position and orientation) while simultaneously creating the map of the environment. One of the main challenges is not only to provide a precise estimation of the obstacles in the environment, for the robot to avoid them and safely arrive to a destination, but also to differentiate between static and dynamic obstacles. This is important for the SLAM problem since the static obstacles allow us to make an easier correspondence that will be used for estimating the robot's position.

The SLAM algorithms provide the tools to have a simultaneous estimate of both robot and landmarks location. They consist in using the robot's observations to detect landmarks and add them to a map, which can be a simple metric map with the location of obstacles or a more complex topological map. Sequentially the map is updated when new observations are made. Once the robot returns to observe the same landmarks we can use their locations to reduce the accumulated error of both robot localization and the complete map. This step is known as loop closure and is a challenging step of SLAM. It is important to notice that to perform the map update with accuracy it is necessary to have a good estimation of the localization, that also depends on a good landmark matching. This is why the the localization and mapping tasks are codependent, which makes it a complex problem to solve.

In this chapter, we present in more details the SLAM problem, along with its notation and the state-of-the-art methods for solving it. We separate the presented methods in two main classes: model-based and learning based SLAM. The first consists of the classic SLAM approaches, while the second represents new methods that have emerged with the popularity of Deep Learning techniques. Sequentially, we conclude the state-of-the art by showing the objective of this thesis considering what has been explored so far.

3.2 Problem Formulation

The general SLAM process consists in a robot moving around a region through a given dynamic model, measuring relative locations of landmarks, and building a map of the landmarks. The map is simultaneously used to update the robot's pose. The robot's movement and perception generates noise and uncertainty, making gradually more difficult to determine the robot or the landmarks positions. The Figure 3.1 illustrates the presented situation, where for each instant t the following notations are used:

- x_t: robot state at time t. It usually stores the position and orientation of the robot.
- u_t: the control vector that contains the commands issued to move the robot from the position at $t-1$ to the current position at time t.
- $m_{(i)}$: the position of the i^{th} landmark.
- $z_{t,(i)}$: observation of the i^{th} landmark from the position x_t at time t.

As a result of the noise and uncertainty, the SLAM problem is usually solved in a probabilistic form. The methods employ probabilistic models for the robot movement and perception, and they rely on probabilistic inference for estimating the state of the robot and the map. The dominant scheme used is the Bayes filter, which extends the Bayes' rule to temporal estimation process. It consists in a recursive estimator for computing a sequence of posterior probability distributions over quantities that cannot be observed directly, such as a map.

The classical formulation of the SLAM problem requires that the probabilistic distribution

$$P\left(x_t, m | Z_{0,t}, U_{0,t}, x_0\right) \tag{3.1}$$

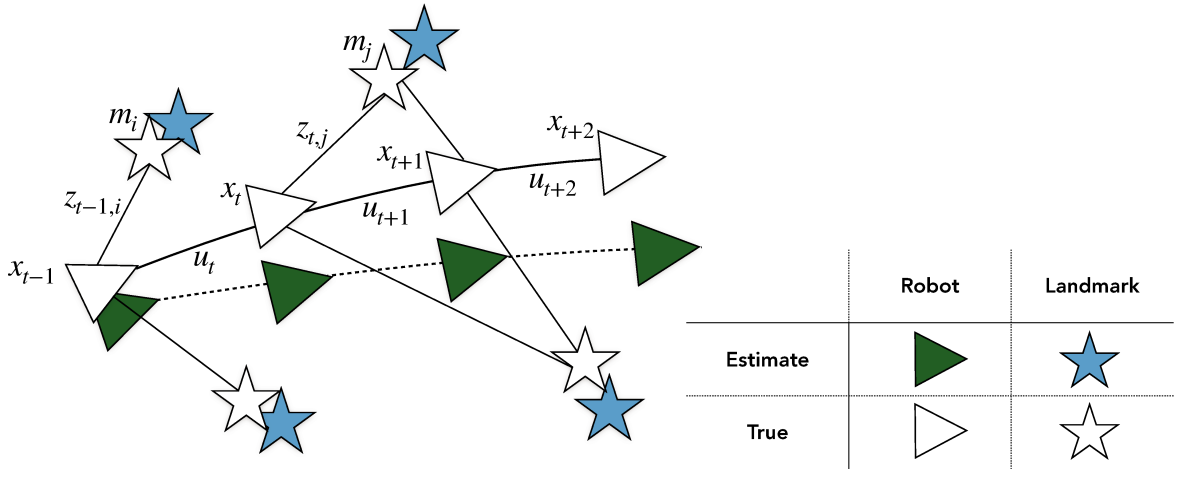


Figure 3.1: SLAM problem representation.

is computed for all times t. This distribution describes the joint posterior density of the robot position x at time t and the landmark locations m, considering the observations Z and motion commands U from time 0 until time t, and the initial pose x_0.

In order to solve this problem, we need to define a model that describes the control of the robot and the observations. These models are usually called state transition model and observation model, respectively.

The state transition model, or motion model, is generally described in the form

$$P\left(x_t|x_{t-1}, u_t\right). \tag{3.2}$$

That is, it is assumed as a Markov chain in which the next state x_t depends only on the immediately preceding state x_{t-1} and the applied control u_t. Considering this, if x_{t-1} and u_t are known, the knowledge of the previous states, control and landmarks are useless.

The observation model is generally described in the form

$$P\left(z_t|x_t, m\right). \tag{3.3}$$

It describes the probability of making an observation z_t when the vehicle localization x_t and landmark locations m are known.

Using the Bayes filter framework, Equation 3.1 can be calculated in two steps: prediction and update.

1. Prediction: it calculates the prior probability distribution based on the posterior probability computed one step-time before and the state transition probability distribution.

$$P\left(x_t, m|Z_{t-1}, U_t, x_0\right) = \int P\left(x_t|x_{t-1}, u_t\right) P\left(x_{t-1}, m|Z_{t-1}, U_{t-1}, x_0\right) dx_{t-1} \tag{3.4}$$

2. Update: it employs Bayes' Theorem, based on the observation model distribution and the prior distribution in order to compute the joint posterior distribution.

$$P\left(x_t, m|Z_t, U_t, x_0\right) = \frac{P\left(z_t|x_t, m\right) P\left(x_t, m|Z_{t-1}, U_{t-1}, x_0\right)}{P\left(z_t|Z_{t-1}, U_{t-1}, x_0\right)} \tag{3.5}$$

This formulation is known as the online SLAM problem. There are other two main types of approaches to solve this problem: full SLAM [Hess et al., 2016][Kohlbrecher et al., 2011] and trajectory-oriented SLAM [Montemerlo et al., 2002][Mendes et al., 2016]. In the full SLAM problem instead of just estimating the current robot state, the entire robot path and the map are estimated. While in the trajectory-orientated SLAM we only estimate the entire robot path and use observations to landmarks only to refine the path estimates. In other words, we don't estimate the landmarks locations, this is usually applied when the number of features is much bigger than the number of robot poses.

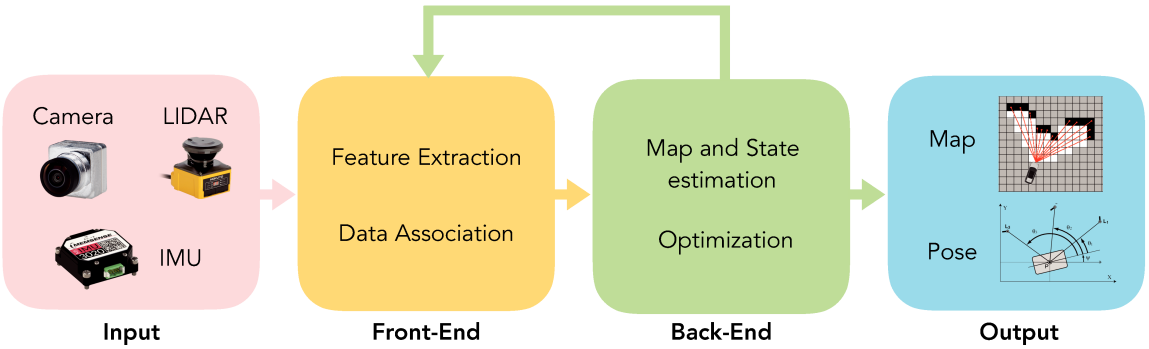


Figure 3.2: Architecture in a classic model-based SLAM.

3.3 State-of-the-Art for SLAM methods

The SLAM problem is a popular topic in the robotics community. Its solutions started with the classic probabilistic models, known as Kalman Filters, and since then many different solutions emerged. The classical approach blocks are illustrated in Figure 3.2: first, the information from the sensors is pre-processed and, if more than one sensor is being used, their data can be fused. Sequentially the front-end methods will analyze the input information extracting the important features and will perform data association between new features and the old ones. Finally, back-end methods can be used to optimize the map and pose estimation. The output is the map of the environment, in the chosen format, and the state of the robot, which consists of its position and orientation.

Recently, along with the new trend of Deep Learning technology in the computer vision community, researchers started to explore this type of techniques for the SLAM problem as well. It started mainly with the use of camera images, in order to perform Visual SLAM, but lately every year new methods are presented for a variety of sensors and applications. In the Deep Learning SLAM methods all blocks defined for the classical SLAM can be turned to one single main block, the neural network, in case of end-to-end learning methods. However, it is also common to see solutions where they only replace one task of the SLAM solution, like the feature extraction, for a neural network, maintaining the rest of the classical blocks. This type of solution can be interesting because it can get the advantages of the neural networks, such as fast processing and learning of features, along with the probabilistic information coming from classic approaches, which allows to understand better the quality of the results, that can be necessary for safety reasons.

Considering this, in this state-of-the-art section we present the most popular SLAM methods, divided in two categories: model-based SLAM, which consists of the classic approaches, such as probabilistic and optimization methods, and the Learning-based SLAM, which explores the use of neural networks for this problematic.

3.3.1 Model-based SLAM

The classic architecture of a SLAM system includes two main components (Figure 3.2): the front-end and the back-end. The front-end interprets the data coming from the different sensors in order to extract the constraints and to match them in different times, while the back-end typically applies optimization techniques to estimate the robot and the constraints states, providing the robot's localization and the map.

It is important to mention that not all SLAM solutions have the two components, back-end and front-end. Some solutions can be able to estimate the map of the environment and the vehicle localization using only directly the front-end [Valente et al., 2018b]. However, drift of the pose estimation is inevitable, and back-end methods can be introduced to minimize the noise in motion (an estimate of the robot's future position) and observation (the actual measurement) models.

First, we will introduce the most popular front-end approaches for camera and laser scanner SLAM. Most of those methods have a back-end method behind taking care of the map and pose optimization. For a better understanding of these optimization methods, we will sequentially briefly introduce the most common approaches.

3.3.1.1 Front-end: Feature Extraction and Data Association

The SLAM front-end is highly dependent on the application and the type of sensors used. The main tasks of the front end is to extract features and to perform data association. In the full SLAM problem, the front-end must choose the landmarks from the raw data and associate them with previously viewed landmarks. While in the pose SLAM problem, the front-end must identify pairs of poses with overlapping views of the environment and produce a relative constraint between them. Data association is a challenging task when the mobile robot returns to a previously explored location after a long period of time: this is known as the loop-closure problem.

By performing data association we recover the motion estimation of the robot, which involves estimating the relative spatial transformation between two frames. Although dead reckoning using frame-to-frame motion provides a good local estimate of the robot's trajectory, the gradual accumulation of error over time causes inconsistencies. The back end presented in the next section can be one of the approaches to solve these inconsistencies and construct a globally consistent map from frame-to-frame measurements.

1. **LIDAR (Light Detection and Ranging)**

   The process of data association for LIDAR sensors consists in matching two different scans, this task is also known as scan matching or point matching. There are two main categories for laser range scan matching: point to point and feature to feature matching methods. Depending on the application, the scan matching can happen between two laser scans or between a laser scan and a map.

   The most widely known algorithm for point to point scan matching is iterative closest point (ICP) [Besl and McKay, 1992]. This method is used to find a linear transformation that best aligns two point clouds and in doing it, estimates the

localization mismatch. There are two fundamental simplifying assumptions in the ICP method, which are somehow optimistic and decrease this method's accuracy, as follows: i) Matching assumption: the corresponding points of two scans are successfully matched; ii) Correspondence assumption: the points of two scans, which are correctly matched with each other, correspond to exactly the same point of the environment's boundary [Aghamohammadi et al., 2008].

Several variants of the ICP algorithm address this problems and try to compensate it [Hong et al., 2010; Minguez et al., 2005; Moosmann and Stiller, 2011; Pfister et al., 2002; Zhang and Singh, 2014]. Along those methods, LiDAR Odometry And Mapping (LOAM) by [Zhang and Singh, 2014] has become considered state-of-the-art in 6-DOF LiDAR SLAM. LOAM runs two different algorithms in parallel to achieve real time processing. One algorithm can run faster to obtain a low fidelity odometry, and another one slower for a more precise matching. More recently, Implicit Moving Least Squares SLAM (IMLS-SLAM) [Deschaud, 2018] received attention for its high accuracy. IMLS-SLAM consists of scan-to-model matching framework. Initially, it uses an algorithm to sample the 3D scans and uses IMLS for surface reconstruction, which is claimed to have an improved matching quality.

On the other hand, features can be used in order to perform scan matching. There are different approaches [Aghamohammadi et al., 2007; Lingemann et al., 2005; Madhavan and Durrant-Whyte, 2004] that define characteristics in the scan data, such as maximum curvature points and discontinuities in the scanning, in order to extract landmarks that can be used posteriorly to algorithms such as the EKF-SLAM.

After the LIDAR raw data is processed, some algorithms can apply back-end methods to optimize the pose and the mapping of the environemnt. For example, [Hahnel et al., 2003] and [Grisettiyz et al., 2005] combine Rao-Blackwellized particle filters and scan matching. They proposed a method where the scan matching process is used to correct the odometry and this corrected path information is used as input for the sampling step in the Rao-Blackwellized particle filter. The details of this type of back-end methods are presented in the next subsection.

2. **Cameras**

SLAM approaches that use camera information are known as Visual SLAM. [Taketomi et al., 2017] defines three modules for the visual SLAM: initialization, tracking and mapping. To start, it is necessary to define a coordinate system for camera pose estimation and 3D reconstruction in an unknown environment. After it, tracking and mapping are performed to continuously estimate camera poses. To achieve this goal, 2D-3D correspondences between image and map are first obtained from feature matching or feature tracking in the image. Then, the camera pose is finally computed by solving the Perspective-n-Point (PnP) problem.

There are two main types of visual SLAM algorithms:

(a) Featured-based methods

In this type of approach an input image is abstracted to a group of features. Sequentially the features are tracked in order to estimate the state of the camera and the features. The idea of bringing vision into SLAM came from A.Davison's MonoSLAM [Davison, 2003; Davison and Murray, 2002; Davison et al., 2007], which is a feature-based monocular SLAM solution. The method consists in using image features to represent landmarks in the map, performing frame-to-frame matching to recover their 3-D positions and hence initialize feature-based sparse map. Finally, it uses the EKF framework to update the full state vector with the robot pose and the features locations. EKF based visual SLAM approaches have a problem of high computational effort as the map grows, since the computation grows quadratrically with the number of landmarks. The details of the EKF back-end SLAM are presented later in this section. Due to this problem, there is another group of methods that rely on the use of Bundle Adjustment (BA) instead of Bayesian filtering for pose and map refinement. PTAM [Klein and Murray, 2007] was the first SLAM solution of this kind. The method consists in parallelizing the motion estimation and mapping tasks and apply a key-frame based BA. Most of the modern visual SLAM systems are based on this approach because of its efficiency [Forster et al., 2014; Mur-Artal et al., 2015]. As a matter of classification, solutions that use the BA back-end can be defined as part of the Graph-Based SLAM classification, as keyframes and map points are treated as nodes in a graph.

(b) Direct methods

In contrast to feature-based methods, direct methods use an input image without any abstraction. In general, photometric consistency is used as an error measurement in those methods, just like geometric consistency of feature positions is used in feature-based methods. Earlier direct methods use planar patches to model the scene. [Silveira et al., 2008] presents such an approach for monocular cameras which simultaneously estimates the correspondences, camera pose, illumination changes and scene structure. More recent methods use per-pixel depth estimates. This is well established for RGB-D and stereo sensors [Kerl et al., 2013] [Comport et al., 2007]. For monocular cameras, these depth estimates need to be determined by stereo comparisons with previous frames. Newcombe et al. proposed a fully direct method [Newcombe et al., 2011] for monocular cameras called DTAM. In DTAM, the tracking is done by comparing the input image with synthetic view images generated from the reconstructed map. Currently one of the most famous approaches is called LSD-SLAM [Engel et al., 2014], where the camera is tracked using direct image aliment and geometry is estimated in the form of semi-dense depth maps. A pose-graph of keyframes is created, which allows to build large scale maps with loop-closure.

3.3.1.2 Back-end: Estimation Methods

The approaches to solve the back-end SLAM problem can be separated and presented along many different aspects. [Thrun and Leonard, 2008] defines three SLAM basic paradigms from which most other solutions are derived: Kalman Filter, Particle Filters, and Graph-based SLAM. These solutions define a generic back-end for different applications and sensors.

1. **Extended Kalman Filter (EKF)**

   The first solution to the SLAM problem is based on the Kalman Filters [Kalman et al., 1960], which consists of Bayes Filters whose distributions are Gaussians. In order to guarantee that every posterior probability is always a Gaussian, three assumptions are made. First, the motion and the observation model must be linear with added Gaussian noise. Second, the initial uncertainty must be Gaussian. And third, the process follows the Markov chain assumption, where the current state is conditionally independent of all earlier states given the immediately previous state.

   However, a robot moving in a real world taking measurements is governed by non linear models. For example, the sensor measurements usually are nonlinear with non-Gaussian noise. To deal with such nonlinearities, we use the Extended Kalman Filter, which approximate the robot motion and sensor models through a first order Taylor series expansion.

   This type of approach for SLAM was first introduced in [Smith and Cheeseman, 1986][Smith et al., 1990] and [Moutarlier and Chatila, 1989][Moutarlier and Chatila, 1990]. They propose a SLAM solution using a single state vector in order to estimate the location of the robot and the landmarks. The map is composed by a large vector stacking the sensors and landmarks states, and it is modeled by a Gaussian variable.

   Considering f as the function that models the vehicle's kinematics, and h the function that describes the geometry of the observation, the transition and observation models are defined as follows:

$$\begin{cases} x_t = f(x_t, u_t) + w_t \\ z_t = h(x_t, m_t) + v_t \end{cases} \tag{3.6}$$

   where w_t and v_t are the Gaussian noise with variances Q_t and R_t respectively.

   The map is maintained by the EKF through the process of prediction and update described above:

   - Prediction: the robot pose $\hat{x}$ and the predicted pose covariance P are predicted. F represents the Jacobian of f evaluated at the estimate $\hat{x}_{t-1|t-1}$. In this step the state m and its covariance do not change.

$$\hat{x}_{t|t-1} = f(\hat{x}_{t-1|t-1}, u_t), \tag{3.7}$$

$$P_{xx,t|t-1} = FP_{xx,t-1|t-1}F^T + Q_t, \tag{3.8}$$

- Update: the state vector $\hat{y}$ (robot pose and vector of map features) and the covariance are updated. H represents the Jacobian of h evaluated at $\hat{x}_{t|t-1}$ and $\hat{m}_{t-1}$.

$$\hat{y}_t = \begin{bmatrix} \hat{x}_{t|t-1} \\ \hat{m}_{t|t-1} \end{bmatrix} + K_t(z_t - h(\hat{x}_{t|t-1}, \hat{m}_{t-1})), \tag{3.9}$$

$$P_t = (I - K_t H)P_{t,t-1}, \tag{3.10}$$

  where

$$S_t = HP_{t|t-1}H^T + R_t, \tag{3.11}$$

$$K_t = P_{t|t-1}H^T S_t^{-1}. \tag{3.12}$$

This solution to the SLAM problem comes with some challenges and difficulties, such as consistency[1], filter convergence, data association and computational effort. Several papers [Castellanos et al., 2004; Frese, 2006; Julier and Uhlmann, 2001; Martinelli et al., 2005] have shown that the linearization process of the EKF-SLAM produce filter inconsistency and that convergence and consistency cannot be guaranteed in the non linear cases.

A possible solution to reduce the inconsistency error is another adaptation of the Kalman Filter, the Unscented Kalman filter (UKF) [Julier and Uhlmann, 1997]. This filter avoids linearizations through mean and covariance parameterization by particles chosen in a deterministic and geometrical way around the probability distribution. SLAM solutions were also presented using this type of filter [Andrade-Cetto et al., 2005; Martinez-Cantin and Castellanos, 2005; Wang et al., 2013].

The second challenge in EKF-based SLAM approaches is data association, which consists in the mapping between observations and landmarks. In the real world scenario, it is not possible to consider that we know the associations of all the observations. The standard approach to this problem is to assign every observation to a landmark using a maximum likelihood rule. However, since the EKF does not represent the uncertainty over data association, the effect of adding an observation from a wrong data association can never be undone. As a consequence, if a large number of incorrect observations are added into the EKF, the filter will diverge.

The concept of batch gating was an important advance in order to reduce the effect of wrong data association. It consists in simultaneously consider multiple associations exploiting the geometric relationship between landmarks. There

[1]Consistency can be understood as the quality of incertitude given by the filter in comparison to the real error. In order to measure the consistency, it is necessary to test the algorithm on simulation or to have a ground truth.

are two most used forms of batch gating: joint compatibility branch and bound method [Neira and Tardós, 2001], which is a tree-search method; and combined constraint data association method [Bailey, 2002], which is a graph search technique.

The computational complexity is another drawback of the EKF as a solution to the SLAM problem. In the original algorithm both the computation time and memory required for the method scale quadratically with the number of landmarks in the environment. In order to compute the SLAM posterior in a more efficient way, we can consider the sparsity aspect of the problem. For example, two landmarks that are far from each other are weakly correlated. A number of EKF-SLAM solutions came from this concept, where they exploit this by breaking the map into smaller submaps. In this approaches, for each submap a smaller EKF is applied and the computational time can be decreased [Guivant and Nebot, 2001; Leonard and Feder, 2000].

2. **Particle Filters**

A Particle Filter, also known as sequential Monte-Carlo (SMC) method [Metropolis and Ulam, 1949], is a Bayes filter that represents the probability distribution as a set of samples (particles). In the SLAM problem, each particle can be defined as a concrete guess as to what the true value of the state may be. Having a set of particles, the filter captures a representative sample from the posterior distribution. This approach was introduced into the SLAM literature in [Doucet et al., 2000], followed by [Montemerlo et al., 2002], where it got the name of FastSLAM. Its essential structure is a Rao-Blackwellised state, where the robot trajectory can be represented by weighted samples and the map is computed analytically. The main idea of the algorithm is based on the fact that conditionally to the trajectory, each landmark is independent of the others. For that reason, we can apply, for each particle, K Kalman filters to estimate K landmark locations conditioned on the path estimated [Montemerlo et al., 2002]. The method can be applied by performing the following four steps:

(a) Sampling: for each particle, compute a proposal distribution (π) and draw a sample of the next generation of robot poses from it.

$$x_t^{(i)} \sim \pi(x_t | z^{1:t}, u_{0:t}) \tag{3.13}$$

(b) Importance Weighting: to each particle assign an individual importance weight, according to:

$$w^{(i)} = \frac{p(x_t^{(i)} | z_{1:t}, u_{0:t})}{\pi(x_t^i | z_{1:t}, u_{0:t})} \tag{3.14}$$

(c) Resampling: particles with low importance weight are replaced by samples with a high rate.

(d) Map Estimation: for each particle, perform an EKF update on the observed landmarks as a simple mapping operation with known vehicle pose.

3. **Graph-Based SLAM**

The third family of back-end algorithms to the SLAM problem is known as Graph-Based or Network-Based. In this representation, the robot locations and landmarks are kept as nodes in a graph, joined by constrains that determines the spatial relationship between them.

Once the graph is constructed, the algorithm seeks to find the best configuration of robot's poses that satisfies the constrains. This is equivalent to minimizing a specified error function to obtain an optimal pose configuration.

Let $x = (x_1, ..., x_T)^T$ be the vector with the graph's nodes positions, $z_{i,j}$ be the registration algorithm transformation between nodes x_i and x_j, $\Omega_{i,j}$ be the information matrix (inverse of the covariance) of this transformation, and $\hat{z}_{i,j}$ be an estimate of registration transform received from initial configurations. The log-likelihood of the measurement $z_{i,j}$ is then defined as:

$$l_{i,j} = (\hat{z}_{i,j} - z_{i,j})^T \Omega_{i,j} (\hat{z}_{i,j} - z_{i,j}) \tag{3.15}$$

where $(\hat{z}_{i,j} - z_{i,j})$ is the difference between expected measurement and real measurement. Considering this, the goal of the maximum likelihood approach is to find the configuration of nodes that minimizes the negative log likelihood F(x) of all observations:

$$F(x) = \sum_{<i,j> \in G} (\hat{z}_{i,j} - z_{i,j})^T \Omega_{i,j} (\hat{z}_{i,j} - z_{i,j}). \tag{3.16}$$

The graph-based formulation for the SLAM problem was first introduced in [Lu and Milios, 1997]. However, the difficulty in solving the error minimization problem was a drawback to make this technique popular. Recent insights into the structure of the SLAM methods and advancements in the fields of sparse linear algebra resulted in efficient approaches to the optimization problem at hand [Grisetti et al., 2010]. In the last years several solutions for this problem were proposed. The use of sparse factorization in order to solve the linearized problem is a famous approach to the graph-based SLAM. This kind of method was first introduced in [Dellaert and Kaess, 2006] as a solution to the off-line SLAM. Similarly, Thrun et al. [Thrun and Montemerlo, 2006] introduced the GraphSLAM method as another solution for offline SLAM, where the graph is reduced using variable elimination techniques to have a lower-dimensional problem. In [Kaess et al., 2007] the iSAM approach was presented, which is an online solution to SLAM that exploits partial reorderings to compute the sparse factorization.

As mentioned before at the front-end Visual SLAM, a variation of the graph-based SLAM, denoted as Bundle Adjustment (BA) [Triggs et al., 1999], is frequently used on computer vision methods. In this type of approach the goal is to refine a visual reconstruction to produce jointly optimal 3D structure and camera pose.

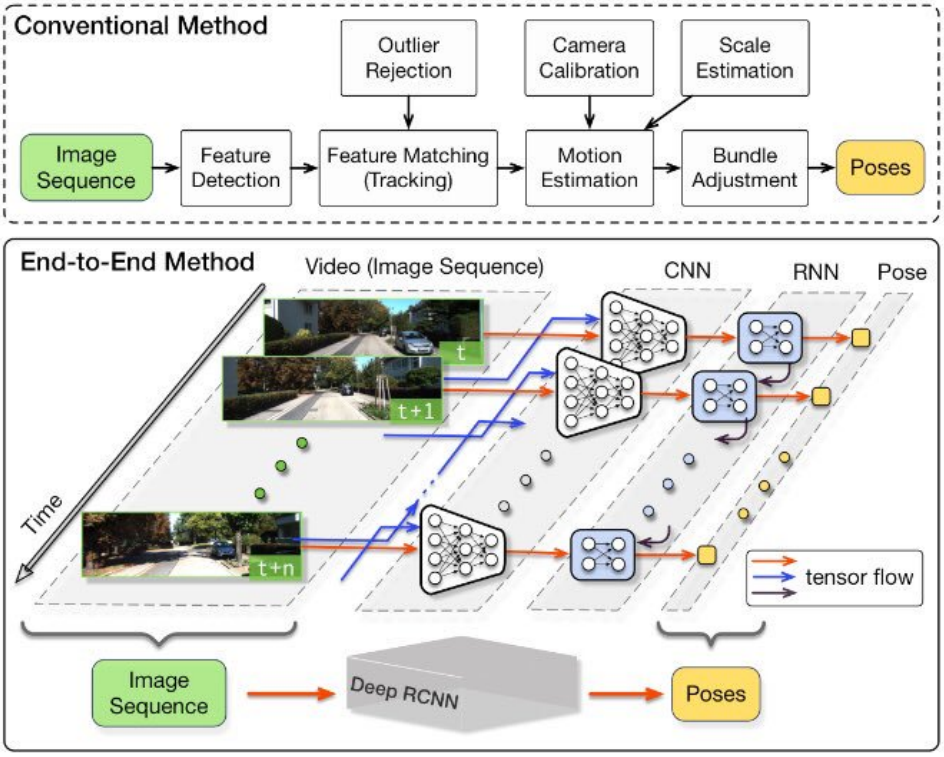


Figure 3.3: Comparison between a classic Visual SLAM algorithm and an end-to-end Deep Learning method. Image from [Wang et al., 2017].

3.3.2 Learning-based SLAM

Deep Learning is a hot topic in the computer vision, robotics and machine learning communities. This trend comes with the growth of computational power and the large amount of training datasets that became available in the last years. The main success of this type of approach is due to the fact that most of the parameters are learned from the data, using general-purpose learning procedures, that can facilitate a lot of different complex tasks.

Nowadays most of the best and competitive methods for computer vision methods, such as image classification, object detection and action recognition, are based on the use of deep learning techniques. The SLAM problem is still quite new in this field, however it has been receiving a lot of attention and several new papers using this paradigm are presented every year. In the previous section we presented how classical approaches need different blocks, front-end and back-end, performing several different tasks, such as feature detection and feature matching, to estimate the poses of a robot over time. In Figure 3.3 we can observe how deep learning techniques can replace all these blocks to one deep neural network that performs all these tasks; these methods are known as end-to-end SLAM. In this subsection, we present the most popular learning-based approaches up to this moment specifically for the SLAM problem, separating them by two types of sensors: cameras and laser scanners. The fundamentals of deep learning are presented later in chapter 6.

3.3.2.1 Deep Neural Networks for Visual SLAM

The objective of Visual SLAM using Deep Learning techniques is to learn camera motion model and estimates poses directly without explicitly model. The first solutions were based on the relocalization problem, since it is easier to collect labeled data in already visited places. Sequentially, new solutions for direct ego motion estimation were introduced by the use of supervised and unsupervised learning.

1. **Learning-based Camera Relocalization**

   PoseNet [Kendall et al., 2015] was the first method proposed to address the pose regression problem using Convolutional Neural Networks (CNNs). The network was trained with supervised learning, where the CNN used the ground truth poses of the camera to learn how to estimate the pose in pre-visited environments. After the training, the network could perform relocalization in those pre-visited places with more accuracy than classic model-based methods.

   Later, the same authors proposed the Bayesian PoseNet [Kendall and Cipolla, 2016] with the goal to estimate the uncertainty of the network pose estimation results. They also improved the results in a new paper [Kendall and Cipolla, 2017], where they explored new loss functions based on geometry and scene reprojection error. Another extension was also proposed by [Li et al., 2017b] to use the same type of method with RBD-D cameras. More recently, Bresson et al. [2019] propose to use images from Google Street View to help train a network based on PoseNet. Their experiments show that augmenting the dataset can considerably improve the accuracy of the network.

   Recurrent Neural Networks (RNNs) are commonly used for tasks where the temporal dynamics can be useful for the learning. Walch et al. [2017] proposed to add a Long Short-Term Memory (LSTM) module into PoseNet to learn the temporal dynamics of the camera motion and they were able to improve the robustness of the system. Clark et al. [2017] also proposed to use RNNs to implement a network for pose estimation using video clips. By considering the image sequences in the network input, the pose estimation was more accurate, improving the relocalization of the camera.

2. **Learning-based Visual Odometry Estimation**

   Wang et al. [2017] proposed the method called DeepVO, which is a monocular visual odometry system composed by a Recurrent Convolutional Neural Network (RCNN) to estimate the camera motion. The same authors also presented the method UndeepVO [Li et al., 2018], which proposes an unsupervised deep learning method to estimate the pose of a monocular camera. Later, Costante and Ciarfuglia [2018] presented a new network for odometry estimation called LS-VO. It first learns the optical flow representation, followed by a pose estimation network. All these networks have the advantage of learning effective and robust features in different situations, however they still suffer from different problems, accumulating drift, making the classic VO methods still outperform these methods.

3.3.2.2 Learning-based LIDAR SLAM

The application of deep learning techniques for odometry estimation using laser scanners is still considered as a new challenge and only few papers have addressed it. Nicolai et al. [2016] were the first to propose to apply 3D laser scanner data in CNNs to estimate odometry. Their approach provides a reasonable estimation of odometry, however still not competitive with the efficiency of state-of-the-art scan matching methods. Later, Velas et al. [2018] presented another approach for using CNNs with 3D laser scanners for IMU assisted odometry. Their results were able to get high precision and close results compared to state-of-the-art methods, such as LOAM [Zhang and Singh, 2014], for translation, however the method is not able to estimate rotation with sufficient precision. Considering their results, the authors propose that their method could be used as a translation estimator and use together an Inertial Measurement Unit (IMU) to obtain the rotation. Another drawback is that, according to the KITTI benchmark, even using CNNs the method is slower than LOAM. Up to the proposed work of this thesis, Li et al. [2017a] were only authors to use this kind of approach to estimate odometry using only 2D laser scanners. They propose a network to perform scan matching and loop closure using CNNs. Although the loop closure networks shows good accuracy, the scan matching results are still very inaccurate compared to classic methods.

3.4 Conclusion

This chapter presented the literature review of the localization and mapping problem. We exposed the most popular and competitive methods from classical SLAM approaches to Deep Learning approaches. We can conclude from this review that the SLAM problem is still challenging depending on the environment it is applied to and the type of sensors we use. While classical approaches have the advantage of modelling directly the problem, allowing to understand the errors and accuracy of the method, it still suffers from challenging environments and a lot of times from long processing time. At the same time, deep learning methods have the disadvantage of the lack of current error information in the estimation results. However, they are highly adapted to different environments and usually can run easily in real-time using GPUs. The work presented in this thesis is influenced by this context, where we explore the from classical SLAM approaches to Deep Learning methods. Our goal is to understand the interest of these two types of paradigms, creating new solutions that can improve the mapping and localization problem for autonomous vehicles.

Considering this, the proposed work of this thesis is separated in two parts: Evidential based SLAM (classical methods) and Deep Learning Localization (learning-based methods). We first explore the use of classical approaches to increase the understanding of the environment of an autonomous vehicle using two low cost sensors, and to create a new localization solution that takes into consideration the dynamic difficulties of the road scenario. Sequentially, we create new solutions to the localization problem based on the use of neural networks using two types of sensors: monocameras and 2D laser scanners.

Part II

Evidential-based SLAM

This part presents new mapping and localization solutions based on classical approaches, such as occupancy grid maps and the Evidential theory. In the first chapter, we overview the theory of occupancy grid maps and how the evidential model can be applied to it. Sequentially, we present a novel sensor fusion method to fuse 2D laser scanners and stereo cameras in order to increase the perception of the environment. In the second chapter, we introduce a localization method that explores the advantages of using evidential maps along with image registration techniques.

Chapter 4

Sensor Fusion in Evidential Grid Maps

Contents

4.1 Introduction 50

4.2 Preliminaries: Occupancy Grid Maps 51

4.2.1 Bayesian Framework 52

4.2.2 Dempster-Shafer Framework 53

4.3 Proposed Approach 56

4.3.1 Laser Scanner Model 56

4.3.2 Sensor model for stereo camera 57

4.3.3 Temporal Grid Fusion 61

4.3.4 Life-Long Grid 62

4.4 Experimental Results 63

4.4.1 Multi-sensor local fusion evaluation 63

4.4.2 Temporal fusion evaluation 65

4.4.3 Quantitative results 66

4.5 Conclusion 67

Résumé du chapitre 4

Les données de plusieurs capteurs doivent être fusionnées pour améliorer la perception de l'environnement d'un véhicule autonome. Pour cela, les grilles d'occupation sont souvent utilisées pour représenter les données du capteur et son incertitude. La cartographie dans ces grilles est effectuée en stockant dans chaque cellule la probabilité qu'une zone soit occupée ou libre. Dans la première section de ce chapitre, nous introduisons la théorie de cartographie à base d'une grille d'occupation qui servira de contexte pour mieux comprendre l'approche proposée. Le reste du chapitre présente une nouvelle méthode qui utilise une représentation enrichie du monde basée sur la fusion de capteurs. Nous présentons un algorithme pour créer une représentation en grille evidentielle à partir de deux capteurs très différents, un scanner laser et une caméra stéréo, qui permet une meilleure gestion des aspects dynamiques de l'environnement urbain ainsi qu'une bonne gestion des erreurs pour créer une carte plus fiable, et ainsi une localisation plus précise.

4.1 Introduction

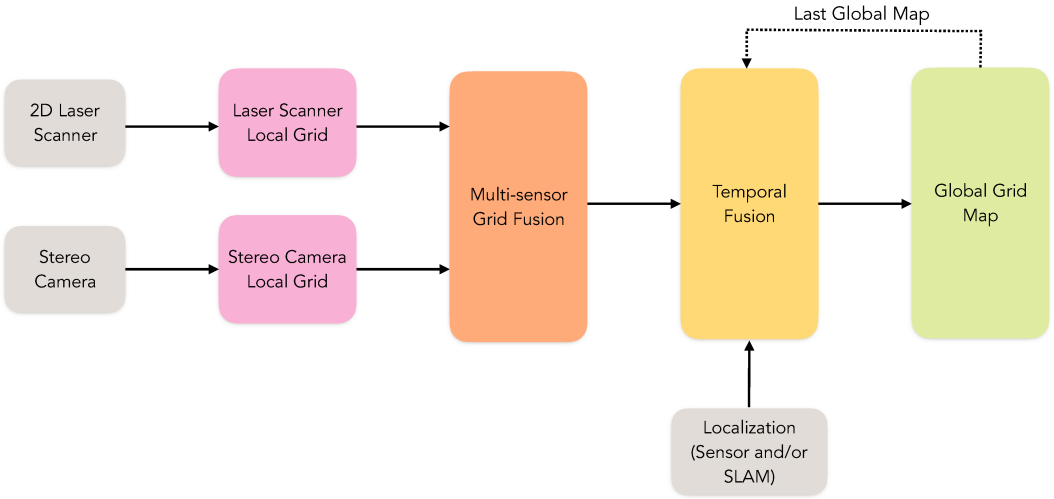


Figure 4.1: Schematics of the proposed system.

As presented in chapter 2, there is a large number of sensors available for intelligent vehicles to perceive the environment (e.g. laser scanners, sonars, stereo and mono cameras), and each sensor has its own particular limitations and advantages, in addition to different uncertainty characteristics. Due to its limitations, a single sensor cannot provide alone a robust reconstruction of the surroundings and hence cannot reliably perform important tasks like obstacle avoidance, path planning and localization.

Considering this, the data from multiple sensors must be merged to enhance the accuracy of the environment perception. This type of fusion can be performed at different levels of representation and by the use of different fusion methods. Occupancy grids are often applied as a representation of the sensor data and its uncertainty, by mapping the obstacles detected by sensors. The mapping is performed by storing the probability of an area being occupied or free in each cell of the grid. In the first section of this chapter we introduce the theory for occupancy grid mapping that will serve as background to better understand our proposed approach.

The remaining of the chapter introduces a new framework that uses an enriched representation of the world based on sensor fusion. We present a framework to create an Evidential grid representation from two very different sensors, laser scanner and stereo camera, that allows a better handling of the dynamic aspects of the urban environment and a proper management of errors to create a more reliable map, thus having a more precise localization. Moreover, a life-long layer with high level states is presented, it maintains a global map of the entire vehicle's trajectory and distinguishes between static and dynamic obstacles. Figure 4.1 presents the architecture of the proposed framework. Results on a real road dataset show that the environment mapping data can be improved by adding relevant information that could be missed without the proposed approach.

4.2 Preliminaries: Occupancy Grid Maps

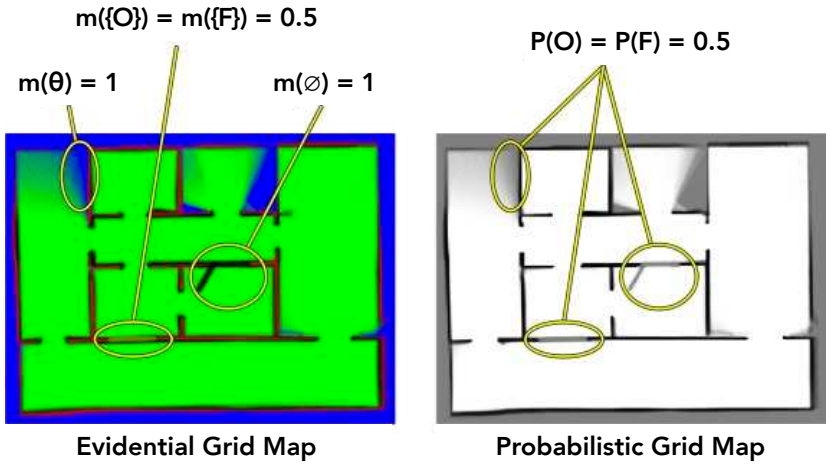


Figure 4.2: Comparison between the Evidential and the Bayesian approaches for occupancy grid maps. F, O, $\emptyset$ and Θ represent the free, occupied, conflict and unknown states respectively. We can observe how in the probabilistic grid map the lack of information and the conflict information end up having the same representation ($P(O) = P(F) = 0.5$), while in the evidential grid map we can analyze them separately ($m(\Theta) = 1$, $m(O) = m(F) = 0.5$ and $m(\emptyset) = 1$). Image from [Clemens et al., 2016].

Perceiving the environment is a complex task and how and where to store what the sensors are able to detect is a critical decision for any mobile robotic system. Different map representation have been proposed over the years, and choosing the right one depends on the sensor information we want to store, what kind of environment we are exploring and finally what kind of tasks we want to perform.

Elfes [1989] introduced the occupancy grid framework that became one of the most used approaches to environment mapping using a variety of sensors. An occupancy grid consists of a 2D map representation where each cell in the grid receives a value that corresponds to the state of occupancy of the cell. The grid framework is in charge of inferring the state of a cell at each sensor measurement, taking into consideration each sensor model and its uncertainty.

There exist different inference framework theories that can be used to update the state of a cell when a new sensor information is available. We first introduce in this section the most common framework for occupancy grids: the classical Bayesian inference theory; and sequentially we introduce our chosen framework, the Dempster-Shafer theory of evidence. Figure 4.2 illustrates the two different frameworks, where we can already observe how different situations are defined in the Bayesian framework (probabilistic occupancy grid maps) in the same form, not allowing us to distinguish them, while using the theory of evidence (evidential grid maps) we can make this distinction. In complex environments like the one of an autonomous vehicle the distiction of these states can be crucial, and for this reason we chose to apply this framework to our proposed approach.

4.2.1 Bayesian Framework

In the Bayesian (probabilistic) grid maps we use the Bayesian inference theory to estimate the probability of occupation of each cell in the grid. This theory is based on the Bayes theorem [Bayes, 1991]:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{4.1}$$

Elfes [1989] proposed to use for the first time this theory with grid maps, where the cells receive a probability of occupation. The formulation of the probabilistic occupancy grid mapping problem consists on computing the posterior probability of the map, given all the sensor measurements $z_{1:t}$ until time t. The most common case assumes that the grid map g is static and is updated over time when new sensor measurements arrive. Therefore, considering the grid map cell g_{ij} we estimate the probability of occupation $P(O \mid z_{1:t})$ using Bayes theorem:

$$P(O \mid z_t, z_{1:t-1}) = \frac{P(z_t \mid O, z_{1:t-1}) P(O \mid z_{1:t-1})}{P(z_t \mid z_{1:t-1})} \tag{4.2}$$

where $P(z_t \mid O, z_{1:t-1})$ is known as the forward sensor model and represents the probability of obtaining a sensor measurement considering the current status of the cell. Using this model we can also define the probability of a cell being free $P(F)$, since $P(O) + P(F) = 1$. We also consider that all states are initialized with $P(O) = 0.5$.

The main goal of the occupancy grid usually is to determine if a cell has an obstacle or not. Therefore, for this framework the following decision rule is generally applied to find the state of a cell s_{ij}:

$$s_{ij} = \begin{cases} \text{free} & \text{for } P_{g_{ij}}(O) < 0.5 - thr \\ \text{occupied} & \text{for } P_{g_{ij}}(O) > 0.5 - thr \\ \text{unknown} & \text{for } \mid P_{g_{ij}}(O) - 0.5 \mid \leq thr \end{cases} \tag{4.3}$$

where thr is a small threshold around 0.5 that allows the states to be classified as empty or occupied only if the probability is high enough, helping remove false positives. Moreover, because the probability of the unknown cells will rarely be exactly 0.5, we need to use this threshold to define the unknown state.

As we can observe in Figure 4.2, the main problem of the probabilistic decision rule is the lack of definition to differentiate the states that are unknown and states that have conflicting information, like being detected as free and occupied in different cases. This drawback can be even more significant when we have dynamic environments, like roads with pedestrians and other vehicles, where it is important to know where there is an absence of information and where there is actually just a conflicting information, probably caused by a moving obstacle.

4.2.2 Dempster-Shafer Framework

The Dempster-Shafer theory of the evidence proposed by Shafer [1976] solves the difficulty of the probabilistic model to deal with conflicting information and its lack of representation for the absence of information. For this reason, this framework has received attention in the field of autonomous vehicles [Moras et al., 2011][Moras et al., 2014], improving the mapping of dynamic environment. We will present in this subsection some basic definitions of this framework that will allow later a better understanding of the evidential occupancy grids, which will be used in our proposed method.

In the probabilistic model the state of a grid cell is modeled with a single probability $P(O)$. As we mentioned before, the problem of this approach is that different states of belief (like unknown and conflict information) are mapped to the same probability value. However when we use the Dempster-Shafer theory we can represent the uncertainty of one cell in multiple dimensions. These multiple dimensions in this framework are called the frame of discernment (FOD), which is a finite set of mutually exclusive and exhaustive propositions X:

$$\Theta = \{X_1, X_2, ..., X_K\} \tag{4.4}$$

Any subset of the frame of discernment can receive a basic probability assignment (BPA) function, also known as mass function, which is basically the equivalent to the probability density function in the probabilistic framework. In our case of evidential grid maps, the BPA represents the probability of occupancy of the grid map cells. Considering this, the mass function m can be defined as follow:

$$m : 2^{\Theta} \rightarrow [0, 1] \tag{4.5}$$

where 2^{θ} is the set of all possible subsets of the field of discernment Θ. For every subset $A \in 2^{\Theta}$, the mass function follows these conditions:

$$\begin{aligned} \sum_{A \in 2^{\theta}} m(A) &= 1 \\ m(\emptyset) &= 0 \end{aligned} \tag{4.6}$$

Considering this, the frame of discernment, Θ and its power set 2^{Θ} can be defined as followed for our specific case of evidential occupancy grid maps:

$$\begin{aligned} \Theta &= \{F, O\} \\ 2^{\Theta} &= \{\emptyset, F, O, \{F, O\}\} \end{aligned} \tag{4.7}$$

where $\emptyset$ is the null set, that in our case represents the conflict information, and $\{F, O\}$ the unknown set. As you can see, this representation allows us to express that a cell might have an unknown state, with a certain probability, which was not possible using only the Bayesian model.

In order to combine different source of information using the Dempster-Shafer theory several rules of combination can be used. In the case of evidential grid maps these operators are necessary for us to update the grid when new information arrives, or even to fusion it with different type of information. Each operator has its particular characteristics, which can make it a better fit for specific applications. We will summarize the most commonly used operators that we can apply to the evidential occupancy grid maps:

1. Conjunctive Combination
   The conjunctive combination rule allows to combine two independent sources of information that comes from the same frame of discernment. This rule can be applied when we have two reliable sources of information. The following equation is used to perform this combination:

$$\forall A \in 2^{\theta} m_1 \textcircled{\cap} m_2(A) = \sum_{B \cap C = A} m_1(B) \cdot m_2(C) \tag{4.8}$$

2. Disjunctive Combination
   Similarly to the previous rule, the disjunctive combination rule combines two independent sources of information that comes from the same frame of discernment. However, in this case one of the sources is non reliable, without knowing which one. The following equation is used to perform this combination:

$$\forall A \in 2^{\theta} m_1 \textcircled{\cup} m_2(A) = \sum_{B \cup C = A} m_1(B) \cdot m_2(C) \tag{4.9}$$

3. Dempster Combination
   Dempster [1968] introduced a new operator known as Dempster Combination rule, which is the most common operator used in the Dempster-Shafer theory. The combination is performed in two steps, first the conjunctive rule is applied and sequentially the conflict mass $\emptyset$ is normalized:

$$\begin{aligned} m_{1\oplus 2}(A) &= \frac{m_{1\textcircled{\cap}2}(A)}{1 - m_{1\textcircled{\cap}2}(\emptyset)} \\ m_{1\oplus 2}(\emptyset) &= 0 \end{aligned} \tag{4.10}$$

In appendix A we define these equations for the specific case of evidential grid maps with $\Theta = \{F, O\}$. The Table 4.1 shows how these combination rules work for this specific case. In the first example we can observe the results of each combination when a new information arrives in an unexplored region. The new information is not added to the result when the disjunctive combination rule is applied, while in the other two combinations it is. This happens because the disjunctive operator considers that one of the sources is not reliable.

The second example shows how conflict information is differently addressed by the three combination rules. We can observe that the conflict mass is generated when the

Assignment			Fusion		
	m_1	m_2	$m_1 \textcircled{\cap} m_2$	$m_1 \textcircled{\cup} m_2$	$m_{1\oplus 2}$
$\emptyset$	0.0	0.0	0.0	0.0	0.0
F	0.5	0.0	0.5	0.0	0.5
O	0.0	0.0	0.0	0.0	0.0
Θ	0.5	1.0	0.5	1.0	0.5

(a) Example when new information arrives in an unexplored cell

Assignment			Fusion		
	m_1	m_2	$m_1 \textcircled{\cap} m_2$	$m_1 \textcircled{\cup} m_2$	$m_{1\oplus 2}$
$\emptyset$	0.0	0.0	0.64	0.0	0.0
F	0.8	0.0	0.16	0.0	0.44
O	0.0	0.8	0.16	0.0	0.44
Θ	0.2	0.2	0.04	1.0	0.12

(b) Example when there is conflicting information in a cell.

Table 4.1: Fusion of two cells using different combination rules.

conjunctive operator is used, and it can be normalized using the dempster combination. For the same reason as in the first example, the information is not considered when the disjunctive operator is used and there is only evidence of unknown in the result.

There are also two concepts commonly used from the Dempster-Shafer theory in order to analyze an evidential grid map: the belief and plausability. The belief represents the degree to which one proposition X, a subset of 2^θ, is believed to be true and it can be calculated using the following formula:

$$Bel(X) = \sum_{A \subseteq X} m(A) \tag{4.11}$$

The plausability has a similar definition, it represents the degree to which one proposition is believed not to be false:

$$Pl(X) = 1 - \sum_{A \cap X = \emptyset; A \in 2^\theta} m(A) \tag{4.12}$$

These functions define the upper and lower bounds of the probability of X:

$$(Bel)(X) \leq P(X) \leq Pl(x) \tag{4.13}$$

The concepts presented in this subsection make the theory of evidence a powerful tool to represent the environment using evidential grid maps. Its robust combination methods allows the grid maps to be able to manage both noisy and conflicting information. Moreover, it is a useful tool to combine different sensor information with distinct uncertainty parameters. For these reasons, we chose the evidential framework for our proposed method, that will make use of these characteristics to enhance the mapping for an autonomous vehicle.

4.3 Proposed Approach

In this work we have developed both a more robust representation of occupation (i.e. presence/absence) and an architecture that focuses on the data fusion process of different sensors to obtain this representation of the current and previous traversed driving environment. The proposed approach applies the Evidential framework to merge occupancy grids created by two different sensors: a 2D laser scanner and a stereo camera. It consists in computing two evidential grid maps, one for the laser scanner and one for the stereo camera, and fuse them to have a common representation of the environment. Subsequently, we perform temporal fusion over the new grid and the stored grid to generate a global map. Finally, the output of the system is a global map that represents the environment where the vehicle drove. The architecture of the whole system is presented in Figure 4.1 (page 50).

The creation of a grid is performed by interpreting the raw data of the sensors to metric information, which will represent the presence or absence of obstacles on that location. Moreover, to have a reliable map, we need to model the uncertainty of the sensor into the grid cells. We present a sensor model for a 2D laser scanner and for a stereo camera, that will deal with the sensor uncertainty to create the grid by using Dempster-Shafer theory of evidence [Shafer, 1976]. The evidential grid for the two sensor models has as a frame of discernment $\Omega = \{O, F\}$, referred as the states occupied (O) and free (F) of each cell. Therefore, the power set is defined as $2^{\omega} = \{\emptyset, F, O, \Omega\}$, and each cell will store a basic belief assignment (BBA) with four beliefs $[m(F), m(O), m(\Omega), m(\emptyset)]$. Each belief represents respectively the evidence of being free, occupied, unknown or conflict.

4.3.1 Laser Scanner Model

Laser data gives reliable information about free and occupied space in the environment. At each time step, the sensor provides a scan, which corresponds to a set of points measured during one laser rotation. Although this process is not instantaneous, we assume it is fast enough to map all points at the same time on the grid. We apply the solution proposed in [Moras et al., 2011] to determine the evidence of a cell in a grid map by increasing the occupied evidence where there are laser impacts and the free evidence on the crossed cells.

The proposed method defines the mass of each cell in the laser scanner grid LG at timestamp t as follow:

$$\begin{aligned} m_{LG,t}(A) &= \lambda & m_{LG,t}(B) &= 0 \\ m_{LG,t}(\emptyset) &= 0 & m_{LG,t}(\Omega) &= 1 - \lambda \\ with\ A = \begin{cases} O\ if\ cell\ impacted \\ F\ if\ cell\ crossed \end{cases} & & B = \begin{cases} F\ if\ cell\ impacted \\ O\ if\ cell\ crossed \end{cases} & \end{aligned} \tag{4.14}$$

where λ represents the laser scanner confidence.

The objective of this model is to attribute a belief mass in the occupied state to the detected obstacles by the sensor; and to attribute a mass in the free space to the region crossed by the laser scan without detecting any obstacle. At the same time, all the sensor's incertitude is added to the unknown state.

4.3.2 Sensor model for stereo camera

The second sensor used in our system is a binocular stereo-vision sensor. Our goal is not to estimate the free space with this sensor (that is detected with more accuracy by the laser scanner), but rather to improve the detection of obstacles that could be challenging for the laser scanner. Although 2D laser scanner are reliable and accurate, certain obstacles can be missed because of its height limitations, for example a door of a truck opened or a high gate. In addition, once one obstacle is detected it cannot detect anything further. Considering this, we define a stereo camera model that is based on the disparity space for obstacle detection.

After receiving the two raw stereo images as input, we perform image rectification to bring the two images to a common image plane, which simplifies image matching methods. Once this is executed, the disparity map between the images is computed and the method for obstacle detection and mapping described below is performed.

1) V-disparity map and ground estimation: the first step is to generate the V-disparity map by accumulating the pixels with the same disparity along the rows of the disparity image. As presented in [Labayrade et al., 2002], the V-disparity space can provide a representation of the geometric structure of the road and, for this reason, it can be used to estimate the ground plane pixels. In Figure 4.3 we can observe the v-disparity map in the bottom right.

2) Obstacles detection: once we have defined the ground plane pixels, one can classify the obstacles pixels by separating the pixels that are over the ground and thresholding the height of the pixels. Sequentially, the obstacles pixels are mapped to the U-disparity space [Hu et al., 2005]. Instead of projecting the pixels related to the rows of the images, like in the V-disparity space, now we project them related to the columns. Figure 4.3 shows the separation of the obstacles pixels along with the V-disparity and U-disparity maps.

3) Obstacle projection: the next step is to project the obstacles pixels to the world. First, we select on the U-disparity map the cells are over a threshold that eliminates false obstacle detections. Then, we adopt the method detailed in [Perrollaz et al., 2010b] to project these cells to the world. The stereo coordinate system has as origin the point O_S, which is the middle point of the baseline (Figure 4.4). The detection plane P_D (supporting the grid) coordinate system has as origin the point O_D, the projection of O_S on the plane. Considering U_D the coordinates of a point in the U-disparity map (u, d) and X_D the 2D point related to the camera coordinate system (x, y) on the detection plane, the transformation between U_D and X_D is performed by the function G_D:

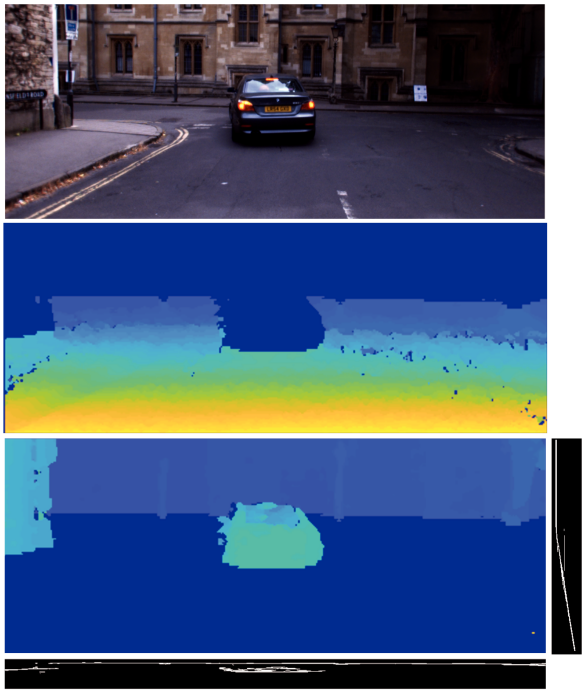


Figure 4.3: Top: original image from one of the cameras. Middle: ground disparity map. Bottom: obstacle disparity map with the V-disparity map on the right and the U-disparity map on bottom.

$$\begin{aligned} G_D : \mathbb{R}^2 &\rightarrow \mathbb{R}^2 \\ U_D &\mapsto X_D \end{aligned} \tag{4.15}$$

with

$$\begin{cases} x = \frac{b}{2} + \frac{(u-u_0)\cdot b}{d} \\ y = \frac{f\cdot b}{d} \end{cases} \tag{4.16}$$

where b is the stereo camera baseline, $(u_0,\ v_0)$ the principal point and f the focal length (Figure 4.4).

4) Evidential mapping: in the last step the points previously acquired are projected onto the 2D evidential occupancy grid. As a consequence of the errors in the (u, d) coordinates, one pixel can have multiple observations in the grid. For this reason, we apply a Gaussian observation model centered on U_D; in addition, to transform directly the pixel to the grid cell, we linearize G_D around the observed pixel. We base our model on the work of Perrollaz et al. [2010a], modifying it to consider only the (u, d) coordinates. This modification allows to map the obstacle points straight from the U-disparity map to the occupancy grid, which can reduce the computation time of the transformation. Considering U_i a point from the U-disparity map and c_k a cell affected by this point, we define a projection factor $f(U_i, c_k)$. The factor is calculated under the Gaussian measurement model with linearization on the G_d function as:

$$f(U_i, c_k) \propto \exp(-\frac{1}{2}\|c_k - \mu_D^m\|^2_{K_D^m}) \tag{4.17}$$

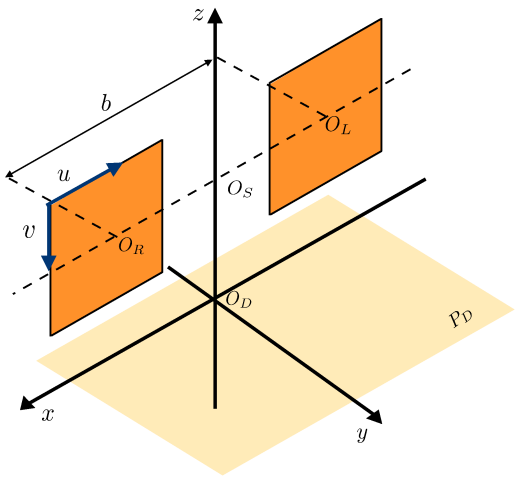


Figure 4.4: Geometrical configuration of the stereo camera's coordinate system.

where the Mahalanobis norm is defined by $\|v\|_A^2 = v^T A^{-1} v$ and the mean and covariance are given by:

$$\begin{cases} \mu_D^m = G_D(U_i) \\ K_D^m = J_G(U) \cdot \begin{bmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_d^2 \end{bmatrix} \cdot J_G^T(U) \end{cases} \tag{4.18}$$

J_G being the Jacobian matrix of the function G_D, while σ_u and σ_d are related to the errors produced by the stereo matching method.

Since multiple obstacle points can be mapped to the same cell on the occupancy grid, we propose to apply an accumulation strategy to define the basic belief assignment of each cell. We define the total contribution of a cell as the sum of contributions of the n points that are projected onto it:

$$C_{TOTAL}(c_k) = \sum_{i=1}^{n} f(U_i, c_k) \cdot I_U^{obst}(U_i) \tag{4.19}$$

where $I_U^{obst}(U_i)$ is the obstacle u-disparity map that represents the accumulation of obstacles pixels in the U-disparity space.

In order to avoid that taller objects have higher occupied evidence compared to shorter obstacles, we add an activation function based on the pixels contribution of each cell. In this way, the obstacle evidence grows quickly with respect to the number of obstacle pixels in that location with a threshold achieving the maximum stereo camera obstacle evidence. Considering this, the BBA of a cell k in the stereo camera grid SG at time slot t is filled as follow:

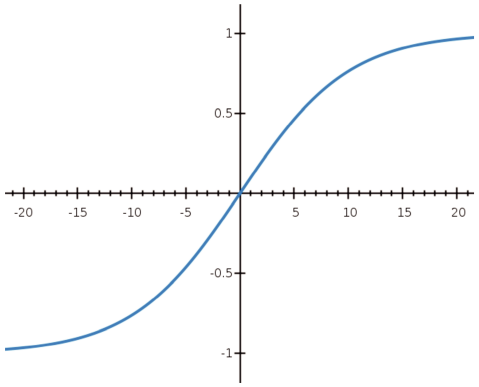


Figure 4.5: Plot of the activation function $\tanh(\alpha_a C_{TOTAL}(c_k))$ when $\alpha_a = 0.1$.

$$\begin{aligned} m_{SG,t}(F) &= 0 \\ m_{SG,t}(O) &= \tanh(\alpha_a C_{TOTAL}(c_k)) = \frac{2}{1+\exp(-2\alpha_a C_{TOTAL}(c_k))} - 1 \\ m_{SG,t}(\emptyset) &= 0 \\ m_{SG,t}(\Omega) &= 1 - \tanh(\alpha_a C_{TOTAL}(c_k)) \end{aligned} \tag{4.20}$$

where α_a is a scale factor defined empirically. Figure 4.5 shows the plot of the activation function when $\alpha_a = 0.1$.

4.3.2.1 Multi-sensor grid fusion

To have a common representation of the environment, we create a combined map that merges the information from the two sensors grids. In order to perform the fusion, the grids need to be in the same coordinates. For this reason, each measurement has to be mapped to a joint reference system. This can be achieved using the results from the external calibration method between the stereo camera and the laser scanner. For this application, we consider that both sensors are already calibrated and these values are known. Figure 4.6 first step illustrates this process.

After we have obtained a common coordinate system, the fusion is executed in two steps. First, we apply a factor that models the *a priori* knowledge related to the stereo camera sensor, in which the obstacle depth confidence decreases proportionally to the distance from the sensor. This reduces the influence of far obstacles detected by this sensor, and increases the influence of laser data in these cases. Considering this, a factor α_c is used to discount the probability masses of the stereo camera grid SG at timestamp t as follow:

$$m^{\alpha_c}_{SG,t}(A) = \begin{cases} \alpha_c \cdot m_{SG,t}(A) & \forall A \subsetneq \Omega \\ 1 - \sum\limits_{B \subseteq \Omega} m^{\alpha_c}_{SG,t}(B) & A = \Omega \end{cases} \tag{4.21}$$

where α_c is inversely proportional to the euclidean distance between the sensor location and the cell.

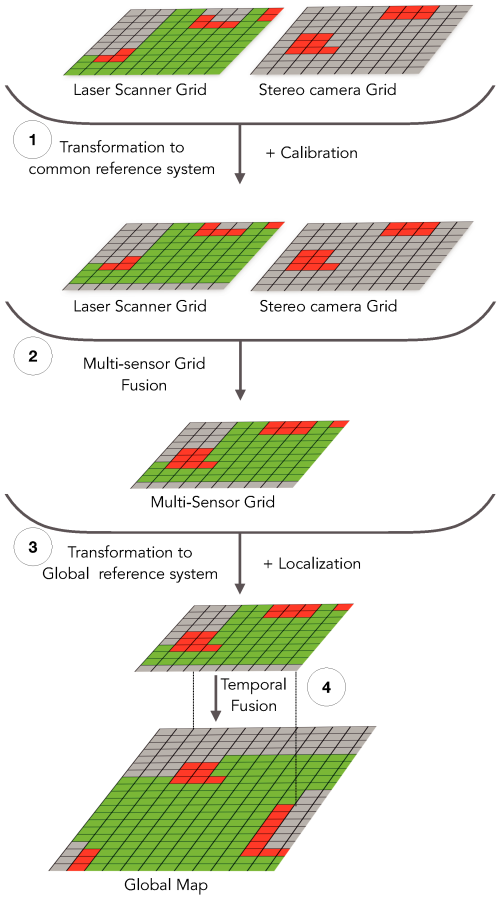


Figure 4.6: Illustration of the multi-sensor and temporal fusion steps.

Sequentially, the laser scanner grid LG and the stereo camera grid SG are combined by applying the Dempster's rule (Equation 4.10) generating a fused grid FG. We choose to use this combination rule because the two sources of information can be considered reliable, independent and have the same frame of discernment.

4.3.3 Temporal Grid Fusion

After merging the two sensors grids, we need to accumulate the information gathered at different timestamps by applying a fusion process as illustrated by Figure 4.6. Like in the multi-sensor grid fusion, the map from the previously time slot and the new one need to be in the same coordinate system. For this purpose, the vehicle position and orientation at the current time slot have to be estimated. This can be performed by a Simultaneous Localization and Mapping (SLAM) method or by using information coming from sensors, such as GPS and IMU. This process is out of scope of this chapter and we consider that this information is already provided.

After the new local fused grid FG is transformed to the global grid coordinates, we merge it to the previous global grid GG by applying the Dempster's combination rule previously described:

$$m_{GG,t} = m_{GG,t-1} \oplus m_{FG,t} \tag{4.22}$$

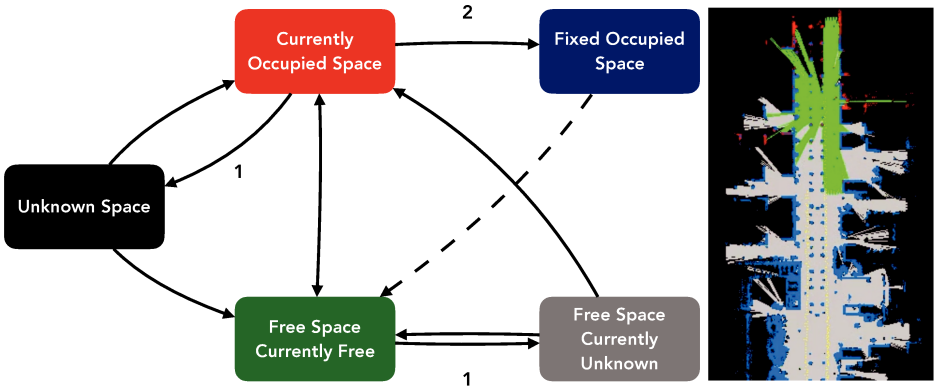


Figure 4.7: Diagram illustrating how the states of the life-long grid can change. 1 represents the timeout parameters and 2 the accumulator. The lines represent how the states can be updated considering the new information in the fusion grid. The dash line represent a change in state that can happen in case a cell was wrongly classified as FO.

4.3.4 Life-Long Grid

Previous methods for evidential grid mapping [Kurdej et al., 2015; Moras et al., 2014] choose to apply a time decay factor that decreases the belief of the cells during the temporal grid fusion. Over time, this factor erases mapped information and keeps only a recent map. In order to not lose the global map, we introduce a new life-long layer that provides more information than just the evidence of being free, occupied, uncertain or unknown. The new layer has five new different possible states:

- Free space currently free (CF): free space which is considered free at that moment.
- Free space currently unknown (CU): free space that it is not possible to know if it is still free anymore.
- Fixed occupied space (FO): occupied space that is considered a static obstacle.
- Currently occupied space (CO): occupied space that is a dynamic obstacle or it was not detected long enough to be defined as static.
- Unknown (U): no state information about the cell.

These states generate a semantical interpretation of the metric information provided by the evidential global grid. The current state of a cell is determined by analyzing the changes in the belief masses and by using two parameters: an accumulator and a timeout. First, every cell is assigned as unknown and when the evidence of free or occupied space increases they can be classified as CF or CO. After a number of time slots, the timeout parameter establishes if a CF cell state information is not recent anymore and should change to state CU. Another application of the timeout occurs when no new information arrives at a cell with state CO and the state needs to be changed to U. The second parameter, the accumulator, is used to determine if an obstacle can be considered fixed or not. It accumulates the number of time slots that

this cell was classified as CO. After it arrives to a threshold, the cell state changes to FO. An important aspect is that the values assigned to the two parameters vary linearly according to the vehicle's velocity. The states behavior is summarized in Figure 4.7.

The introduced life-long layer permits us to obtain a global map with accurate information without loosing the previous mapped regions. In other words, the states of the grid give us extra information that shows that a cell was not recently updated and it is not necessary to erase past information. Furthermore, the new layer defines the fixed obstacles in the map. The removal of dynamic information is crucial to localization algorithms, once they should only use fixed landmarks as reference. This knowledge can improve the localization coming from the sensors or even to allow the mobile robot to not rely on them anymore.

4.4 Experimental Results

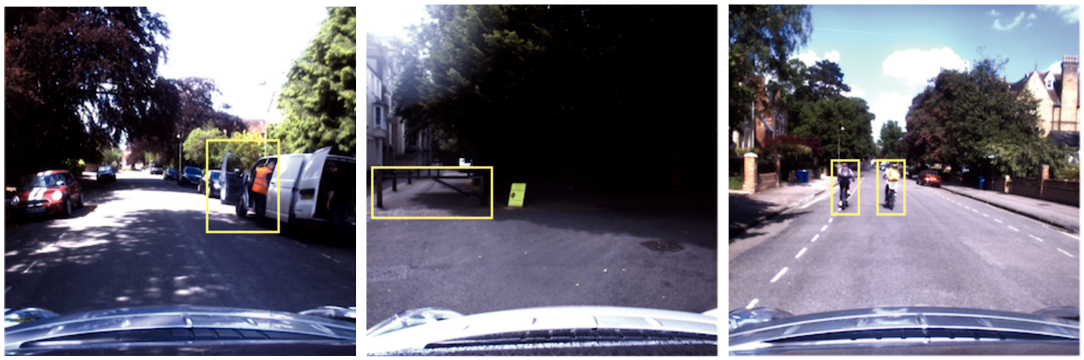
The presented framework was tested on real-world urban traffic scenarios. We use different sequences from the public Oxford RobotCar dataset [Maddern et al., 2017], which allow us to experiment the framework in different weather conditions and scenario situations. Our stereo camera sensor model uses the images coming from the wide baseline of the Point Grey Bumblebee XB3 (BBX3-13S2C-38) trinocular stereo camera. For the laser scanner sensor model we use one layer of the SICK LD-MRS 3D LIDAR. The laser scanner layer is extracted to simulate a low-cost 2D laser scanner, as expected by the proposed approach.

In this section we will first present the multi-sensor fusion results by showing situations where the fusion of the two sensors enhanced significantly the map of the environment. Sequentially, we show an example of the life-long map created for a sequence. Finally, we present quantitative results, such as the entropy and the specificity, to support the proposed method.

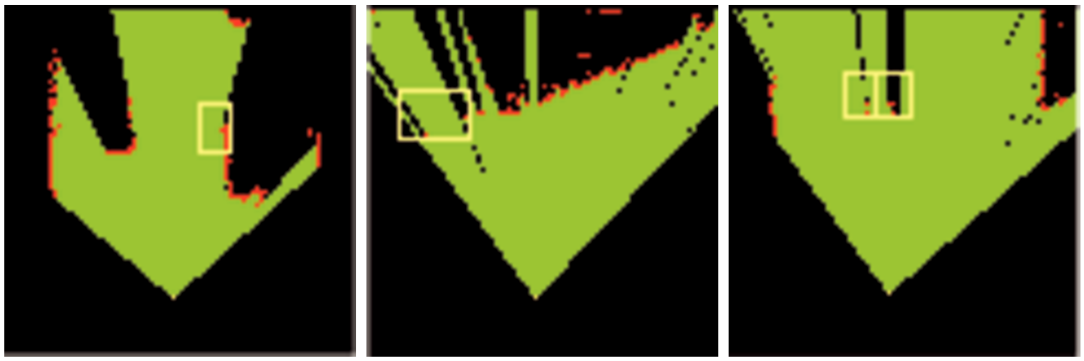
4.4.1 Multi-sensor local fusion evaluation

The results for three different time slots are presented in Figure 4.8. The grid cell size was set to 0.25 m, which can model the environment considering the confidence of the two sensors. The occupied space in red represents the cells with $m(O) > 0.5$, while the free space in green the cells with $m(F) > 0.5$.

Figure 4.8 shows different situations where the fusion grid can provide a richer and more robust environment representation. As we can see by the camera image in the first situation, there is a vehicle with an open door which the laser scanner was not able to detect because of its height. However, in the fusion grid we have the door information due to the stereo camera mapping. The second example shows a similar problem, where a gate was not detected by the laser scanner because it was higher than the laser scanner, but was correctly added to the grid using the stereo camera information. The last example represents the situations where objects have few points detected by the laser scanner and can not be well mapped into the grid. In our example,



(a) Left camera image



(b) Laser scanner grid

(c) Stereo camera grid

(d) Fusion grid

Figure 4.8: Examples of spatial multi-sensor grid fusion at three different time slots.

Figure 4.9: Global map after performing temporal fusion. The colors represent the different states of the life-long grid: CF (green), CU (gray), CO (red), FO (blue) and U (black).

we show that the two bikes in front of the vehicle are better recognized and mapped using the stereo camera images.

These results demonstrate how the proposed method is able to increase the environment understanding and thereby provide more information that can help different tasks of an intelligent vehicle. They illustrate how it is possible to improve the mapping process of a 2D laser scanner by fusing it with a stereo camera. The results also display how, even with less confidence in the data coming from the stereo camera, we can use it to add relevant obstacles to the grid. These improvements provide more knowledge about the current scene obstacles that can increase the safety of path planning methods.

4.4.2 Temporal fusion evaluation

The result from a sequence of time slots is presented in Figure 4.9. It presents the output of the life-long grid, where each state has a different color on the grid. The traversed region is represented by the color gray for the state free space currently unknown (CU) and the color blue for the fixed occupied space (FO). While the current scene is represented by the colors green for the free space currently free (CF) and red for the currently occupied space (CO). In addition, the color black represents the unknown (U) space and the yellow the trajectory of the vehicle during the data collection.

The camera image is also presented in Figure 4.9 and it allows us to understand better what is currently being mapped. We can observe that the pedestrians and the bikes were correctly defined with the CO state and the light pole and buildings as FO. We can also note that some regions that were traversed by the vehicle ended with the state U. There are three possible reasons for this result: the sensors did not detect them, the fixed occupied threshold was not achieved or the last information was a moving obstacle in that region.

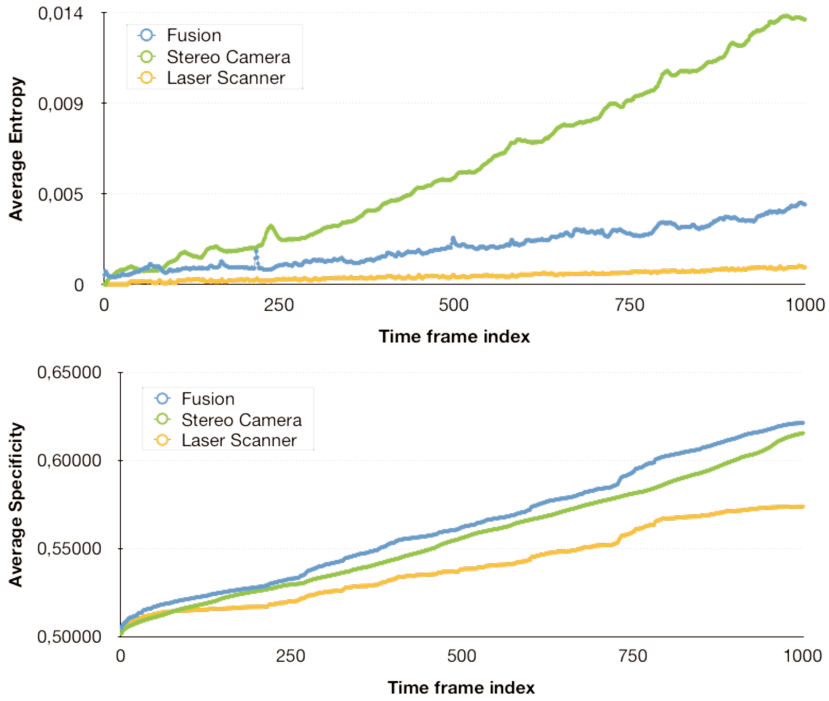


Figure 4.10: Average entropy (top) and specificity (bottom) for each frame

The global map created by the temporal fusion provides a robust estimation of the fixed obstacles and the previous free space. The fixed obstacles are fundamental for localization methods and we can enhance their detection by the proposed spatial fusion. We also improve the confidence on determining if the obstacles are truly fixed by the addition of the life-long layer. Moreover, the cells with states CU can be used for long-term strategies in the path planning process. They can be considered as possible locations for the vehicle to move when it goes back to a previously mapped region. If the region changes its state, our method will update the grid once the vehicle starts to go back to this region.

4.4.3 Quantitative results

Yager [1983] introduced the concepts of entropy and specificity in the framework of Dempster-Shafer's theory. These parameters are complementary and can be used to indicate the quality of evidence. They were applied in different papers [Reineking and Clemens, 2014; Yu et al., 2015] in order to obtain quantitative parameters to analyze the performance of evidential grid maps.

A high value of entropy can indicate inconsistency in the distribution of the mass beliefs. The entropy of a cell can be calculated as follow:

$$E_c = -\sum_{A \subseteq \Omega} m(A) \cdot \ln(Pl(A)) \tag{4.23}$$

where $pl(A)$ is the plausibility of A (please refer to section 4.2 for further details).

The specificity parameter provides an indication of how the belief mass is dispersed. Therefore, it has a higher value if the mass distribution is less doubtful. The specificity

value of a cell can be calculated as follow:

$$S_c = \sum_{A \subseteq \Omega, A \neq \emptyset} \frac{m(A)}{card(A)} \tag{4.24}$$

Considering the two parameters, we can conclude that: the lower the entropy is, the more consistent is the evidence; and the higher the specificity is, less doubtful is the evidence. Therefore, for better certainty we need low entropy and high specificity.

In Figure 4.10 we present the average entropy and specificity of the global grid at each time frame (considering every cell of the map). We compare our proposed method with the related work of laser scanner [Moras et al., 2011] and stereo camera [Yu et al., 2015] evidential grid built separately without performing sensor fusion.

In general, the multi-sensor fusion grid has higher specificity compared to the other two grids. Therefore, we can suppose that the grid created by the proposed method is less doubtful and has a more reliable representation of the environment. At the same time, the fusion grid has higher entropy than the laser scanner grid. This is expected since we are performing the fusion of two sources of information with different fields of view. Nonetheless, it still remains a very low value of entropy compared to the stereo camera alone. One can conclude from these values that even with the small addition of entropy, the addition of specificity is significant and provided more information about the environment and thus increases the quality of the map.

4.5 Conclusion

In this chapter, we presented a framework to perform environment mapping by executing multi-sensor data fusion. This fusion allows intelligent vehicles to increase the perception of challenging environment using only two low cost sensors. We also designed a life-long layer that allows us to create global maps and distinguish the different types of information in the map. This layer gives important information such as free space detection, that can be necessary for path planning algorithms, and mobile obstacles location, that are essential for an intelligent vehicle to drive safely.

Results demonstrate how the fusion of the stereo camera grid with the laser scanner grid can enhance the quality of the mapping process and therefore better characterize the environment, detecting obstacles that would not be detected when only one of the sensors were used. Moreover, quantitative results show that we can still have a satisfactory evidential grid quality, even with the fusion of different sources of information.

The proposed approach only performs the mapping of the environment and we considered that the localization of the vehicle was already given. This approach shows that it is possible to maintain a life-long map of the environment using the evidential model and defining different states for the obstacles. The possibility of determining the static obstacles of the environment is crucial for localization algorithms to perform well their task. Considering this, we will present in the next chapter how we can use the proposed life-long map to localize the vehicle while simultaneously mapping the environment.

Chapter 5

Grid Matching Localization

Contents

5.1 Introduction 71

5.2 Proposed Solution 72

5.2.1 Pre-processing 74

5.2.2 Intensity-based grid matching 75

5.2.3 Re-localization 76

5.3 Experimental Results 76

5.3.1 Odometry Drift 78

5.3.2 Influence of the weight matrix 78

5.3.3 Influence of the time window 79

5.4 Conclusion 80

Résumé du chapitre 5

Dans ce chapitre, nous proposons une solution qui fusionne les techniques d'enregistrement d'image avec la méthode évidentielle pour effectuer simultanément la cartographie de l'environnement et la localisation du véhicule. Nous estimons la trajectoire du véhicule à chaque moment en faisant correspondre deux images de la carte et en trouvant la transformation géométrique entre les deux. Ces images sont les cartes locales et globales des grilles d'occupation évidentielles représentées en différents niveaux de gris. L'objectif principal de la méthode proposée n'est pas seulement de pouvoir trouver une localisation fiable, mais aussi de créer simultanément une carte cohérente de l'environnement.

5.1 Introduction

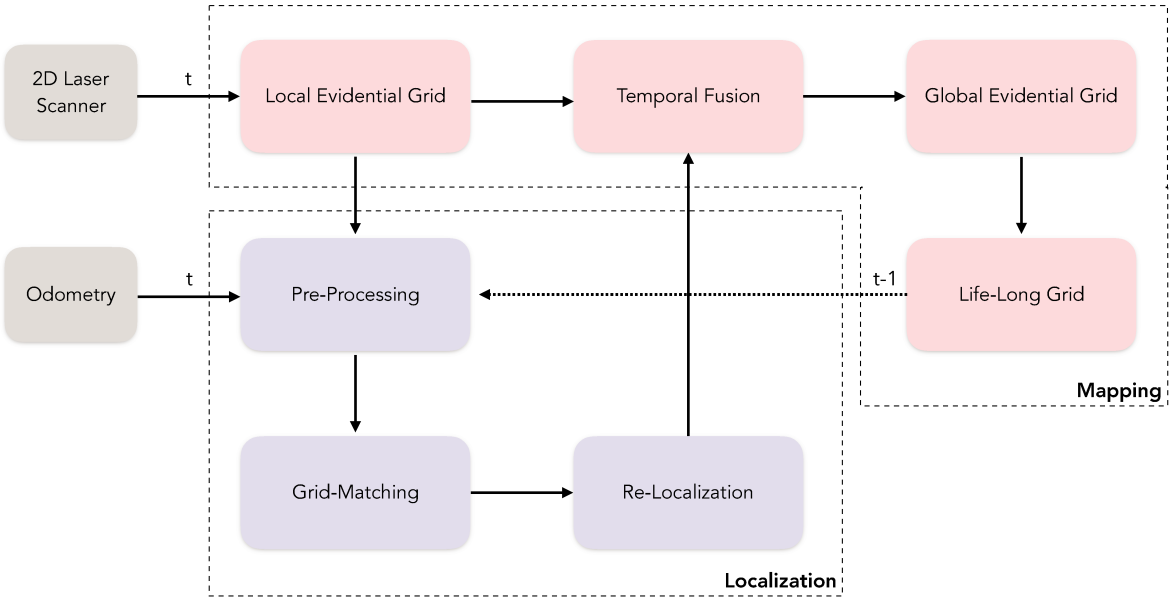


Figure 5.1: Schematics of the proposed system.

In the previous chapter, we proposed the creation of a life-long grid using the evidential framework in order to maintain a global map of the environment and differentiate the static obstacles from the dynamic ones. The method addressed only the mapping process and considered that the trajectory of the vehicle was known.

In this chapter, we propose a solution that merges image registration techniques with the evidential framework to perform not only the mapping of the environment but also the localization of the vehicle. The mapping algorithm is based on the use of evidential grid maps, along with the life-long layer, presented in the previous chapter. At the same time, we estimate the vehicle's trajectory at each timestamp by matching two map images resulting from transforming the evidential grid maps (local and global) into grayscale images and finding the geometric transformation between them. The main goal of the proposed method is not only to be able to find a reliable localization, but to simultaneously create a consistent map of the environment. The architecture of the proposed system is presented in Figure 5.1.

The proposed localization approach can be applied with any sensor that can generate evidential grid maps. In the previous chapter, we explored the use of laser scanners and stereo cameras. During our tests we observed that the mapping with the stereo camera generates noisy information that is less precise than with only the laser scanner. This can be observed in Figure 4.8 (page 64). The addition of the stereo camera data can be necessary to add relevant information for the vehicle to avoid obstacles. However, during the localization algorithm we need a map with higher precision to obtain an accurate localization. For this reason, we use in this chapter only the 2D laser scanner for our proposed method.

5.2 Proposed Solution

The key point of the proposed approach is to estimate the vehicle's localization by performing grid matching. This process occurs at each creation of a new local evidential grid and the goal is to find the most accurate geometric transformation between the new grid and one with previous information. After we obtain the transformation, it is used to transform the current grid map to the global map frame, that represents the evidential masses accumulated over time. Once the transformation is performed, the two grids are merged using the method described in subsection 4.3.3. Finally, the new global grid map will update the life-long grid and represent the environment with the different states presented in subsection 4.3.4.

During our grid matching approach we will use three types of images:

1. **Local grid**: image representing the current sensor acquisition.
2. **Recent global grid**: image created from the life-long grid considering only the states that were updated recently.
3. **Complete global grid**: image created from the life-long grid considering all the information gathered since the beginning of the mapping process.

Taking into consideration these images, the localization method can be divided into three steps, as illustrated on Figure 5.2:

1. **Pre-processing**: it takes as input the odometry measurement along with two grids, the local evidential grid and the life-long grid. The objective is to transform these two grids into the previously defined images that will be used in the grid matching process.
2. **Grid Matching**: we apply an intensity-based registration algorithm to find a geometric transformation between the previously craeted images. This process is performed in two different situations, at each creation of a new evidential grid and every 3 seconds. In the first case, we match the new local grid with the recent global grid information that was updated in the last 3 seconds. The objective of this first matching is to correct the odometry error of the current timestamp. In the second case, we match the local grid (after being corrected by the first matching) with the complete global grid. This second matching is a way of reducing the possible drift created over time.
3. **Re-localization**: in this step we correct the localization estimated by the odometry using the output of the grid matching. It defines a new pose for the vehicle and uses it to transform the local evidential grid to be merged in the temporal fusion process described in subsection 4.3.3.

The details and specifics of each one of the steps will be presented in the following subsections.

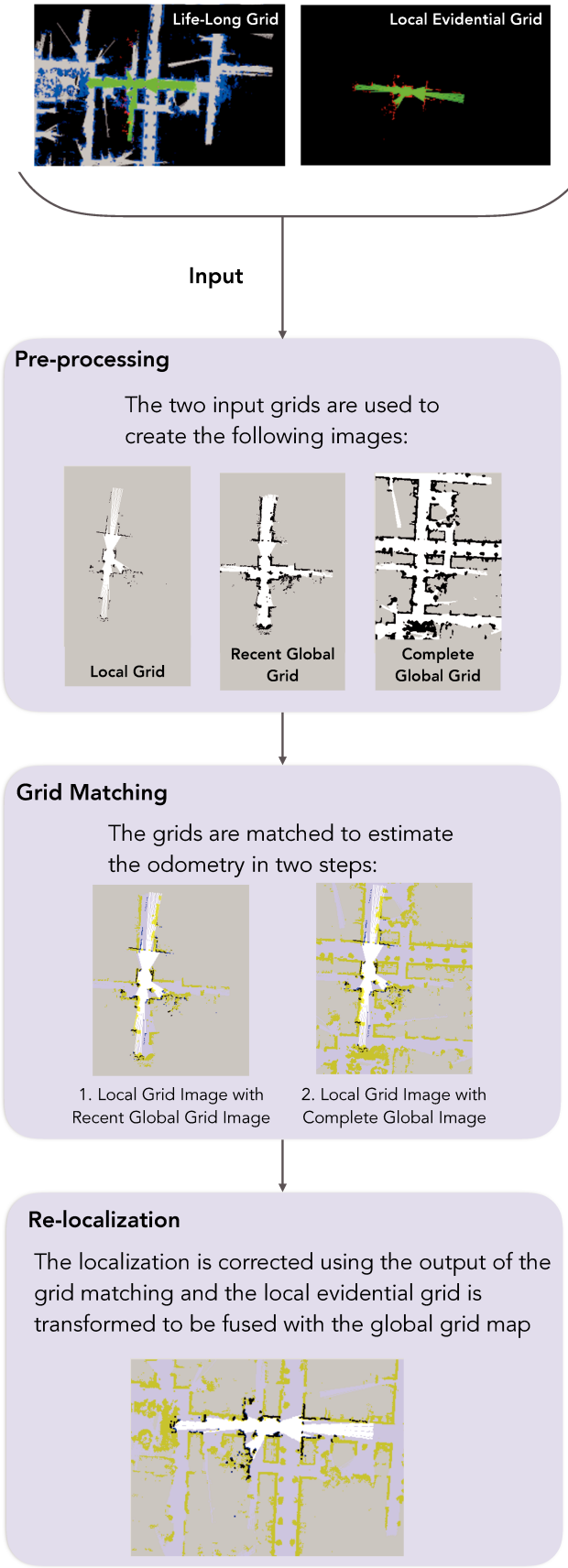


Figure 5.2: The three steps of the grid matching process for localizing the vehicle. In the pre-processing step, the local grid represents the current scan grid map, while the recent grid stores the last 3 seconds of the life-long grid and the complete grid all the information gathered in the life-long grid since the beginning.

5.2.1 Pre-processing

The input of our algorithm is the local evidential grid, created by the current sensor acquisition, and the life-long grid map. In order to find the right geometric transformation between these two grid maps, we need to extract from both the relevant information to perform image matching. Figure 5.2 shows an example of the grayscale images created during this step. The first image, the local grid image LG, will represent the local evidential grid and we create it by transforming the evidential belief of each cell into a grayscale pixel.

First, we use the odometry value to have an estimation of the positioning of the vehicle related to the last pose. We found experimentally that only the translation values of the odometry are necessary for the proposed method, the orientation can be more precisely estimated directly in the grid matching step. The local grid map is then transformed using the translation values of the odometry to the estimated location. In this way, we transform the local grid map to the same coordinate system as the life-long grid, considering a first estimation of localization.

Sequentially, we find the region of interest in the local grid map which consists in the region limited by the current scan with a fixed margin. Finally, in this region the cells with $m(O) > 0.5$ will be represented by a pixel in black, while the cells with $m(F) > 0.5$ are represented in white. The remaining of the cells receives the gray color.

The second image, the global grid image GG, is created based on the states represented by the life-long grid. Depending on the timestamp two different images can be produced to match with the local grid: the recent global grid and the complete global grid. The recent global grid is produced by applying a time window in the life-long grid that makes possible to represent only the recent information added in this grid. We chose a time window of 3 seconds. In other words, we consider the information of the life-long grid that was updated not more than 3 seconds before the current timestamp. This image will be used in the matching process to reduce the odometry error at each creation of a new local grid by comparing the current local grid to the recent global grid image.

The complete global grid stores all the information of the life-long grid gathered until the current timestamp. It represents all the sensor acquisitions obtained since the beginning of the trajectory. This image is created every 3 seconds, since this matching is not executed at every time slot. While the recent global grid is used to reduce the odometry error of each time slot, the complete global grid is used to reduce the global drift accumulated in the map.

To create these two images we use the cells of the same region of interest defined for the local grid but relative to the life-long grid. The cells with the states fixed occupied or currently occupied are represented in black, while the cells with the states currently free space and free space currently unknown are represented in white. The unknown space receives the color gray. In Figure 5.2 we can observe an example for the two global images. It is possible to notice how the recent global grid is more clean, which makes the matching process easier and faster. For this reason, this image is used at

each timestamp, while the other only every three seconds.

In the creation of the global grid images we represent the dynamic and static obstacles with the same color, which can add conflict information. For example, we would not want to use the information of a moving vehicle in the same way that we use the fixed obstacles to estimate the displacement between the two grids. We minimize this problem by generating a weight matrix to give value to the most reliable information in the image. The matrix is generated by giving for each pixel of the image a weight that takes into consideration the state of the life-long grid, where the cells with the state fixed occupied receive a higher weight compared to others. The values of the matrix were defined experimentally, fixed obstacles receive a weight of 1.0, free space 0.8, dynamic obstacles 0.3 and unknown space 0.0.

5.2.2 Intensity-based grid matching

Let the local grid image LG and the global grid image GG be the two grayscale images created in the previous presented steps, and x be the coordinates of a pixel in the image. Our proposed grid matching algorithm is based on intensity-based image registration, where the goal is to find a geometrical transformation such that $LG(x)$ matches as much as possible $GG(x)$ for every x. The algorithm accomplishes this goal by maximizing a similarity measure based on the image intensity values.

Let $W(x;p)$ be the set of allowed warps, where p is a vector of parameters related to translation and rotation. Thus, the method can be formulated as a classic image registration problem where we minimize the sum of squared error between two images. The objective is to find the value of p that minimizes the following equation:

$$\sum_{x}[LG(W(x;p)) - GG(x)]^2 \cdot w(x) \tag{5.1}$$

where $w(x)$ represents the weight matrix created in the previous pre-processing step. The goal of the matrix is to give more value to the error caused by the pixels of fixed obstacles compared to the ones caused by possible moving obstacles.

In practice, the process can be described as an iterative solution known as the Inverse Compositional Algorithm [Baker and Matthews, 2001], and is summarized as follows:

1. Calculate the matrix H:

$$H = \sum_{x}[\nabla GG\frac{\partial W}{\partial p}]^T[\nabla GG\frac{\partial W}{\partial p}] \tag{5.2}$$

   where ∇GG is the gradient of image GG.

2. Warp the local grid image LG with $W(x;p)$

3. Compute the new increment to the warp parameter Δp:

$$\Delta p = H^{-1}\sum_{x}[\nabla GG\frac{\partial W}{\partial p}][LG(W(x;p)) - GG(x)]^2 \cdot w \tag{5.3}$$

4. Update the warp parameter:

$$p = p + \Delta p \tag{5.4}$$

Step 2 to 4 are performed until it reaches the maximum number of iterations, or Δp is smaller than a predetermined threshold.

Finally, the output is a new local grid image warped considering the final p value and the translation and rotation values related to p. As previously explained, this step is performed at two different moments, at each new creation of a grid and at every 3 seconds. In the second case, the result of the first matching is the input for the second one, where the local grid is compared to the complete global map.

5.2.3 Re-localization

The output of the grid matching step is the delta heading angle and the translation vector in 2D related to the transformation between the local grid map and the life-long grid. Let (x_p, y_p) and θ_p be the previous position and heading of the vehicle on the map before the grid matching method, p the translation vector and $\Delta\theta$ the delta heading angle. The new corrected position of the vehicle in 2D (x_c, y_c) and new corrected angle (θ_c) is then calculated as follows:

$$\begin{aligned} x_c &= x_p + p \cdot \cos(\theta_p + \Delta\theta) \\ y_c &= y_p + p \cdot \sin(\theta_p + \Delta\theta) \\ \theta_c &= \theta_c + \Delta\theta \end{aligned} \tag{5.5}$$

This new updated pose will serve as an estimation along with the odometry measurements to the vehicle's localization in the next timeslot. The warped local grid is used as an input for the temporal fusion, presented in subsection 4.3.3, where it will be merged to the global grid map. The new global grid evidential values are then used to update the different states stored in the life-long grid.

5.3 Experimental Results

For validation the KITTI dataset [Geiger et al., 2013] was used as data input. For the laser scanner sensor model, we extracted one 360° layer from the Velodyne data. The layer is extracted to simulate a low-cost 2D laser scanner. We also simulated an odometry value by adding a white noise to the ground truth velocity, with standard deviation of $\sigma_w = 0.5$ m/s, and to rotation speed, with standard deviation $\sigma_w = 0.5$ rad/s. These parameters are similar to the configuration presented in [Trehard et al., 2015], which we use to compare our results.

We evaluate 10 sequences of the KITTI odometry database to analyze the proposed method. In this database there are 11 sequences, however, we eliminate the sequence 01 which consists of a sequence in a highway. In this sequence the simulated 2D laser scanner is not able to detect obstacles most of the time, making impossible for the proposed approach to work.

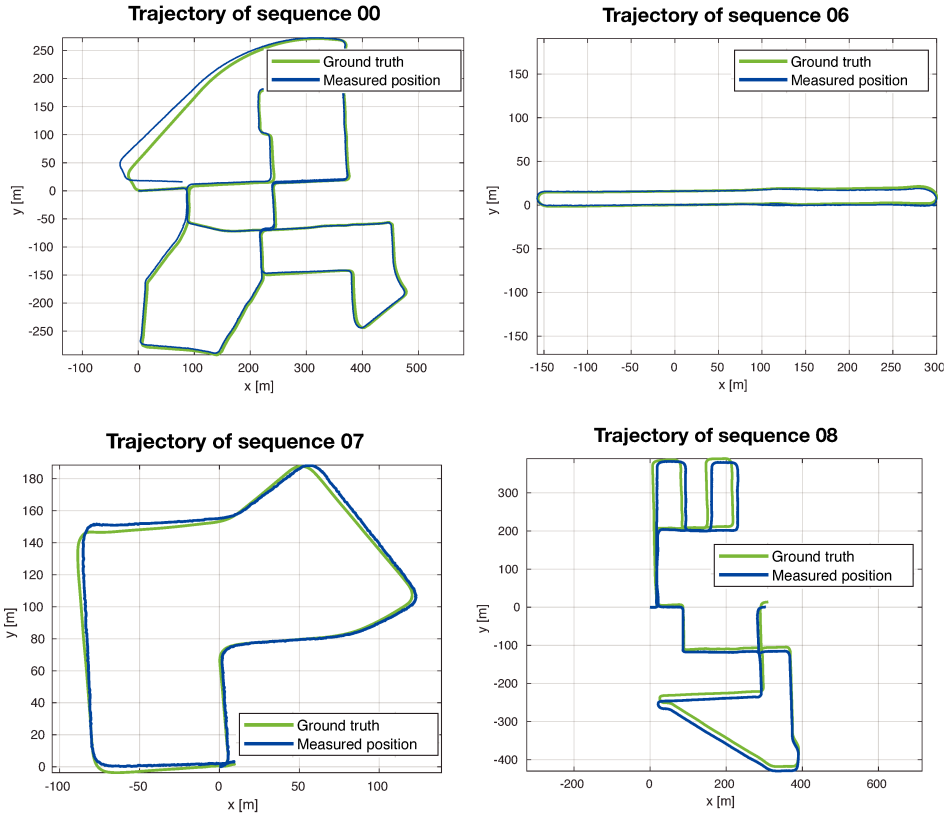


Figure 5.3: Example of trajectories from the KITTI odometry dataset applying the proposed method.

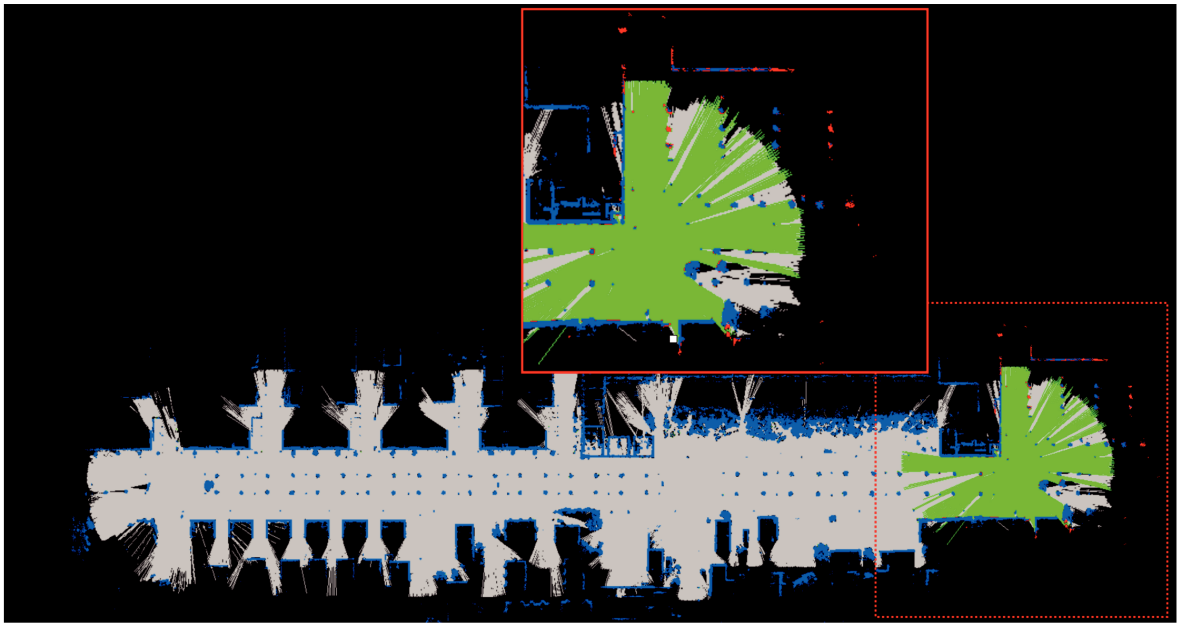


Figure 5.4: Life-long grid of sequence 06 of the KITTI odometry database. The map shows all five states: free space currently free (green), free space currently unknown (gray), currently occupied space (red), fixed occupied space (blue) and unknown (black).

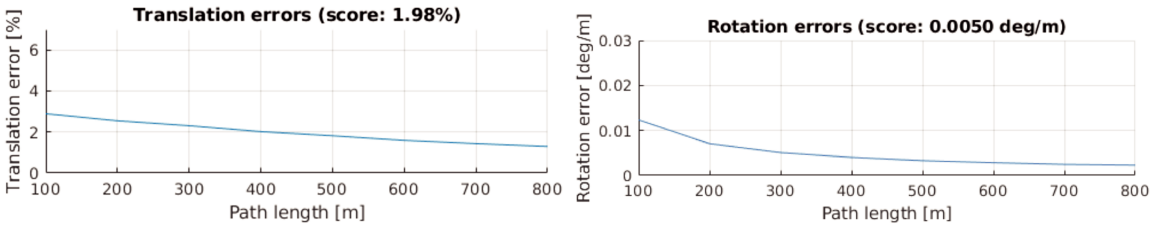


Figure 5.5: Translation and rotation errors for 10 sequences of KITTI odometry database applying the proposed method.

Figure 5.3 shows the trajectory of the vehicle with the measured positions and the ground truth positions in four different sequences. The complete life-long map result can be observed on Figure 5.4, where it presents the entire environment traversed on the sequence 06 along with the five different states with resolution of 0.4m. These results show qualitatively how the proposed method is able to localize the vehicle generating a smooth trajectory close to the ground truth, and at the same time create a consistent global map of the environment.

5.3.1 Odometry Drift

In order to validate our method and compare to other solutions, we tested it over 10 different sequences of the KITTI database for odometry and we calculated the drift using the method presented in [Geiger et al., 2013]. Since we perform 2D SLAM, the method is adapted to calculate the drift in 2D only, as proposed by Trehard et al. [2014]. The score is calculated by being the mean error in the different trajectory lengths considering all sequences. All the sequences were evaluated using evidential grid maps with a resolution of 0.4 meters.

Figure 5.5 shows that the translation error varies between 1.29% and 2.89%, with a score of 1.98%; and the rotation error is between 0.0024 deg/m and 0.012 deg/m, with a score of 0.0050 deg/m. As a comparison, the method presented in [Trehard et al., 2015], that also uses only one layer of the LIDAR in the KITTI dataset, has a score of 5.74% for translation and 0.0112 deg/m for rotation. Therefore, the proposed method has not only the advantage of creating a consistent and reliable map of the environment, but also the advantage of improving the localization accuracy.

5.3.2 Influence of the weight matrix

In subsection 5.2.2 we included the information from the life-long grid into the grid matching process by adding the weight matrix w. This information allows to give more value to the pixels representing fixed obstacles in the environment. We evaluate the addition of this parameter by comparing the map drift in the 10 sequences using w (Figure 5.5) and without using it in the grid matching step (Figure 5.6).

We can observe that the error in translation and in rotation is reduced by more than half when we apply the weight matrix. It shows how the evidential information leads to a more precise representation about what is relevant in the environment to perform grid matching. By analyzing the results, we can presume that the addition of

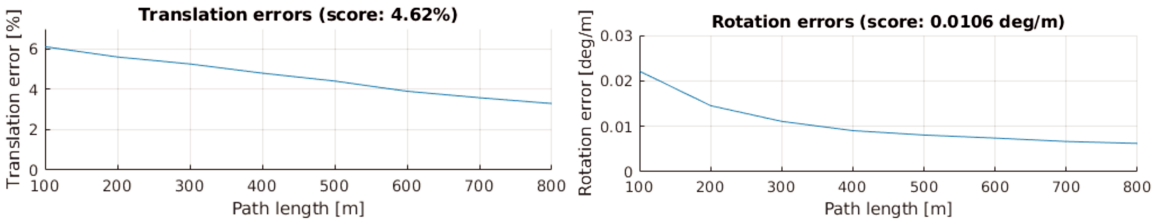


Figure 5.6: Translation and rotation errors for 10 sequences of KITTI odometry database without the weight parameter.

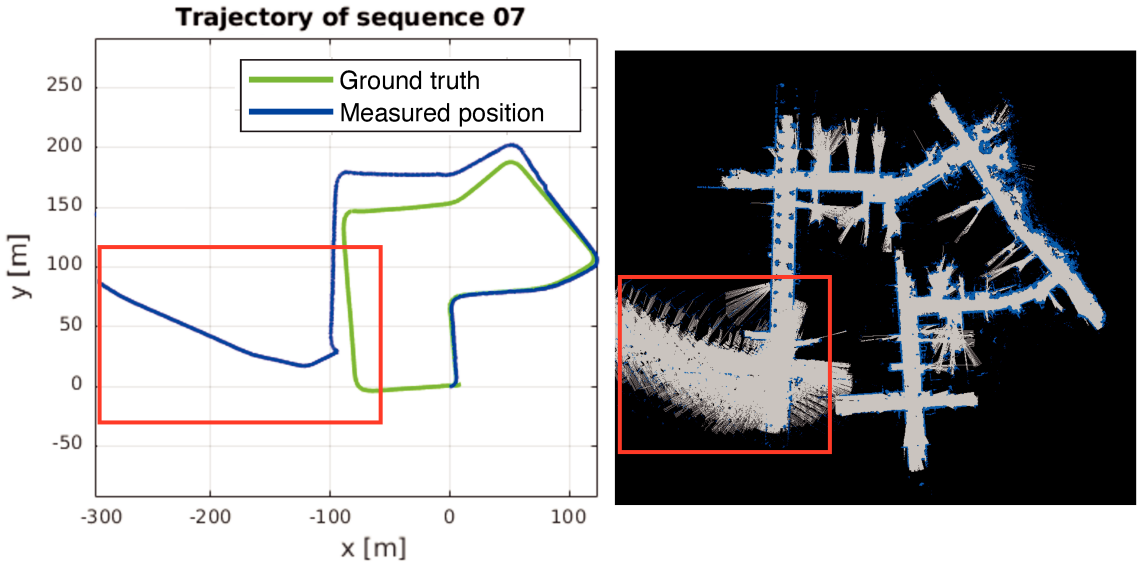


Figure 5.7: Trajectory and global map for sequence 07 without using the time window in the grid matching step.

a higher value to the fixed obstacles provides a better reference for the grid matching step which improves the localization.

5.3.3 Influence of the time window

In subsection 5.2.2 we explained that the grid matching process can be performed at two different moments. In the first, we use a time window of 3 seconds to only match the current grid map to the recent information added to the map. While in the second moment, we match every 3 seconds the current grid map (after being corrected in the first matching) with the complete map. We evaluate the addition of this time window by presenting a sequence (Figure 5.7) where we always only match the current grid map with the complete map, without using any time window.

The figure shows that the method is not able to obtain the trajectory and map the environment correctly when the time window is not used. We can analyze from the results that this occurs generally when there are few LIDAR points in the current grid map, while there are a lot of information in the complete map. In the case presented in Figure 5.7, the vehicle was in a curve with not a lot of information around, and the image matching method was not able to find the right transformation between the

images and as a result it obtained the wrong position of the vehicle.

Without the time window, the image matching algorithm has a larger zone to explore in order to find a good match with the current scan. If the algorithm does not find a good match, it can generate larger errors trying to match it with further regions. Considering this, we believe that restraining the global map region is one of the reasons why the algorithm is able to work with more precision when the time window is added.

Another situation that could lead to this type of error is when there are not a lot of points in the current scan and the vehicle is in a new region but with similar information to where the vehicle has traversed before. In this case, the algorithm can end up estimating that the vehicle came back to that previous region, generating large errors. The older information is not considered when the time window is added, making the image matching algorithm simpler. Therefore, we can conclude that the addition of the time window makes possible to perform SLAM using the method presented.

5.4 Conclusion

In this chapter we presented a solution to the SLAM problem based on evidential grid map matching. We introduced a new grid matching technique that uses the evidential information to treat differently static and dynamic obstacles. One of the main drawbacks of the proposed method is the lack of loop closure, which can result in an accumulation of drift errors. However, we can observe that even for long sequences the drift error was low, because we were able to reduce it using the global map for the grid matching process.

Results demonstrate that the proposed solution is able to create a consistent global map of the entire vehicle's trajectory while simultaneously providing a good estimation of the position. The proposed approach has competitive results compared to other methods using this sensor configuration. It proves that the use of low cost sensors can be enough to have an accurate localization and mapping in challenging environments.

Part III

Deep Learning Localization

This part explores the use of Deep Learning techniques focusing on the localization problem of autonomous vehicles in urban environments. In the first chapter, we present a novel learning-based solution for odometry estimation using only 2D laser scans as input. In the second chapter, we extend the previous work by proposing an end-to-end network that performs the fusion of a 2D laser scanner and a monocamera to estimate the odometry. Finally in the last chapter, we present a novel method where we use the evidential grid maps presented in the previous part along with the data from a 2D laser scanner as input of a network to perform re-localization.

Chapter 6

An LSTM Network for Real-Time Laser-based Odometry Estimation

Contents

6.1 Introduction 85

6.2 Preliminaries: Deep Learning 86

6.2.1 Machine Learning 86

6.2.2 Neural Networks 87

6.2.3 Recurrent Neural Networks 90

6.3 Proposed Solution 93

6.3.1 Data encoding 93

6.3.2 Network Architecture 94

6.3.3 Training 96

6.4 Experimental Results 98

6.5 Conclusion 101

Résumé du chapitre 6

Dans ce chapitre, nous présentons une nouvelle méthode de localisation qui explore l'utilisation des réseaux de neurones récurrents (RCNN) en se servant uniquement de scanners laser 2D. L'application des RCNN permet non seulement d'extraire les caractéristiques des données du scanner laser grâce aux réseaux de neurones convolutifs (CNN), mais en plus, de modéliser les connexions possibles entre les scans consécutifs à l'aide du réseau de neurones Long Short-Term Memory (LSTM). Les résultats sur un ensemble de données montrent que la méthode peut fonctionner en temps réel sans utiliser l'accélération GPU tout en ayant des performances compétitives par rapport à d'autres méthodes, ce qui en fait une approche particulièrement intéressante qui pourrait venir compléter les systèmes de localisation traditionnels.

6.1 Introduction

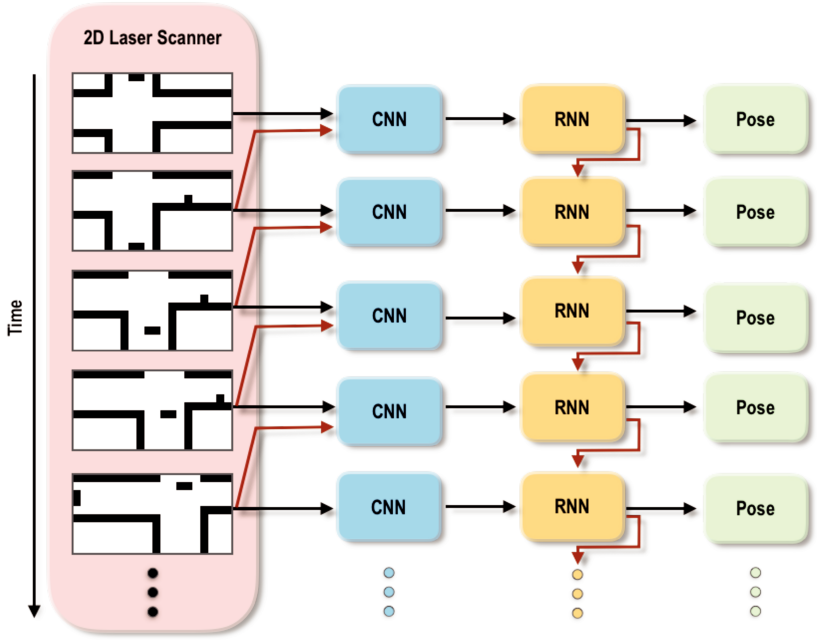


Figure 6.1: Overview of the proposed system. The complete Recurrent Convolutional Neural Network takes a sequence of 2D laser scanner measurements as input, learns its features by a sequence of CNNs, which are used by the RNN to estimate the poses of vehicle. The output is a 2D pose of the vehicle composed by two values, one for translation and another one for rotation.

In the last years, Deep Learning methods have received attention in the field of autonomous driving. The most common applications use cameras or laser scanners for tasks such as obstacle detection and classification. These tasks are mainly in the field of environment understanding and mapping. However, besides the need to understand the environment, the localization of the vehicle is still a challenging process and it has not yet been extensively explored by machine learning techniques.

In this chapter, we present a novel framework that explores the use of deep Recurrent Convolutional Neural Networks (RCNN) for odometry estimation using only 2D laser scanners (Figure 6.1). The application of RCNNs provides the tools to not only extract the features of the laser scanner data using Convolutional Neural Networks (CNNs), but in addition it models the possible connections among consecutive scans using the Long Short-Term Memory (LSTM) Recurrent Neural Network. Results on a real road dataset show that the method can run in real-time without using GPU acceleration and have competitive performance compared to other methods, being an interesting approach that could complement traditional localization systems.

The chapter is organized as follows. First, we briefly introduce the main concepts of deep learning in section 6.2; the proposed method and the design of the network are presented in section 6.3; experimental results are shown in section 6.4; finally conclusion and perspectives are given in section 6.5.

6.2 Preliminaries: Deep Learning

The term Deep Learning is used to call the area of research in the machine learning field that uses algorithms known as Artificial Neural Networks (ANNs). This type of algorithm is inspired by the structure and function of the brain and is a powerful tool to learn how to perform a variety of tasks. In this section we present a brief description of the main concepts of deep learning techniques that are necessary for understanding the next sections. There are many crucial concepts of this field that are not closely related to the content of this thesis, which we will not cover for brevity.

We start this section by presenting the idea behind machine learning algorithms, then we move towards neural networks and their main concepts and we finalize by presenting a special kind of neural network that is able to learn sequential information, the recurrent neural network.

6.2.1 Machine Learning

The term machine learning is generally used for algorithms that are able to understand the structure of data and fit this data into models that can be used to specific tasks. A more common definition of machine learning is the one given by Mitchell [1997]: "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E*". One can imagine the variety of tasks T and experiences E that this could be applied to, which is why machine learning became a powerful tool for so many different applications.

Considering the different forms that E, P and T are used, we can differentiate four main types of learning:

- **Supervised learning** uses labeled training data to predict a target variable. The algorithm learns based on the labeled dataset, which provides an answer key that the network can use to evaluate its accuracy on training data. This type of learning can be divided into two main groups: classification and regression tasks, depending if the output belongs to a discrete set of variables or a continuous set. [Wang et al., 2017][Nicolai et al., 2016]

- **Unsupervised learning** uses an unlabeled training data to discover some structure on the set of examples to find a potential solution. The main goal for this type of learning is more to understand well the data itself and not only apply it to a particular task. [Li et al., 2018]

- **Semi-supervised learning** is a middle point between the two previous learning methods, where only some part of the data is labeled. [Mei et al., 2019]

- **Reinforcement learning** learns actions in an environment based on maximizing cumulative rewards about the appropriateness of its actions. This technique is applied by various different applications to find the best behavior a machine should task in a specific situation. [Jaritz et al., 2018]

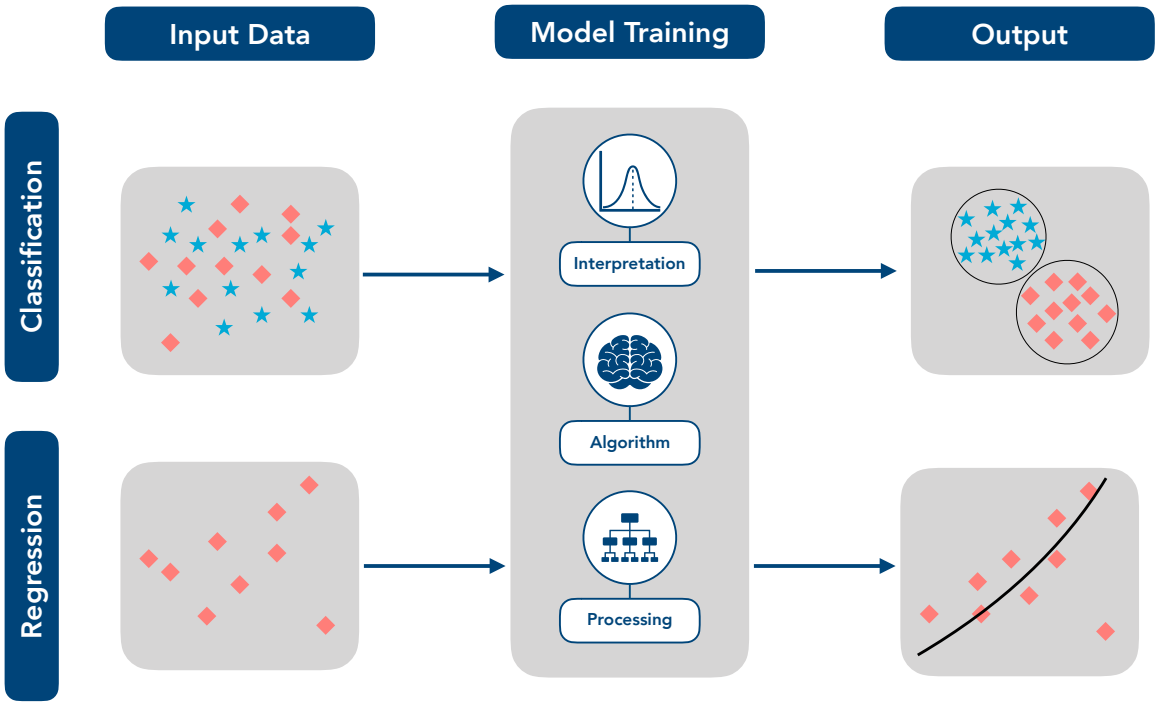


Figure 6.2: Illustration to compare classification and regression in supervised training.

In this work we focus on supervised learning algorithms, where the experience E always consists of observing a set of examples and their labels. There are two main areas where supervised learning is applied to:

- **Classification** predicts discrete label values by identifying if an input data is a member of a particular class or group. The algorithm is evaluated by how accurately it can correctly classify different inputs.
- **Regression** predicts real or continuous data values. The solution is represented by a quantity that can be flexibly determined based on the inputs of the model rather than being confined to a set of possible labels.

The difference between these two types of supervised learning is illustrated in Figure 6.2.

6.2.2 Neural Networks

Artificial Neural Networks (ANNs) are a specific family of machine learning models. They are composed by a set of units (neurons), a connection topology and a learning algorithm. Neurons are a computational unit that is able to interpret the input, process it and generate an output. The units can be split into layers, known as input layers, hidden layers or output layers (Figure 6.3). The input layers represent the input of the network and have fixed parameters, while the hidden layers are the unobserved units that are connected to the output layers, where we obtain the output computed by the network model.

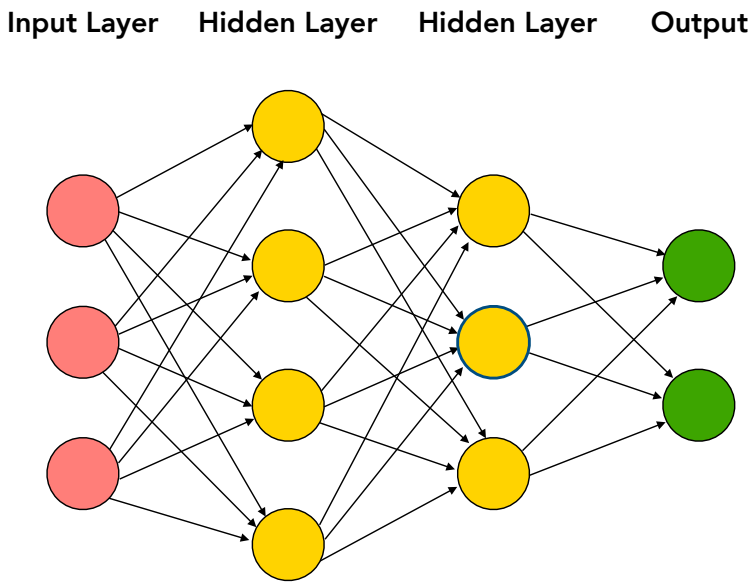


Figure 6.3: An example of the different layers in a Feed-Forward neural network.

Each model has different parameters, weights and biases, that are learned interactively through an optimization technique to match as much as possible the model outputs to the target. The distance measure to this match, also known as loss function, depends on the model and the task of the algorithm. Considering this, we can define how the process occurs at each neuron. A simple unit can be described as receiving a vector of inputs x and computing the affine transformation

$$z = W^T x + b, \tag{6.1}$$

where the weight matrix W defines the mapping from x to z with a bias b. These parameters define each layer of the network. After computing it, the result is passed to an activation function $g(z)$. The activation function is based on the biological notion of the potential of activation of a neuron. The use of these functions make possible to add nonlinearities to the learning process. The most common activation function, and frequently used in this thesis, is the rectified linear unit (ReLU), which consists of the max function:

$$g(z) = \max\{0, z\} \tag{6.2}$$

There are several other options for the activation function, such as Maxout, Sigmoid and Softmax [Nwankpa et al., 2018]. The choice for the right function depends on the network task and the kind of data used.

After passing the input data through the network and generating an output, the next step if to calculate the loss function. This function is used as a metric to evaluate the network and as the activation function, the right choice of loss function depends on the task and the type of output generated by the network. The most common functions usually are Cross-Entropy [Janocha and Czarnecki, 2017], if its a classification task, and mean-square-error (MSE) or root-mean-square-error (RMSE) [Nie et al., 2018], if

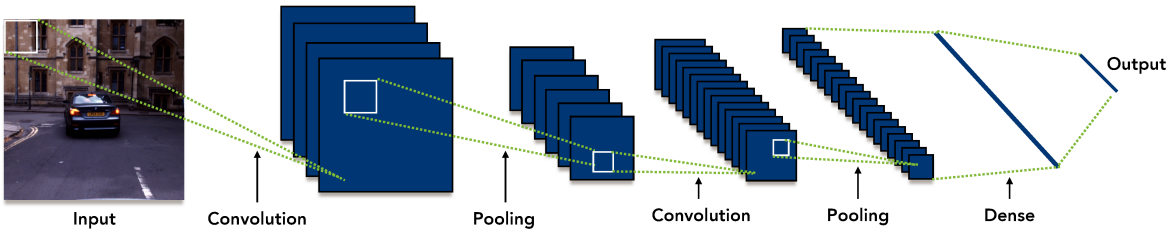


Figure 6.4: Example of a network composed by convolutional, pooling and dense layers.

it is a regression.

As mentioned before, the network parameters are learned using an interactive optimization process. This process is based on the method of Gradient Discent (GD), where the gradient is used at each step to determine the best values for weight W and bias b. By default, the information in a neural network is passed in a forward direction from the input layer to the output layers, this is known as forward propagation. During the training, the back-propagation algorithm is used to give backward to the network the information from the loss function in order to compute the gradient. By repeating it in an interactive way, the network is trained to learn the desired task.

In Equation 6.1 we showed the most simple operation performed by a unit layer. However, there are other types of operations that can be performed by a unit. These operations determine the type of layers that we have in our network. In this work the main layers used are the dense layer, the convolutional layer and the pooling layer. An example of network with those types of layers is presented in Figure 6.4. We will summarize these layers and their characteristics for a better understanding of this work:

- **Dense layer** uses the Equation 6.1 to set linear connections between the input and output. For this type of layer, all units are fully connected to all units in a sequential layer. This leads to a large set of trainable parameters and for this reason, this layer can be successful for low dimensional data, but very expensive to large amounts of data. The dense layer can be also known as fully connected layer or linear layer.

- **Convolutional layer** applies convolutions to the input using a set of kernels, also known as filters. Each filter is applied to a location in the input, where it performs the product between each element of the kernel and the part of the input that it overlaps with. The result of the convolution is summed to obtain the output at each location. The new composed output is known as feature map. In every convolutional layer there are three types of parameters that can be adjusted: the kernel size, the step size and the zero padding. The kernel size is simply the size of the filter that will be applied to the input. The step, also known as stride, consists in the distance between two consecutive positions of the kernel. Finally, the zero padding is the number of zeros that is added at the input, which is commonly used when the dimensions of the input need to be preserved in the output.

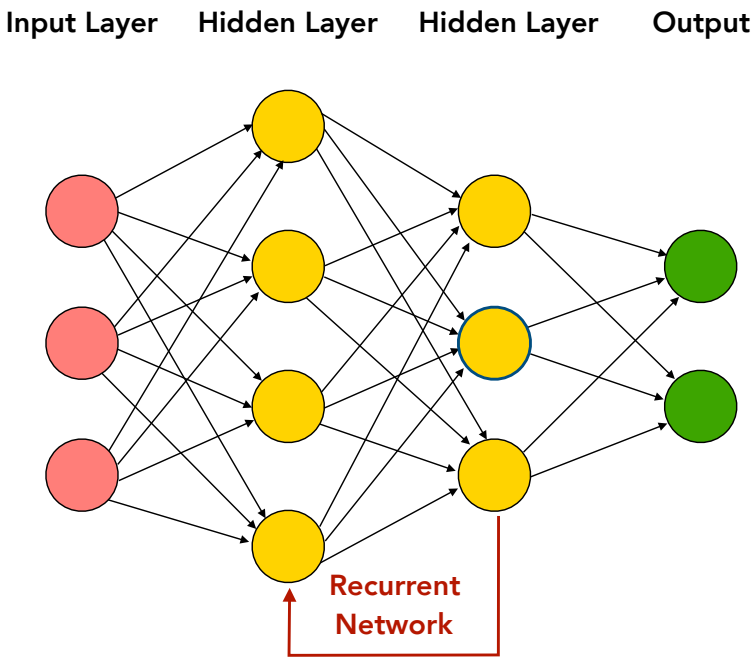


Figure 6.5: Recurrent Neural Network architecture.

- **Pooling layer** is commonly used in networks after convolutional layers to reduce the spatial size of the input, the amount of parameters and the computation time for the network. It applies an operation similar to the convolution, but replaces the linear combination with some other function, such as the maximum or average of the input of the neighbors. In the same way as the convolutional layer, the pooling layer has the parameters kernel size, step and zero padding.

6.2.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a special kind of neural network, in which there is at least one recurrent (cyclic) connection (Figure 6.5). This type of network is commonly used to deal with sequential inputs or sequential tasks, such as language translation, speech recognition, video detection. The main difference between traditional neural networks and the recurrent ones is that the parameters of the network are shared between different parts of the model. In an RNN, each component of the output is produced by applying the same update rule that was applied to each component of the previous output. This is how the weights are shared through the model. Moreover, the recurrent model usually has an output layer that uses information from the hidden state to make predictions. Since the state of a hidden unit at each time step is a function of all inputs from previous time steps, the recurrent connections work similar to a kind of memory. For this reason, a hidden unit that keeps information across multiple time steps is known as memory cell. The behavior of a memory cell can be generally described as follows:

$$h_t = f(Wx_t + Uh_{t-1}) \tag{6.3}$$

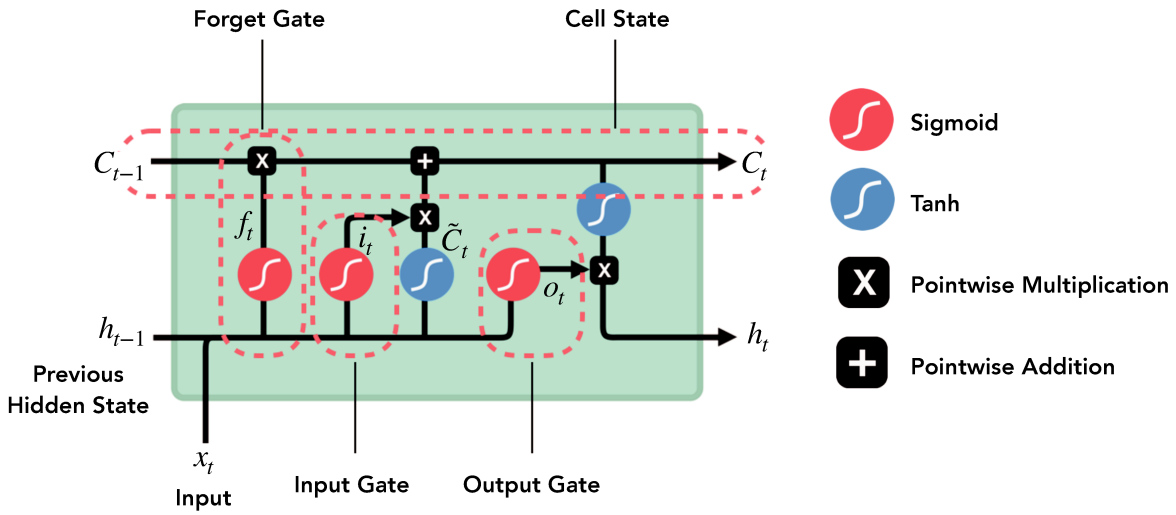


Figure 6.6: Long-Short Term Memory architecture.

The hidden state at time t is h_t. Its value is calculated by passing to an activation function f the input state modified by the weight matrix W and added to the previous hidden state multiplied by its own weight matrix U.

In theory, RNNs are able to remember long sequences, but in practice their memory is limited by their finite size and by the exploding gradient problem while training their parameters. The exploding gradient problem problem occurs when the influence of a given input on the hidden layer, and therefore on the network output, either decays or explodes exponentially as it cycles around the recurrent connections. When this happens, it is hard for the network to learn tasks from more than a small number of time steps. In order to overcome these memory limitations, advanced cell structures were suggested, the two most commons are the Gated Recurrent Unit (GRU) and the Long-Short Term Memory (LSTM), where the use of gated structures was applied to deal with these difficulties. The gates are internal mechanisms that can regulate the flow of information and they can learn which data in a sequence are more important to keep or to be thrown away.

In this work, we chose to use LSTMs as our RNN configuration. The architecture of this kind of network is composed by a set of recurrently connected subnets, called memory blocks. At each block there is one or more self-connected memory cells along with three types of gates:

- **Forget gate** decides what information should be kept or discarded. The information from the previous hidden state and the current input goes through a sigmoid function, creating an output between 0 and 1:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{6.4}$$

  A 1 output means that the information should be kept and a 0 that it should be discarded.

- **Input gate** is responsible for the addition of new information to the cell state. This addition is performed in three steps. First, the previous hidden state and the current input are passed to a sigmoid function:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{6.5}$$

  Similarly to the forget gate, this function acts as a filter for all the information. Sequentially a vector is created containing all the possible values that can be added to the cell state, as perceived by the previous hidden state and the current input. This is done by passing the same input of the sigmoid to a tanh function, which outputs values from -1 to 1:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{6.6}$$

  Finally the outputs of the sigmoid and of the tanh are multiplied to create the information that will be added to the cell state:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{6.7}$$

- **Output gate** is responsible to decide what the next hidden state will be. First, we pass the previous hidden state and the current input into a sigmoid function:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{6.8}$$

  Sequentially, we pass the newly modified cell state to a tanh function. Finally, we multiply the tanh and the sigmoid outputs to decide what information the hidden state should carry:

$$h_t = o_t * \tanh C_t \tag{6.9}$$

Figure 6.6 illustrate how the architecture of an LSTM network works along with the previous explained gates and their connections. The described LSTM is the standard type, however, not all the LSTMs are the same and the right choice of type depends on the task of the network [Greff et al., 2016].

6.3 Proposed Solution

The proposed approach consists in finding the vehicle displacement by estimating the transformation between a sequence of 2D laser scanner acquisitions. From two consecutive observations, where each observation is a 360° set of points measured during one laser rotation, the network predicts the transformation $T = [\Delta d, \Delta\theta]$, which represents the travelling distance Δd and the orientation $\Delta\theta$ between two consecutive laser scans (s_{t-1}, s_t) . We only consider the 2D displacement of the vehicle, since we are relying only on a 2D sensor. Therefore, the goal is to learn the optimal function $g(.)$, which maps (s_{t-1}, s_t) to T at time t:

$$T_t = g(s_{t-1}, s_t) \tag{6.10}$$

Once we learn these parameters, we can obtain the 2D pose (x_t, y_t, θ_t) of the vehicle in time t as follows:

$$\begin{aligned} x_t &= x_{t-1} + \Delta d \sin(\theta_{t-1}) \\ y_t &= y_{t-1} + \Delta d \cos(\theta_{t-1}) \\ \theta_t &= \theta_{t-1} + \Delta\theta \end{aligned} \tag{6.11}$$

In this way, we can accumulate the local poses of the vehicle and estimate the global position of the vehicle at any time t. Since the algorithm does not perform any sort of loop closure, drift can be also accumulated, thus reducing the accuracy of the vehicle's localization.

The following subsections will present in details the proposed method. First, we show how the raw data from the laser scanner are encoded. Sequentially, we present the configuration of the network and the specifics of the training process.

6.3.1 Data encoding

We base our data encoding on the previous work [Li et al., 2017a], where the 2D laser scanner point set is encoded into a 1D vector. This can be done by first binning the raw scans into bins of resolution 0.1°. Sequentially, we calculate the average depth value of this group, since a group of points can fall into the same bin if the resolution of the sensor is less than 0.1°. Finally, considering all the bins of a 360° rotation range, we store the depth values into a 3601 size vector, where each possible bin angle average depth is represented by the elements in the vector. This process is presented in Figure 6.7.

Once we have processed two 1D vectors from sequential scans, we concatenate them to use as input for the network. In this way, we create a form of an image of size 2×3601 that represents two acquisitions of the laser scanner. This format allows to use standard convolutional layers to extract the features detected by the sensor in the surrounding environment.

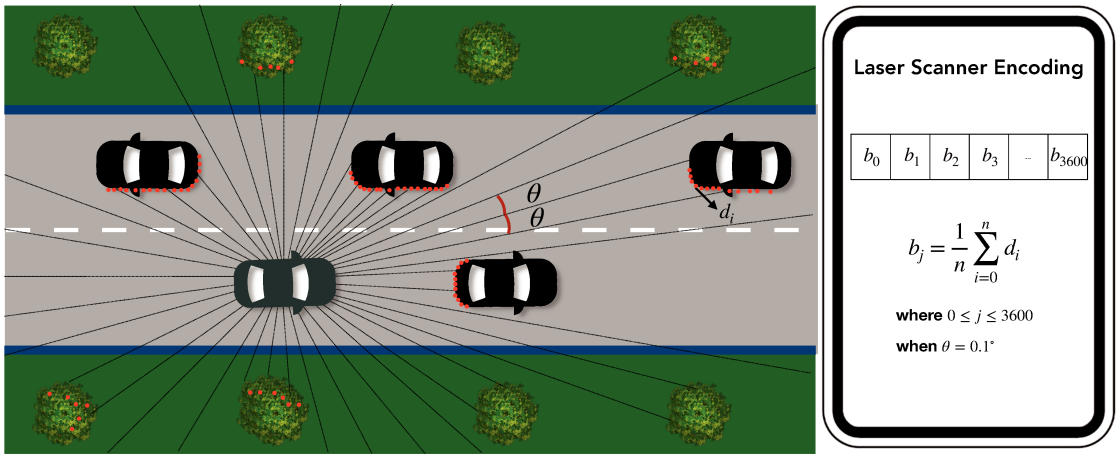


Figure 6.7: Data encoding for the 2D laser scanner with a 360° rotation range. At each time step the raw data from the laser scanner is separated into 0.1° bins. The average depth of all the points in a specific bin is calculated and stored into a vector. The result is a 3601 size vector that stores the depth values for each bin.

6.3.2 Network Architecture

The previous work on laser-based odometry estimation [Li et al., 2017a; Nicolai et al., 2016; Velas et al., 2018], for 2D and 3D sensors, only explored the use of CNNs without any deep temporal structure to estimate the local poses. We propose an architecture based on previous Visual Odometry (VO) methods, such as DeepVO [Wang et al., 2017], that not only extracts the features of the input data but also estimate the possible connections among consecutive image inputs. The use of RNNs is convenient for this purpose because of its ability of modelling sequential dependencies. By adding this to our model, we aim to increase the accuracy of the pose estimation by using implicit information that was detected by the laser scanner in previous frames. This process can be compared to graph-based approaches in classical SLAM methods [Grisetti et al., 2010], where it takes a sequence of poses features and outputs a more precise estimation of these poses.

Figure 6.8 presents the architecture of the proposed network. Two pre-processed laser scanner acquisitions, represented as a one dimensional vector of size 3601, are concatenated to create the input tensor of the network. Sequentially, the tensor is fed into the sequence of 1D convolutional and average pool layers to learn the features between the two acquisitions. Then, at each new data acquisition these features are received by the RNN to estimate new poses. We represent the 2D motion by two variables, the translation (travelling distance Δd) and rotation ($\Delta\theta$). The main goal of this process is to use the neural network to learn the features of laser scanner data while simultaneously matching them using the proposed combination of CNN and RNN. We inspire our network on the configuration used by DeepVO, which tries to achieve the same goal but using as input camera images. Since our data has a considerably smaller size, we adapted the configuration for this purpose. This new configuration is presented in Table 6.1. It has 6 1D convolutional layers, where each layer is followed by a rectified linear unit (ReLU) activation. Between each sequence of two convolutional

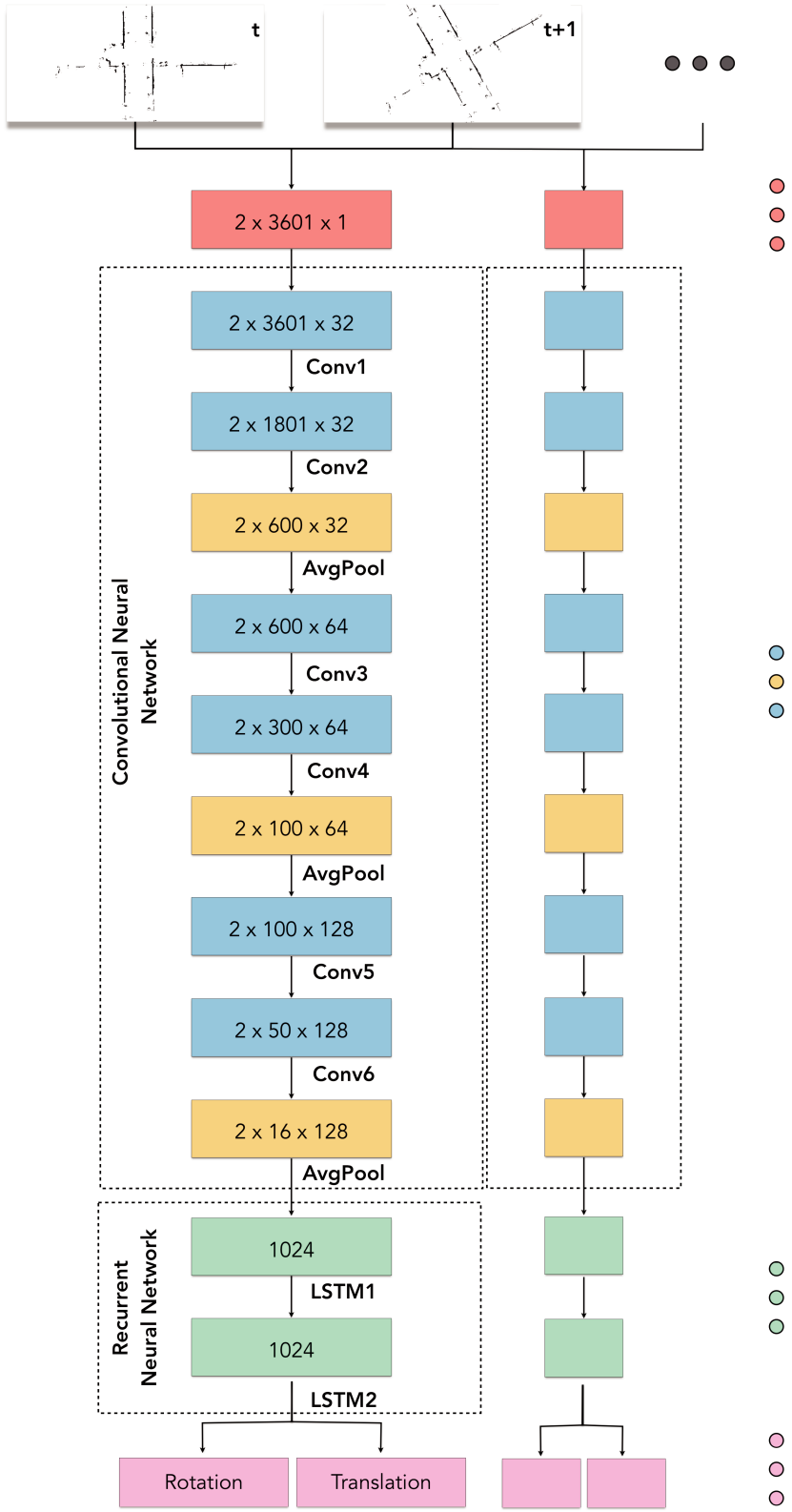


Figure 6.8: Architecture of the proposed RCNN. Each block of the illustration presents the size of the tensors considering that the input is two sequences of laser scanners concatenated after being encoded as presented in subsection 6.3.1.

Layer	Kernel Size	Stride	Number of Channels
Conv1	3	1	32
Conv2	3	2	32
Conv3	3	1	64
Conv4	3	2	64
Conv5	3	1	128
Conv6	3	2	128

Table 6.1: Configuration of the convolutional layers in the proposed network.

layers, there is also one average pool layer. We added the pooling layers to reduce computation complexity by extracting the most important features. We also tested both max and average pooling and we obtained better results applying the average pooling layer. Moreover, considering the size of the input and the size of features we can capture with this kind of sensor, we chose to use only the kernels of size 3 after testing them with different configurations such as 5 and 7 size kernels.

After we learned the features, the output of $Conv6$ is passed to the RNN for sequential modelling. We use as our RNN, Long Short-Term Memory (LSTM) units, which are able to learn long-term dependencies. The use of stacked LSTM layers is common to learn a high level representation and model complex dynamics [Wang et al., 2017]. For this reason, we use the configuration of two LSTM layers with the hidden states of the first LSTM being used as input for the second one. The two layers are defined with 1024 hidden states. Finally, the last LSTM layer outputs two values, the rotation and translation at each time step.

6.3.3 Training

The goal of using RNNs is to discover temporal correlations between the sequence of laser scans. While in principle the RNN is a simple and powerful model, in practice, it can be hard to train it properly to converge to precise results [Pascanu et al., 2013]. For this reason, the training of the network was performed in two steps. First, we trained the sequence of CNNs separately from the RNN. The objective in the first training is to pre-train the convolutional layers using no temporal information, only considering the information obtained from the two sequential laser scans input. Once we obtained the CNN part of the network pre-trained weights, we trained the complete network as presented in Figure 6.8.

6.3.3.1 CNN Pre-Training

To perform the pre-training, the output of the sequence of CNNs is fed to two different fully connected layers, one to estimate the rotation and another the translation. Sequentially, we train this convolutional network to learn the optimal function $g(.)$ presented in Equation 6.10.

In [Velas et al., 2018] the authors suggested that designing the network to regress the relative translation and rotation worked well only for translation, however the predicted

rotation was still inaccurate. They were able to obtain better results reformulating the problem as a classification task for the rotation, and continuing as a regression one for the translation. This is possible because the range of possible rotations between consequent frames is quite reasonable. Considering this, we tested two configurations to pre-train our network, as a regression-only task, and as a regression and classification task.

For the regression-only task, we performed the training based on the Euclidean loss between the ground truth and the estimated translation and rotation values, defining the complete loss function as follows:

$$\begin{aligned} &\mathcal{L} = \mathcal{L}_e(\Delta\hat{d}, \Delta d) + \beta\, \mathcal{L}_e(\Delta\hat{\theta}, \Delta\theta) \\ &\text{where } \mathcal{L}_e(\hat{x}, x) = \|\hat{x} - x\|_2 \end{aligned} \tag{6.12}$$

For the classification task, instead we used the Cross-entropy loss function to classify the angle, defining the new complete loss function for the training as:

$$\begin{aligned} &\mathcal{L} = \mathcal{L}_e(\Delta\hat{d}, \Delta d) + \beta\, \mathcal{L}_c(\Delta\hat{\theta}, \Delta\theta) \\ &\text{where } \mathcal{L}_c(x, class) = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) \end{aligned} \tag{6.13}$$

In Equation 6.12 and Equation 6.13, Δd and $\Delta\theta$ are relative ground-truth translation and rotation values, and $\Delta\hat{d}$ and $\Delta\hat{\theta}$ their output of the network counterparts. We use the parameter $\beta > 0$ to balance the scale difference between the rotation and translation loss values.

Considering all the possible variations of angles between two frames, we created classes in the interval $\pm 5.6°$ with $0.1°$ resolution, resulting in 112 possible classes. As indicated in [Velas et al., 2018], the results as a classification task for rotation were better compared to the only regression, and for this reason we used the CNN network trained as a classification for angle estimation as input for the RNN.

6.3.3.2 RCNN Training

After initializing the weights of the CNN layers at the pre-training stage, we train the complete RCNN network. We defined as the input of the RNN the output of the CNNs layers concatenated with the estimated result for rotation and translation, obtained from the pre-trained network, in a way that this could be used as a first estimation for the RCNN to refine the results.

For the RCNN we also tested the two different configurations (regression and regression with classification). Differently, we obtained more precise results treating the entire task (rotation and translation) as a regression problem, therefore using the loss function in Equation 6.12. We presume that this occurs because the estimation of the rotation as a regression was easier once we had a first estimation of the class using the pre-trained CNNs, and with the possible information obtained by the RNN from previous frames.

Seq	CNN-R	CNN-C	RCNN-R	RCNN-C	DeepVO	[Velas et al., 2018]
05	0.2888	0.2764	0.0293	0.2488	0.0262	0.0235
07	0.1281	0.0756	0.0218	0.1251	0.0391	0.0177
Mean	0.2084	0.1760	0.0255	0.1869	0.0326	0.0206
Time	0.005	0.005	0.015	0.015	1.0	0.7

Table 6.2: RMSE translation drift results for the two testing sequences along with the computation time per frame without GPU acceleration. We show the difference between the use of only CNNs and RCNN, the results for both of the RCNNs are using the pre-trained CNN-Classification network. In addition, we present the error for the different training configurations presented in subsection 6.3.3. We also compare the proposed approach with two other Deep Learning odometry estimation methods, one using as the sensor a monocamera [Wang et al., 2017] and the second using a 3D LiDAR [Velas et al., 2018]. However, the result presented for the 3D LIDAR method is for their training dataset, since they chose different sequences for testing.

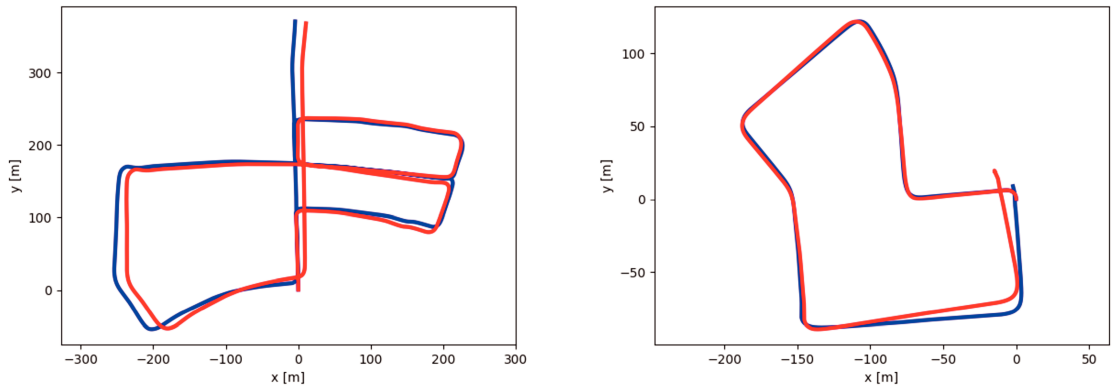
6.4 Experimental Results

For validation we use the KITTI dataset [Geiger et al., 2013], which provides several sequences in different conditions for outdoor environments. To obtain a 2D laser scanner dataset, we extracted one 360° layer from the Velodyne data. As mentioned before, the layer is extracted to simulate a low-cost 2D laser scanner.

We use 10 sequences from the KITTI odometry dataset for the proposed method. In this dataset there are 11 sequences, however, we eliminate the sequence 01 that consists of a trajectory mainly on a highway. During this sequence the 2D laser scanner detects almost no obstacle, making it impossible for the network to predict the odometry. Among the 10 sequences, we separate 8 for training and 2 for testing. We use for training the sequences 00, 02, 03, 04, 06, 08 and 09 and for testing the sequences 05 and 07. We chose these two sequences because they are not very long, leaving more data for the training, but they can still be challenging and present the potential of the proposed method. Additionally, during these two sequences we noticed that most of the time the simulated 2D laser scanner is able to detect obstacles, making it possible for the network to work as expected.

In order to validate our method and compare to other solutions, we calculated the drift according to the KITTI VO [Geiger et al., 2013] evaluation metrics, i.e., averaged Root Mean Square Errors (RMSEs) of the translational error for all subsequences (100, 200,..., 800 meters). In the same way as in the previous chapter, we adapt the proposed method to calculate the error only in 2D, since we only obtain 2D poses. The error score is then calculated by the mean of all subsequence errors.

Table 6.2 presents the error score for the testing sequences 05 and 07. The difference in scores between classification and regression shows why we chose to pre-train the convolutional layers as a classification task for the angle estimation, but to treat it as a regression task when we trained the entire RCNN. These results suggest that estimating



(a) Sequence 05 (b) Sequence 07

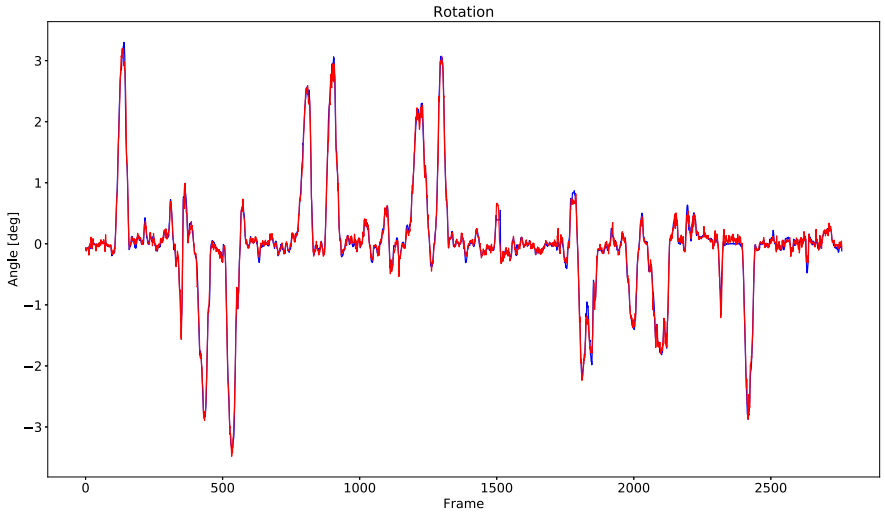
Figure 6.9: Trajectories of the two test sequences (05 and 07) applying the proposed method. The blue lines represent the ground truth trajectory, while in red the predicted one.

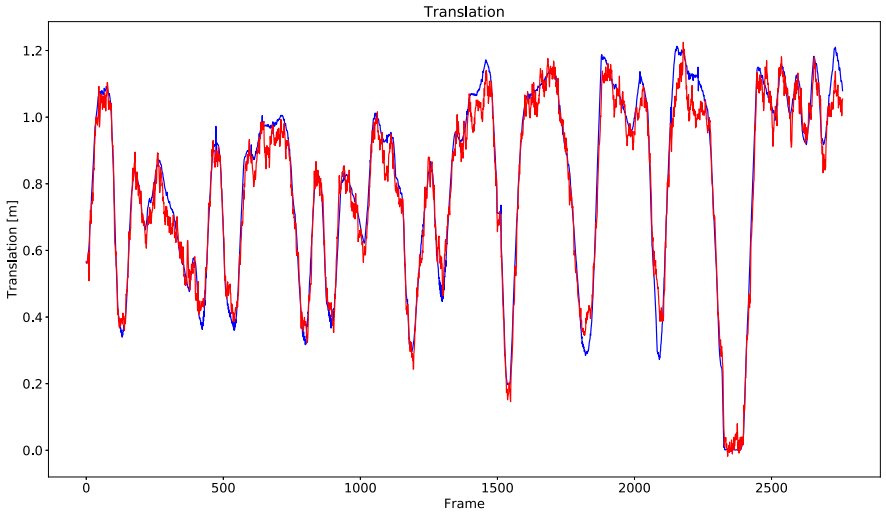
the angle as a regression task was too hard for the network to learn; however, once we had a first estimation about the class in the training of the RCNN, we were able to refine the value and obtain a more precise angle.

We also compare in Table 6.2 the proposed network with other two Deep Learning approaches for odometry estimation: one using the mono-camera and another a 3D laser scanner. We chose these two approaches because they use the same dataset, allowing us to compare the drift results, and also their computational time is presented by the KITTI benchmark. Our approach has better results in comparison to the network using only mono-camera images, however slightly worse results as the 3D laser scanner method. The better result in [Velas et al., 2018] is expected, since in our experiments we are extracting only one layer of the laser scanner, which provides a significant less amount of information. In addition, as the sensor is on top of the vehicle and we always extract the most parallel layer in relation to the vehicle, there are some frames where nothing or not much is detected, causing possible wrong estimations of translation and rotation. However, it is important to mention that the 3D laser scanner approach [Velas et al., 2018] takes 0.23s to run with GPU acceleration, while our approach takes only 0.015s per frame without using GPU acceleration (2,6 GHz Intel Core i5, Intel Iris 1536 MB), and it can be as fast as 0.001s per frame with GPU acceleration (4,0 GHz Intel Core i7, GeForce GTX 1060); i.e. a 230-fold increase in speed (GPU configuration), while obtaining only a 0.49% difference in drift score and using a much cheaper sensor. The faster processing is expected since we have a considerable smaller data input, and it allows to obtain odometry estimation in real-time with simple computational resources.

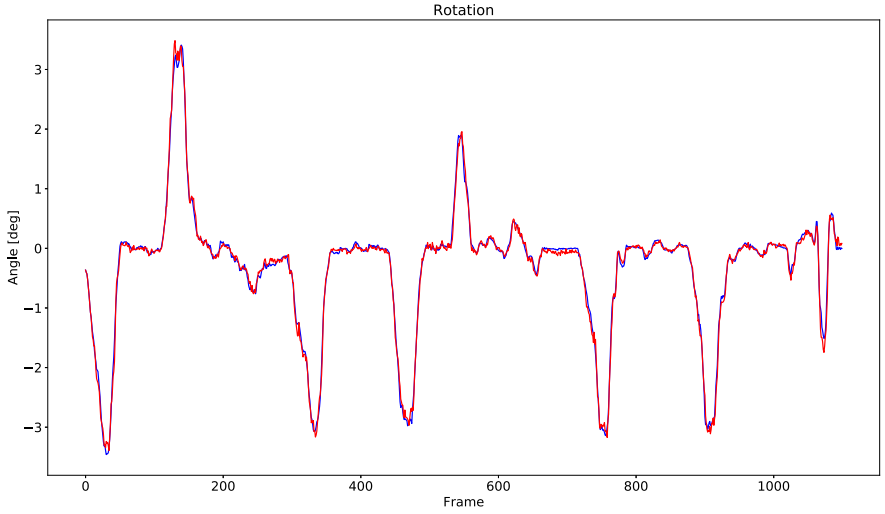
We can observe in Figure 6.9 that even with the eventual errors that can occur, we can still obtain a trajectory close to the ground truth. However, since the proposed approach does not perform any sort of loop closure, one eventual large error can be accumulated over time, generating a large drift like the one we have by the end of sequence 07.

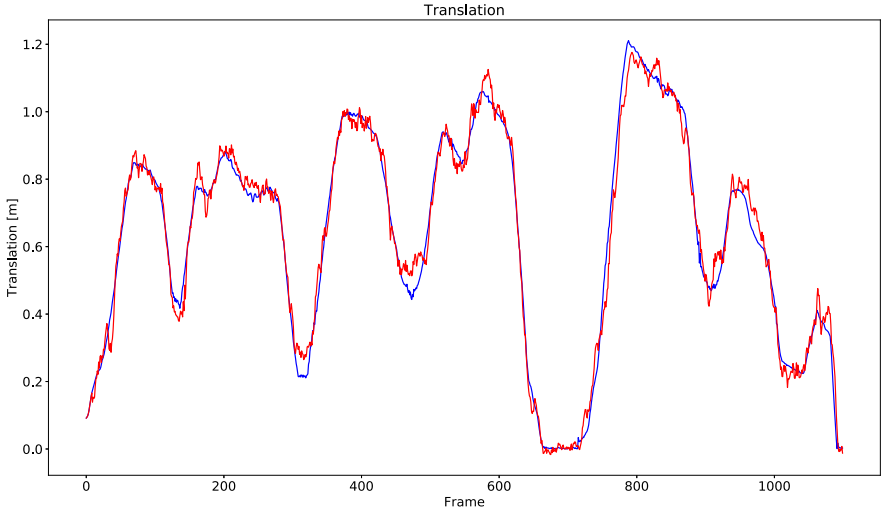
For this reason, a better way to understand the accuracy of the proposed approach is presented in Figure 6.10. It shows the odometry estimation (rotation and translation





(a) Sequence 05.





(b) Sequence 07

Figure 6.10: Results of rotation and translation estimation for the two testing sequences. The ground truth values are presented in blue and in red the output of the network.

increments) together with the ground truth for each frame of the testing sequences. Considering these two sequences, the average rotation absolute error is 0.05 degrees, while the average translation absolute error is 0.02 meters. However, we can encounter errors up to 0.4 degrees to rotation and 0.2 meters to translation in frames where it is harder to estimate the odometry. These values represent the odometry error per frame (not accumulated) and present how the network can most of the time estimate accurate odometry, however, there are still some difficult cases that can result in inaccurate values.

The results show how promising is the proposed method and it could be used as a complement to traditional localization methods for intelligent vehicle or any mobile robot, when for example there are no wheel encoders or GPS signal. We can also expect that if the sensor was located in an ideal position we could obtain even better results. For example an autonomous car with a set of 2D laser scanners around the vehicle in the level of the bumper. It is also important to mention that we trained the network with a relatively small dataset compared to other deep learning applications, therefore the result could be improved using more sequences for training.

6.5 Conclusion

In this chapter, we presented a novel approach based on RCNNs to estimate the odometry using only the data of a 2D laser scanner. The combination of CNNs and RNNs allows us to achieve in real-time the extraction of scan features and learn their sequential model to obtain the localization of an intelligent vehicle. The proposed network presents that the use of 2D laser scanners can not only provide good accuracy with a low cost sensor, but also requires less computational resources to achieve real-time performance.

The results were evaluated using the KITTI odometry dataset, making it possible to compare it with other Deep Learning approaches. Although the results were competitive for this type of approaches, we still do not expect that the deep learning methods could replace classic approaches at this moment, since they can still provide a better accuracy and a better understanding of the quality of the results. However, the proposed approach could be an interesting complement for classic localization estimation methods, since it can be run in real-time and could give relatively accurate values in systems where no wheel encoder data is provided or the GPS signal is absent. Moreover, the proposed method shows a promising use of Neural Networks to understand the environment detected by a 2D laser scanner, since we can assume that the network learned what were the best features to match between different scans.

In the next chapter, we will extend this proposed work by adding another sensor: a monocamera. The goal is to increase the accuracy of the method considering that the addition of the images of a camera can help the network to localize the vehicle in situations where there are not many obstacles detected by the 2D laser scanners.

Chapter 7

Deep Sensor Fusion for Odometry Estimation

Contents

7.1 Introduction 105
7.2 Proposed Solution 106
7.2.1 Data Pre-processing 106
7.2.2 Network Architecture 107
7.2.3 Training 109
7.3 Experimental Results 111
7.3.1 Ordinal Classification 111
7.3.2 Fusion vs Single Sensor Network 112
7.3.3 Comparison to state-of-the-art 113
7.4 Conclusion 115

Résumé du chapitre 7

Dans ce chapitre, nous présentons la première méthode qui utilise les réseaux de neurones convolutifs (CNN) pour l'estimation de l'odométrie en fusionnant des scanners laser 2D et des mono-caméras. L'utilisation de CNN permet non seulement d'extraire les caractéristiques des deux capteurs, mais également de les fusionner et les faire correspondre sans avoir besoin d'un étalonnage entre les capteurs. Nous présentons également une nouvelle méthode pour transformer le problème de régression d'estimation d'odométrie en une classification ordinale qui facilite l'apprentissage du réseau. Les résultats montrent que le réseau de fusion fonctionne en temps réel et est capable d'améliorer l'estimation d'odométrie d'un seul capteur en apprenant à fusionner deux types différents de données.

7.1 Introduction

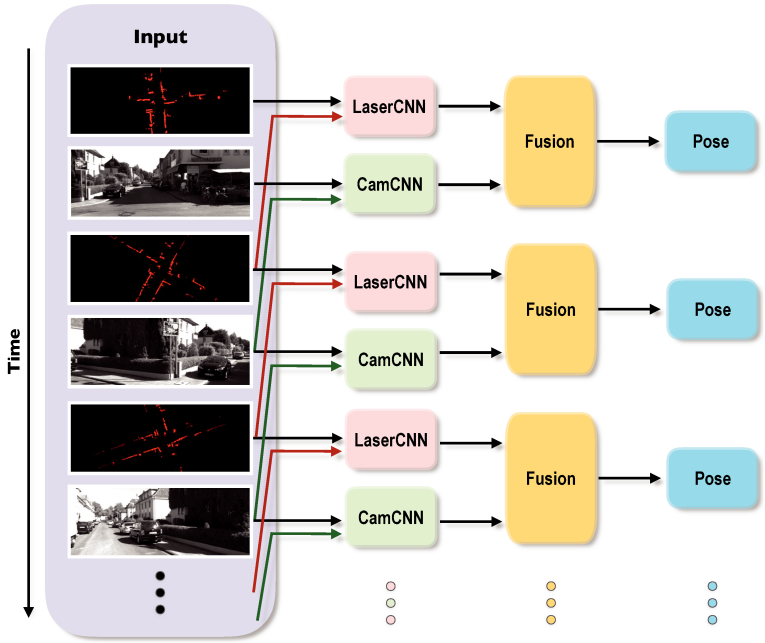


Figure 7.1: Overview of the proposed system. Consecutive laser scans and camera images are used, first in two separate networks and sequentially in a common network, in order to estimate the pose of the vehicle between consecutive frames.

In the previous chapter, we showed that neural networks can be a powerful tool for odometry estimation. We proposed a method that used as only input the data coming from a 2D laser scanner. However, one of the main drawbacks is that this type of sensor only detects a 2D slice of the world, which can be challenging to estimate odometry when there are not many obstacles. Thus, we propose in this chapter to perform sensor fusion in order to address this problem.

Cameras and 2D laser scanners, in combination, are able to provide low-cost, light-weight and accurate solutions, which make their fusion well-suited for many robot navigation tasks. However, correct data fusion depends on precise calibration of the rigid body transform between these two sensors.

In this chapter, we present the first framework that makes use of Convolutional Neural Networks (CNNs) for odometry estimation fusing 2D laser scanners and mono-cameras. The use of CNNs provides the tools to not only extract the features from the two sensors, but also to fuse and match them without needing a calibration between the sensors. A schematic of the proposed approach is presented in Figure 7.1. We also present a new method to transform the odometry estimation regression problem into an ordinal classification that facilitates the training of the network. Results show that the fusion network runs in real-time and is able to improve the odometry estimation of a single sensor alone by learning how to fuse two different types of data information.

7.2 Proposed Solution

The proposed approach consists in finding the vehicle displacement by estimating the transformation between a sequence of camera images and laser scanner acquisitions. From two consecutive observations, where each observation is a 360° set of points measured during one laser rotation and one camera image, the network predicts the transformation $T = [\Delta d, \Delta\theta]$, which represents the travelling distance Δd and the orientation $\Delta\theta$ between two consecutive laser scans (s_{t-1}, s_t) and camera images (c_{t-1}, c_t). Therefore, the goal is to learn the optimal function $g(.)$, which maps the fusion between (s_{t-1}, s_t) and (c_{t-1}, c_t) to T at time t:

$$T_t = g((s_{t-1}, s_t), (c_{t-1}, c_t)) \tag{7.1}$$

In the same way, as the method presented in the previous chapter, once we learn the network parameters, we can obtain the 2D pose (x_t, y_t, θ_t) of the vehicle in time t as follows:

$$\begin{aligned} x_t &= x_{t-1} + \Delta d \sin(\theta_{t-1}) \\ y_t &= y_{t-1} + \Delta d \cos(\theta_{t-1}) \\ \theta_t &= \theta_{t-1} + \Delta\theta \end{aligned} \tag{7.2}$$

In this way, we can accumulate the local poses of the vehicle and estimate the global position of the vehicle at any time t. Since the algorithm does not perform any sort of loop closure, drift can be also accumulated, thus reducing the accuracy of the vehicle's localization. The main goal is to explore the use of CNNs to match laser scans and camera images between two consecutive frames for odometry estimation, and especially, to prove that the fusion between the two of them for this purpose is possible to be executed by neural networks.

The following subsections will present in details the proposed method. First, we show how the raw data of the sensors are pre-processed. Sequentially, we present the configuration of the network and the specificities of the training process.

7.2.1 Data Pre-processing

The raw data coming from the two sensors need to be prepared before they can be used as an input for the neural network. For the laser scanner we use the same data encoding presented in the previous chapter. The sensor point set is encoded into a 1D vector, which is created by binning the raw scans into bins of resolution 0.1°. Since many points can fall into the same bin, we calculate the average depth value. Finally, considering all the bins of a 360° rotation range, we store the depth values into a 3601 size vector, where each possible bin angle average depth is represented by the elements in the vector. After two sequential laser scans are encoded as 1D vectors, we concatenate them to create the input of the laser scanner network. The idea is to create a sort of image of size $(2, 3601)$, allowing to use standard convolutional layers to extract the features detected by the sensor in the surrounding environment.

For the camera raw data, we only resize the images in order to reduce the computational time. We tested different sizes so that the accuracy of the method was not reduced, but we would still be able to produce faster results. Considering this, the best trade-off for accuracy and time was achieved with the image size $(416, 128)$. After resizing them, two consecutive images are stacked together, in the same way of the scans, to form a tensor that represents two camera acquisitions.

7.2.2 Network Architecture

The architecture of the proposed network is presented in Figure 7.2. The input consists in the pre-processed raw data from the two sensors as explained in the previous subsection, while the rest of the network can be separated in the three main parts explained bellow: the laser scanner (CNN-Laser), the camera (CNN-Cam) and finally the fusion. Differently from the previous chapter, we chose to simply use CNNs without recurrent networks. We concluded that the fused network was more stable than the precedent and the addition of a RNN did not improve considerably the results.

CNN-Laser

Two pre-processed laser scanner acquisitions, represented as a one dimensional vector of size 3601, are concatenated to create the input tensor of the network. Sequentially, the tensor is fed into the sequence of 1D convolutional and average pool layers to learn the features between the two acquisitions. We use the same CNN configuration presented in the previous chapter. However, we add a linear layer at the end of the network to have an input on the same size of features from the laser scanner and the camera to be concatenated.

CNN-Cam

The configuration of the camera network is the CNN part of the RCNN proposed by DeepVO [Wang et al., 2017]. However, we use a smaller input as explained before in the pre-processing. This network configuration is inspired by the network for optical flow estimation. The use of CNNs is ideal for this application since it learns geometric features and not the appearance or visual context of the images, making it ideal for a task that need to be applied in unknown environments. In the same way of the CNN-Laser, we added an extra linear layer to reduce the tensor size before the fusion. Therefore, the input is composed by two raw camera images and the output by the reduced features detected by the sequence of CNNs to be then fused with the features detected in the CNN-Laser.

Fusion

After extracting features from consecutive laser scans and camera images using the previously described CNNs, we concatenate their outputs in order to estimate the pose of the vehicle. The concatenated features are fed to two different sequences of linear layers. We tested the use of same linear layers for both rotation and translation estimation together, however better results were found once we separated them. In order to avoid the overfitting problem, the two linear layers are preceded by Dropout layers.

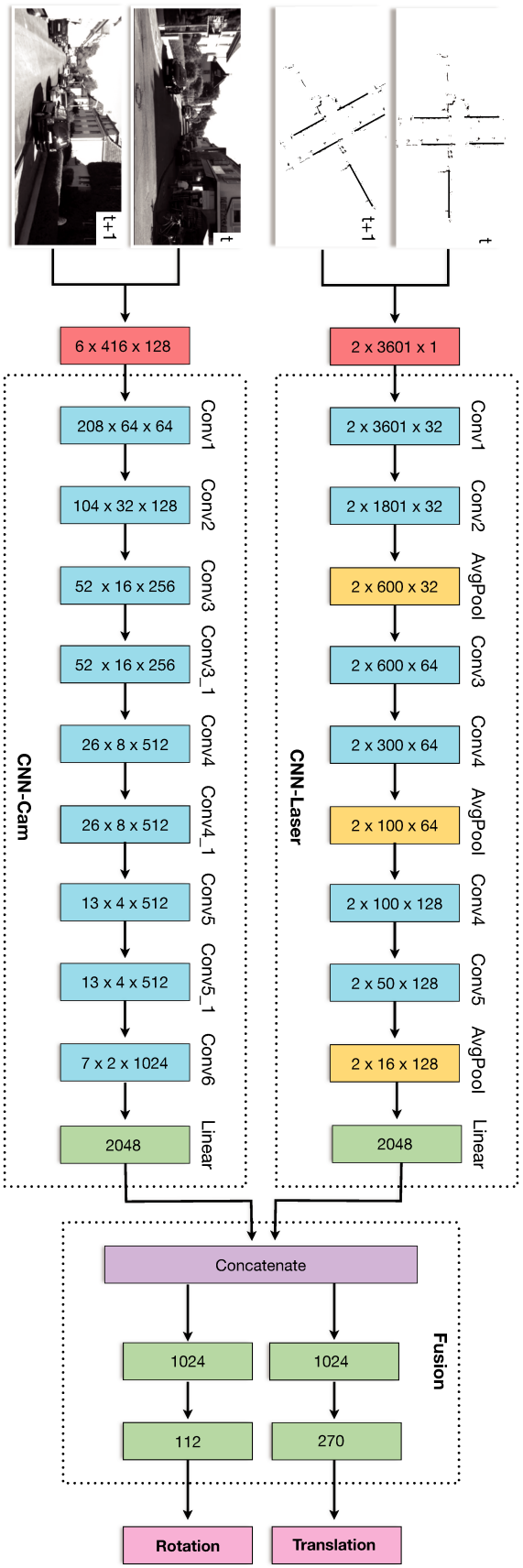


Figure 7.2: Architecture of the proposed network. Each block of the illustration presents the size of the tensors considering that the input is the raw sensor data pre-processed as presented in subsection 7.2.1.

7.2.3 Training

The training of the network was performed in two steps; first, we trained the single sensor CNNs separately to find the best pre-trained weights possible for those networks. For this purpose, we connected them to two separate linear layers, one for the rotation and one for the translation. After it, the pre-trained CNNs are connected to the fusion part of the network and we perform the final training step.

In the previous chapter, we prove that the CNN network get better results if we reformulate the problem as a classification task for the rotation, and continue it as a regression one for the translation. Considering all the possible variations of angles between two frames, we created classes in the interval $\pm 5.6^{\circ}$ with 0.1° resolution, resulting in 112 possible classes.

In this chapter we propose to not only treat the rotation as a classification task, but also the translation in order to facilitate the training. We observed that in all the possible sequences of the KITTI dataset the maximum translation between two frames is around 2.7 meters and the minimum 0.0. Therefore, we created 270 classes for this interval considering a resolution of 0.01 meter. Results showed that it became easier for the network to converge and the accuracy was not reduced by this transformation.

The main problem of transforming the rotation and translation in a classification task is that no order about the data is learned by the network. This happens because machine learning methods for classification problems commonly assume that the class values are unordered. For example, it would not be possible to understand that a difference of 2 degrees was higher than only 0.1 degrees. In [Li and Lin, 2007] the authors introduce a simple method that enables standard classification algorithms to make use of ordering information in class attributes, known as ordinal classification. The idea is to transform the ordinal regression problem into a series of simpler binary classification subproblems. Inspired by this work, the authors of [Niu et al., 2016] applied this idea to solve simpler binary classifications for age estimation by the use of CNNs.

In order to transform our rotation and translation classes into a series of binary classification subproblems it is necessary to change how we label the dataset. Instead of labeling directly with a class that represents the ground truth value, the samples are labeled by an ordinal scale called the rank. Considering k the number of possible classes, we are going to transform the problem into k simpler binary classification subproblems. Specifically, for one sample i each subproblem receives a label $l_i^k \in \{0, 1\}$ indicating if the sample class l_i is larger than r_k as follows:

$$l_i^k = \begin{cases} 1, & \textit{if } (l_i > r_k) \\ 0, & \textit{otherwise.} \end{cases} \tag{7.3}$$

The rank format allows the network to learn the order of the classes by showing that one value is smaller or larger than the other. This type of format can be applied to any kind of classification where the order of the classes is important. A summary of the ordinal classification labeling process is presented in Figure 7.3.

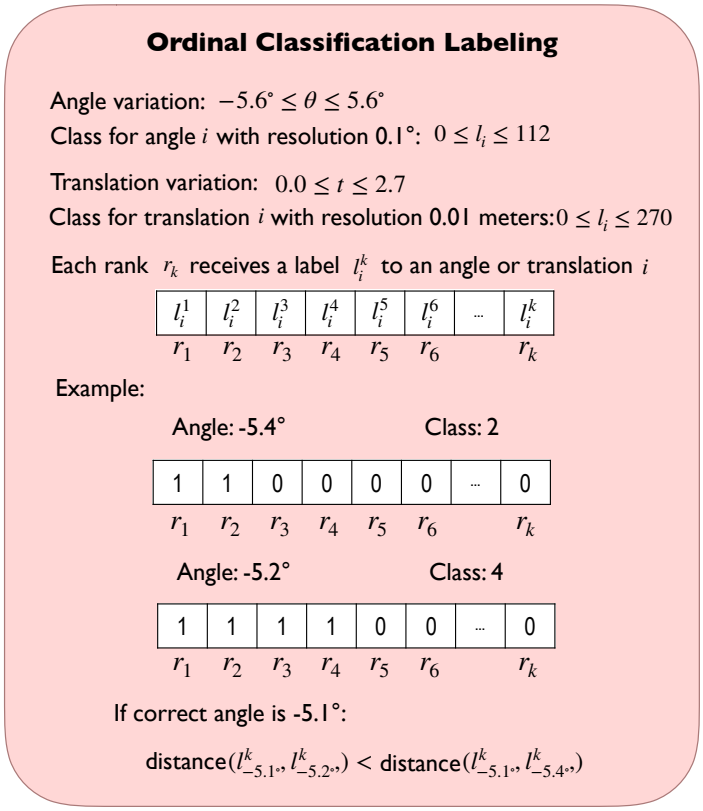


Figure 7.3: Summary of the ordinal classification labeling with an example for the angle classification.

To solve the binary classification subproblems, we calculate the Binary Cross Entropy between the target and the output for rotation and translation and we sum them as follows:

$$\begin{aligned} &\mathcal{L} = \mathcal{L}_{BCE}(\Delta\hat{d}, \Delta d) + \beta\, \mathcal{L}_{BCE}(\Delta\hat{\theta}, \Delta\theta) \\ &\text{where } \mathcal{L}_{BCE}(x, y) = -\sum_k y_k \log(x_k) + (1 - y_k) \log(1 - x_k) \end{aligned} \tag{7.4}$$

Δd and $\Delta\theta$ are the relative ground-truth translation and rotation transformed to rank labels (ordinal classification format), and $\Delta\hat{d}$ and $\Delta\hat{\theta}$ their output of the network counterparts. $\Delta\hat{d}$ and $\Delta\hat{\theta}$ pass by a Sigmoid function before the loss function for numerical stability. We use the parameter $\beta > 0$ to balance the scale difference between the rotation and translation loss values.

Considering this, our network will have a multiple output structure where each output corresponds to a binary classifier. Therefore, we can predict during the testing the class label l for translation or rotation as follows,

$$l = \sum_{k=0}^{K} f_k(s), \tag{7.5}$$

where $f_k(s) \in \{0, 1\}$ is the classification result of the k-th binary classifier for the sample s and K is the number of possible classes. Ideally the output should be

consistent as a sequence of ones followed by zeros. During our tests, we observed that this is easily learned during the training of the network. For this reason, we assume we can apply the Equation 7.5 without ensuring the consistency among the different classifiers as in [Niu et al., 2016].

The network is implemented on the framework PyTorch and the Adam optimizer is applied with learning rate equal to 0.0001. As recommended in [Wang et al., 2017], during the pre-training of the CNN-Cam, we initialize it with pre-trained FlowNet model weights to reduce the training time. We could observe that the pre-training of the network along with the use of pre-trained FlowNet weights increased the quality of the results and reduced the complete training time of the network.

7.3 Experimental Results

For validation we use the same dataset as the previous chapter(the KITTI dataset [Geiger et al., 2013]), which provides several sequences in different conditions for outdoor environments. To simulate a 2D laser scanner data, we extracted one 360° layer from the Velodyne data, while for the camera we use the RGB raw images provided by the dataset (after the downscaling described in subsection 7.2.1). We use the 11 sequences from the KITTI odometry dataset which contain ground truth values to train and test the proposed method. Among the 11 sequences, we separate 8 for training and 3 for testing. We use for training the sequences 00, 02, 03, 04, 05, 06, 08 and 09 and for testing the sequences 01, 07 and 10. We chose these three sequences because they are not very long, leaving more data for the training, but they can still be challenging and present the potential of the proposed method. In the previous chapter, we could not evaluate the sequence 01 because the vehicle is on a highway where the 2D laser scanner could not detect obstacles most of the time, making it impossible for a laser-only network to predict the odometry. For this reason, we add this sequence specially to observe if the fusion could improve the results.

7.3.1 Ordinal Classification

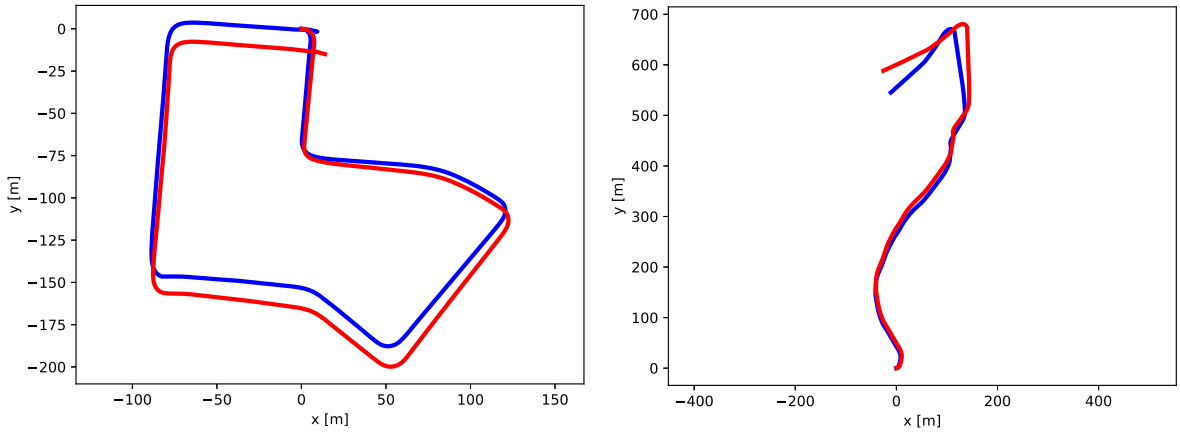
In subsection 7.2.3 we proposed to estimate both rotation and translation values using the method known as Ordinal Classification. The idea is to take advantage of the simplicity of training a network for classification instead of regression, but at the same time, to not lose the notion of order in our data. Therefore, the first test we performed was to analyze if we were able to obtain better results using this approach.

In order to compare standard classification and ordinal classification, we trained the CNN-Laser network in two ways: first, we transformed the translation and rotation values into classical classes and used the Cross-Entropy loss function to train the network; sequentially, we trained another network with the proposed approach described in subsection 7.2.3.

In Table 7.1 we can observe that the network trained with Ordinal Classification obtained better accuracy in both rotation and translation by analyzing the average rotation absolute error (σ_r) and the average translation absolute error (σ_t). Considering

Sequence	Standard Classification		Ordinal Classification	
	σ_r	σ_t	σ_r	σ_t
07	0.08	0.05	0.06	0.03
10	0.11	0.07	0.08	0.04

Table 7.1: Average rotation absolute error σ_r (degrees) and the average translation absolute error σ_t (meters) results for the CNN-Laser trained with Standard Classification and with Ordinal Classification



(a) Sequence 07

(b) Sequence 10

Figure 7.4: Trajectories of two test sequences (07 and 10) applying the proposed CNN-Fusion. The blue lines represent the ground truth trajectory, while in red the predicted one.

these results, we chose to use the CNN-Laser trained with this configuration and train the CNN-Cam and CNN-Fusion in the same way.

7.3.2 Fusion vs Single Sensor Network

We can observe in Figure 7.4 that even with eventual errors in the odometry estimation, we can still obtain a trajectory close to the ground truth. However, since the proposed approach does not perform any sort of loop closure, and does not even use temporal information by recurrent neural networks, one eventual large error can be accumulated over time, generating a large drift like the one we have at the end of sequence 10.

For this reason, a better way to understand the accuracy of the proposed approach is presented in Table 7.2 and in Figure 7.5. Table 7.2 shows the average rotation absolute error σ_r and the average translation absolute error σ_t for the single sensor CNNs and the CNN-Fusion. These values represent the odometry error per frame (not accumulated). We can observe that in all the cases the result of the fusion was equal or better than the result of the single sensor network. Specially for the rotation estimation, the fusion of the features was able to increase the accuracy in all of the sequences, which proves that the network was able to learn how to perform the fusion of the two sensors. In the sequence 01, which is the hardest sequence because of the vehicle's velocity and lack of

Sequence	CNN-Cam		CNN-Laser		CNN-Fusion	
	σ_r	σ_t	σ_r	σ_t	σ_r	σ_t
01	0.09	0.30	0.30	0.33	0.06	0.30
07	0.06	0.06	0.06	0.03	0.04	0.03
10	0.06	0.11	0.08	0.04	0.05	0.04

Table 7.2: Average rotation absolute error σ_r (degrees) and the average translation absolute error σ_t (meters) results for the single sensor CNNs, CNN-Cam and CNN-Laser, and the result after CNN-Fusion.

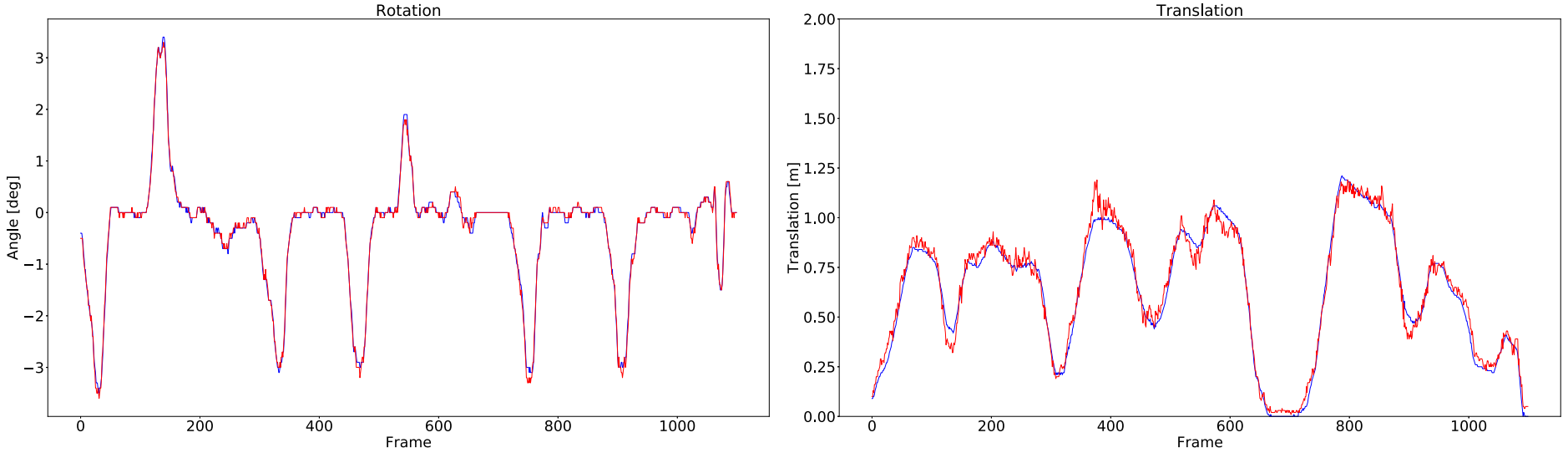
features to detect, it is clear how the laser was not able to estimate the angles because of the few detected points, but after the fusion with the camera the network was able to estimate more accurate angles. However, the translation was still inaccurate because this is the only sequence in the training dataset where the vehicle has a high velocity, therefore there were not other samples for it to learn the translation classes for this case.

Figure 7.5 presents the odometry estimation (rotation and translation) together with the ground truth for each frame of the sequences 07 and 10. These values present how the network can most of the time estimate accurate odometry, however, there are still some difficult cases that can result in inaccurate values. For example, in the sequence 10 we can notice that the highest angle (around frame 900) was not properly estimated, this probably happened because there are not a lot of samples of this type of rotation, making it hard for the network to learn this rotation class. We can expect that training this type of network with a larger dataset, with more samples of the challenging cases, could possibly resolve this type of problems and increase the accuracy.

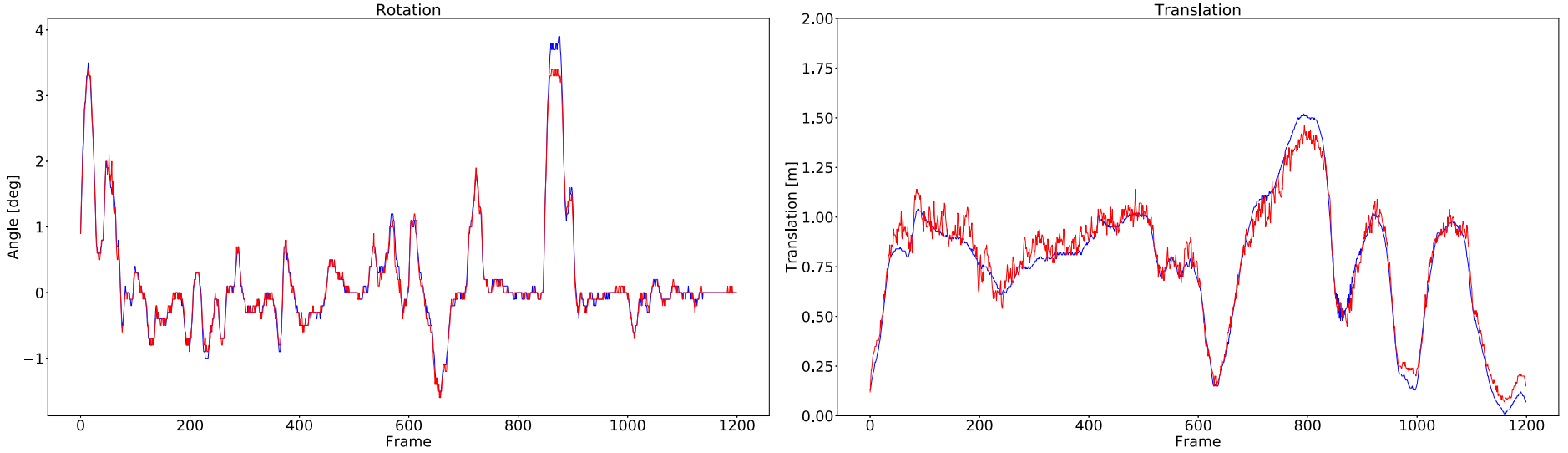
The results show how promising is the fusion between sensors by the use of CNNs, and that the proposed method could be used as a complement to traditional localization methods for intelligent vehicles or any mobile robot. It is also important to mention that we trained the network with a relatively small dataset compared to other deep learning classification tasks, therefore the result could be considerably improved using more sequences for training.

7.3.3 Comparison to state-of-the-art

Finally, in order to validate our method and to compare to the solution DeepVO [Wang et al., 2017], we calculated the drift according to the KITTI VO Geiger et al. [2013] evaluation metrics, i.e., averaged Root Mean Square Errrors (RMSEs) of the translation and rotation error for all subsequences (100, 200,..., 800 meters). However, like in the previous chapter, we need to adapt our 2D results to be able to compare to the 3D errors of DeepVO, therefore we create 3D poses from our 2D values by giving 0.0 to the values we do not estimate (lateral and longitudinal angles and translation in the vertical axis).



(a) Sequence 07



(b) Sequence 10

Figure 7.5: Results of rotation and translation estimation for the two testing sequences. The ground truth values are presented in blue and in red the output of the network.

Sequence	CNN-Fusion		DeepVO	
	t_{rel}	r_{rel}	t_{rel}	r_{rel}
07	2.03	0.85	3.91	4.60
10	7.60	2.80	8.11	8.73
Mean	4.41	1.82	6.01	6.66
Computation Time (s/frame)	0.1		1.0	

Table 7.3: Average translation (%) and rotation (degree/100m) RMSE drift on trajectory lengths of 100 to 800 meters, along with the computation time per frame, for the proposed approach and the DeepVO method.

Table 7.3 presents the translation and rotation error score for the testing sequences 07 and 10 and we compare these values between the proposed approach and the method DeepVO, along with their computational times. Even though we transform our results in 3D, it is important to mention that the results are not yet directly comparable, but we can still estimate if the order of error is around the same and we can compare the computation time of both methods. For the translation error the comparison is easier to perform because the global translation in the vertical axis is very small and not significant compared to the other axes. On the other hand, for the angle comparison, since most of the time the 3 angles are around 0 degrees (the only high rotation rates are on z axis during turns, which represent a very small part of a complete trajectory), all their drifts are in the same order of magnitude and should be relevant for the comparison. For this reason, it is expected that the DeepVO method has a rotation error around three times more than our solution. As a result of these difficulties in the comparison of the two methods, we can only really extract from these results that the errors are around the same order of magnitude, but we are still providing a solution that is 10 times faster using no GPU acceleration (2,6 GHz Intel Core i5, Intel Iris 1536 MB), and can be as fast as 0.01s (per frame) with GPU acceleration (4,0 GHz Intel Core i7, GeForce GTX 1060). This faster processing is mainly due to the fact that we resize the images for smaller sizes and we use only CNNs instead of a RCNN.

7.4 Conclusion

In this chapter, we presented the first Deep Learning approach for sensor fusion to odometry estimation. We used as input only 2D laser scanner data and camera images to match their features in order to determine the translation and rotation of the vehicle using sequences of CNNs. The proposed network presents that the fusion between the sensors is possible by applying a purely CNN method and we can obtain good accuracy using only low cost sensors. We also introduced a new form of treating the odometry problem in deep learning methods. We transform the regression task into smaller binary classification subproblems that facilitates the training of the network.

We evaluated the results using the KITTI odometry dataset, making it possible to compare to other approaches. The results showed competitive accuracy compared to other deep learning methods. However, classic approaches can still provide better results and a better understanding of the quality of their outputs. In spite of that, the proposed approach could be an interesting complement for classic localization estimation methods, since it can be run in real-time and could give relatively accurate values in systems where no wheel encoder data is provided or the GPS signal is absent. Moreover, the proposed method presents that the use of Neural Networks is possible to perform the fusion between 2D laser scanners and mono-cameras, and this method could be used for other tasks in robotic systems.

Even with these promising results, we still can observe that the lack of drift correction is still the major problem of our deep learning approaches for localization so far. Considering this, we expand in the next chapter the deep odometry estimation with the use of global maps to reduce the drift error created over time.

Chapter 8

Deep Learning Localization in 2D Laser Maps

Contents

8.1 Introduction 119

8.2 Proposed Solution 120

8.2.1 Data Creation 121

8.2.2 Data augmentation 123

8.2.3 Network and Training Configuration 123

8.3 Experimental Results 126

8.4 Conclusion 129

Résumé du chapitre 8

Nous proposons dans ce chapitre une nouvelle méthode d'apprentissage profond qui explore l'utilisation des réseaux de neurones convolutifs (CNN) pour estimer l'odométrie d'un véhicule autonome tout en corrigeant la dérive liée à une carte globale. Le seul capteur utilisé dans cette approche est un scanner laser 2D. L'entrée du réseau est composée des données du scanner laser transformées en images de la carte globale de l'environnement. Les deux images de scan consécutives permettent d'estimer l'odométrie du véhicule, tandis que la carte globale permet au réseau de corriger toute dérive éventuelle créée au cours du temps lors de sa trajectoire. La carte globale de l'environnement est construite au préalable en utilisant le modèle évidentiel, qui nous permet de supprimer d'éventuels obstacles dynamiques et ainsi d'avoir une carte propre de l'environnement ne contenant que les informations statiques.

8.1 Introduction

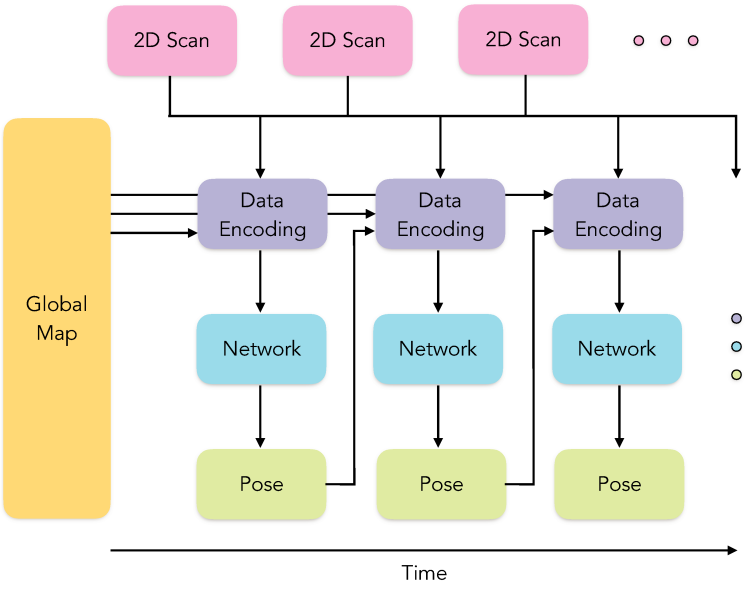


Figure 8.1: Overview schematics of the proposed system. At each timestamp the current laser scans and the previous one are encoded along with the accumulated global map to be fed into a neural network that will estimate the localization of the vehicle inside the global map. The pose of the previous frame is used in the data encoding to define the region of the global map based on the estimated location of the vehicle up to that moment.

In chapter 6 and chapter 7, we showed the potential of neural networks for the estimation of the vehicle's localization. The two proposed solutions were able to provide an odometry estimation with good accuracy in difficult dynamic environments, proving that the network was able to deal with the main difficulties found in localization methods. However, the main drawback of these approaches was that odometry errors were accumulated overtime, and since no loop closure was performed, we could lose the localization the vehicle.

Considering this, we propose in this chapter a novel Deep Learning method that explores the use of Convolutional Neural Networks (CNNs) to estimate the odometry of an autonomous vehicle while correcting the drift related to a global map. The only sensor used in this approach is a 2D laser scanner. We chose to keep only this sensor considering what we discussed on chapter 5: the 2D laser scanner grid maps are more precise for localization than the ones with the camera and can provide a better map-based localization. Therefore, the input of the network is composed by the laser scanner data transformed to images along with images from the global map of the environment. The two local scan images are used to estimate the odometry of the vehicle between two sequential scan acquisitions, while the global map allows the network to correct any possible drift created over time during its trajectory. The global map of the environment is constructed offline using the evidential model, which allow us to remove possible dynamic obstacles in order to have a clear map of the environment with the static information. The schematic of the proposed approach is presented in Figure 8.1.

8.2 Proposed Solution

The main goal of the proposed approach is to correct the accumulated drift we observed in the previous chapters. Considering this, the approach provides the localization of the vehicle using a 2D laser scanner and a global map of the environment created with the same sensor. At each sensor acquisition the data from the sensor along with the global map of the environment are transformed into grayscale images and are fed into a sequence of CNNs that will extract their features. Sequentially, we use linear layers to first estimate the odometry between t and $t-1$, which will provide travelling distance Δd (considering no sideways motion) and rotation $\Delta\theta$ values. Once we estimate these values, we can obtain the 2D pose (x_t, y_t, θ_t) of the vehicle in time t as follows:

$$\begin{aligned} x_t &= x_{t-1} + \Delta d \sin(\theta_{t-1}) \\ y_t &= y_{t-1} + \Delta d \cos(\theta_{t-1}) \\ \theta_t &= \theta_{t-1} + \Delta\theta \end{aligned} \tag{8.1}$$

Similarly to the previous odometry estimation methods, we can accumulate the local poses of the vehicle and estimate its global position at any time t. However, small errors can occur in the estimation of Δd and $\Delta\theta$. The accumulation of these errors can end up generating a high drift in long sequences and we can lose completely the localization of the vehicle. For this reason, we add another output to our network: the translation and rotation values for drift correction. At the same time that the network estimates the odometry, it will also estimate the drift between the scan $t-1$ and the global map, providing 2D translation (Δx_{drift} and Δy_{drift}) and rotation ($\Delta\theta_{drift}$) values. This method is based on the previous work presented on chapter 5, where we use traditional image matching methods instead of CNNs to correct the drift in localization. Therefore, after learning the output parameters for odometry and drift, we can obtain the corrected 2D pose (x_t, y_t, θ_t) of the vehicle in time t as follows:

$$\begin{aligned} x_t &= x_{t-1} + \Delta d \sin(\theta_{t-1}) + \Delta x_{drift} \\ y_t &= y_{t-1} + \Delta d \cos(\theta_{t-1}) + \Delta y_{drift} \\ \theta_t &= \theta_{t-1} + \Delta\theta + \Delta\theta_{drift} \end{aligned} \tag{8.2}$$

It is important to mention that the drift correction is directly dependent on the quality of the mapping process. Therefore, if the map is created with large odometry errors, it will be harder to correct the drift. Considering this, the best results are possible when creating a 2D laser scanner map of the environment offline with more precise localization and after using the proposed network to re-localize the vehicle inside this environment.

The following subsections will present in details the proposed method. First, we show how the raw data of the laser scanner and the global map are transformed into grayscale images to be used as input for the network. Sequentially, we present the configuration of the network and the specifities of the training process.

8.2.1 Data Creation

The input of the network consists of three different images of size $(256, 256)$ concatenated. The two first images correspond to the current t and previous $t-1$ 2D scans, while the third one is the global map cropped in the estimated position at time $t-1$. The three images represent in grayscale the environment around the vehicle during time $t-1$ and t. The objective is to provide enough information for the network to learn how to predict the odometry between these two timestamps and also if there is any accumulated drift related to the global map.

1. **Scan images**

   The first two images are created by projecting the current and previous 2D scan into an occupancy grid image, which is a grid that stores the obstacles detected by the sensor. The scan corresponds to a set of points measured during one laser rotation. We assume that the rotation is fast enough to map the scan at the same time on the grid. Considering a fixed resolution, we set the pixels corresponding to the points detected by the laser scanner at the grid image with value 255, that will represent the obstacles around the vehicle. The remaining of the pixels stays with the value 0, creating in this way a black and white picture that represents the current and previous detected obstacles by the vehicle.

2. **Global map image**

   The third image represents the global map of the environment. It is created using the method proposed in the previous work [Moras et al., 2011]. This approach creates evidential grid maps, which is an occupancy grid based on the Dempster-Shafer theory (DST) [Shafer, 1976], that stores at each cell a basic belief assignment (BBA) with four beliefs $[m(F), m(O), m(\Omega), m(\emptyset)]$. Each belief represents respectively the evidence of being free, occupied, unknown or conflict.

   In order to create the evidential grid map, we first create a local grid map at each timestamp in which each cell receives occupied evidence where there are laser impacts and free evidence where there are crossed cells. Sequentially, to create the global map with all the information from the sensor gathered at different timestamps, we apply a grid fusion process. This process is done in the same way as the evidential grid maps created in chapter 5. The current evidential grid is merged to a global grid using the Dempster's combination rule. It consists in first applying the conjunctive rule of combination denoted by $\textcircled{\cap}$ in (8.3), and then normalizing the masses as presented in (8.4), where $m1$ and $m2$ represent the two evidential occupancy grids. This operator distributes the belief from the conflict to the other states, giving more importance to the state which has the highest mass.

$$(m_1 \textcircled{\cap} m_2)(A) = \sum_{A=B \cap C} m_1(B) \cdot m_2(C) \tag{8.3}$$

$$m_1 \oplus m_2(A) = \begin{cases} \frac{(m_1 \bigcirc\!\!\!\!\bigcap m_2)(A)}{1-(m_1 \bigcirc\!\!\!\!\bigcap m_2)(\emptyset)} & \forall A \subseteq \Omega \wedge A \neq \emptyset \\ 0 & A = \emptyset \end{cases} \tag{8.4}$$

We chose the Evidential model because it allows us to distinguish better the static obstacles, with high evidence of occupied, from the dynamic obstacles, that will have a high evidence of conflict. This is important, because we want the network to compare the current scan to the static information from the global map to estimate the odometry. We considered to use the life-long grid proposed at the chapter 5, however the network obtained better results using the global map with the smoother information of the occupied belief evidence. Therefore, our third image input is a greyscale image created directly from the values of occupied belief from the global occupancy grid map.

After we create an evidential global map of the environment, there are four main steps to create the global map image:

(a) The occupied belief values of the evidential grid maps are extracted around the location of the vehicle at time $t-1$. The global image region size used for the network is 256×256, like in the scan images. However, during the data creation for the training of the network, we extract a region of 512×512 to be able to perform data augmentation, this process will be explained in details later.

(b) The mass belief of occupied space of the cropped region, which are between 0.0 and 1.0, are multiplied by 255.

(c) A Bilateral Filter [Tomasi and Manduchi, 1998] is applied to the image. This filter is used to smooth the image while preserving the edges. It helps to eliminate the possible noise information left in the evidential grid.

(d) Finally, the image is rotated using the accumulated rotation of the vehicle until time $t-1$. This step is done to align the scan image at $t-1$ with the global map.

We extract the previous location in the global map and perform the rotation to align with the time $t-1$ in order to reduce the difficulty of the network to find the correspondence with the previous predicted localization and how much it is drifting from the corrected localization. In this way, we can define a small range of possible odometry and drift values that the network can predict.

The last step in data creation is to simply concatenate the three images, creating an input size of $256 \times 256 \times 3$. Therefore, the scan information along with the global map are transformed into one three channel image that will be used as input for a sequence of convolutional layers to learn its features.

8.2.2 Data augmentation

It is important to mention that during the training phase the global image and the scan image at $t-1$ will be completely aligned, since we use the ground truth poses of the previous timestamp. For this reason, it is necessary to perform data augmentation so that the network can learn different types of possible drifts. By doing it, once we add the predicted $t-1$ pose values during the testing phase, the network will be able to predict and correct any possible drift that has been generated by the accumulation of errors. Moreover, training with only the odometry values of a dataset would limit the learning to the specific cases presented. Since we deal with inputs in the image format, it is easy to simulate new odometry values that can be learned by the network and could solve challenging cases found during the test sequences.

Considering this, there are two main types of data augmentation applied during the training:

1. Odometry augmentation: first we rotate and translate the scan image at time t and we add these values to the odometry ground truth to create an artificial new odometry. As it will be explained later, we limit the rotation and translation values to only possible values that a vehicle could have between two timestamps, considering the frequency of the dataset. Therefore, these artificial new odometry values need to be in the range of realistic values. This augmentation allows us to learn new and unsual odometry values even when we are training with a relatively small dataset.

2. Drift augmentation: we create artificially all the possible drifts we want to correct, since the dataset only provides us with the ground truth positions. To do so, we rotate and translate the global map image so that it will not be anymore completely aligned to the previous scan image, creating an artificial drift. This drift simulates an accumulation of odometry errors from previous timestamps, which generated a wrong estimation of vehicle's localization. In this way, we have different variations of drift that can occur during the trajectory of the vehicle. As we mentioned before, during the data creation the global map images have a size of 512×512 to leave enough information for the data augmentation to be possible. Therefore, after performing the drift augmentation, the images are cropped to 256×256 to have the same size as the scan images.

8.2.3 Network and Training Configuration

In Figure 8.2 the architecture of the proposed network is presented. The input is created by stacking the grid images created as described in the previous subsection, while the rest of the network can be separated in the two main parts: the sequence of convolutional layers and the linear layers. For each estimation, drift and odometry, the network predict the translation and rotation values. For the odometry we only predict one translation (longitudinal) value since we assume no sideways motion of the vehicle.

The network consists of a VGG-based [Simonyan and Zisserman, 2014] CNN architecture that learns the features between the different images to estimate the position

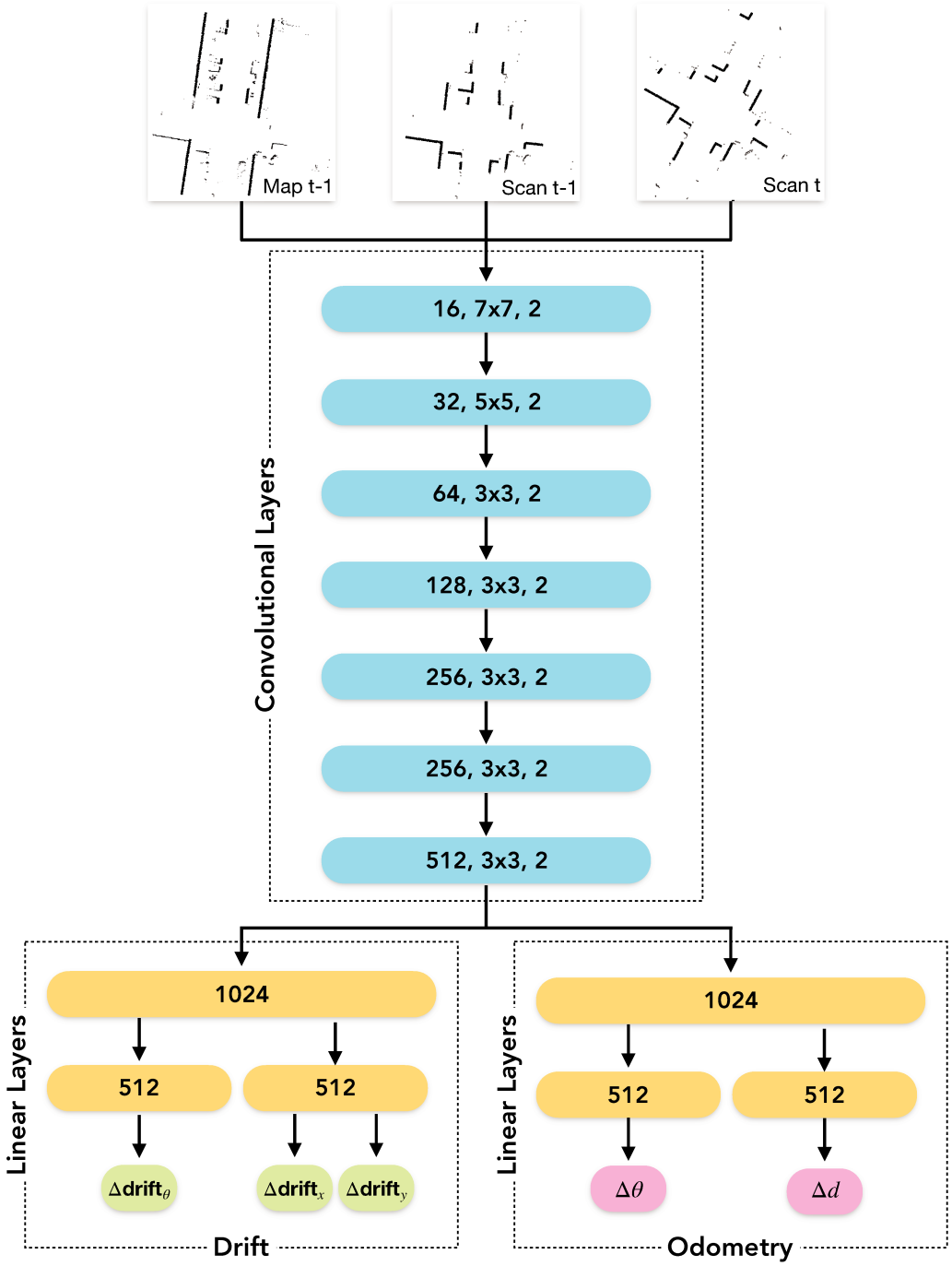


Figure 8.2: Architecture of the proposed network.

of the vehicle. The goal is that the network can learn the odometry between two timestamps analyzing the two sequential scan images, and the drift between the scans and the global map using the map image, that can correct the accumulated position of the vehicle.

In chapter 7 we showed that the rotation and translation estimation can be simplified to a classification problem instead of a regression, since we can define a relatively small range of possible values between two frames. Considering their possible range of values we create a number of classes for each output. First, for the rotation we observed that the possible values between two frames belong to the interval $\pm 5.6°$, therefore with a $0.1°$ resolution we have 112 possible classes to be estimated. While, for the odometry translation we considered a range from 0 to 2.0 meters with a resolution of 0.1 meters, therefore resulting in 20 classes. These were the configurations for the odometry estimation. At the same time, we analyzed what were the possible corrections we could have over time related to the drift without affecting the quality of localization. Considering this, for the rotation, we defined a possible drift correction in the range of $\pm 3.0°$, creating 60 classes; while for the translation in both axis, we considered a possible drift of ± 2.0 meters, resulting in 40 possible classes.

In the same way as in the previous chapter, we define the labels in the format of an Ordinal Classification, which consists in a method that enables standard classification algorithms to make use of ordering information in class attributes. We apply this method for both drift and odometry estimations, but considering their different possible range of values. The ordinal classification for this type of task is explained in more details in chapter 7.

This classification format allows the network to learn the order of the classes by showing that one value is smaller or larger than the other. After labeling each sample, the network solves each binary classification subproblem by calculating the Binary Cross Entropy between the target and the output for rotation and translation (odometry and drift) and we sum them as follow:

$$
\begin{aligned}
&\mathcal{L}_{odom} = \mathcal{L}_{BCE}(\Delta\hat{d}, \Delta d) + \ \beta\mathcal{L}_{BCE}(\Delta\hat{\theta}, \Delta\theta) \\
&\mathcal{L}_{drift} = \mathcal{L}_{BCE}(\Delta\hat{x}_d, \Delta x_d) + \ \mathcal{L}_{BCE}(\Delta\hat{y}_d, \Delta y_d) + \alpha\mathcal{L}_{BCE}(\Delta\hat{\theta}_d, \Delta\theta_d) \\
&\mathcal{L}_{total} = \mathcal{L}_{odom} + \mathcal{L}_{drift} \\
&\text{where } \mathcal{L}_{BCE}(x, y) = -\sum_k y_k \log(x_k) + (1 - y_k) \log(1 - x_k)
\end{aligned}
\tag{8.5}
$$

where Δd and $\Delta\theta$ are relative ground-truth translation and rotation rank labels for odometry, while $\Delta\hat{d}$ and $\Delta\hat{\theta}$ are their output of the network counterparts. At the same time, Δx_d, Δy_d and $\Delta\theta_d$ are the ground-truth translation and rotation rank labels for drift, and $\Delta\hat{x_d}$, $\Delta\hat{y_d}$ and $\Delta\hat{\theta_d}$ their output of the network counterparts. The output values pass by a Sigmoid function before the loss function. We use the parameters $\beta > 0$ and $\alpha > 0$ to balance the scale difference between the rotation and translation loss values.

Figure 8.3: Global map of Sequence 07.

8.3 Experimental Results

For validation we use again the KITTI dataset [Geiger et al., 2013]. To obtain a 2D laser scanner dataset, we simulate a 360° 2D pointcloud from the Velodyne data. We project all the points to 2D, remove the ground points and consider only the points of a certain height. We also perform a raytracing algorithm to remove the points that are in the same ray, keeping only the ones closer to the vehicle. In this way, we have a more realistic simulation of a 2D laser scanner.

We use 11 sequences from the KITTI odometry dataset for the proposed method. We separate 9 for training and 2 for testing. We use for training the sequences 00, 02, 03, 04, 06, 08 and 09 and for testing the sequences 05 and 07. We chose these two sequences because they are not very long, leaving more data for the training, but they can still be challenging and present the potential of the proposed method. The algorithm was tested without GPU acceleration (4,0 GHz Intel Core i7), and it can obtain a real-time performance of 30 frames per second.

In the first step, we create the maps of the environment for each sequence. The map is created using the evidential model as presented in subsection 8.2.1 and using the ground truth poses given by the dataset. Considering the type of sensor data used and the size of the environment, we chose to create the maps with a resolution of 0.2 meter. Figure 8.3 shows an example of a global map, representing only the occupied beliefs as explained in subsection 8.2.1. After the map creation, the network starts with the vehicle located in the initial position of the map with a random error of $\pm$ 3 meters. We also add random error of 0.5 meters in the creation of the map to simulate a more realistic scenario.

In Figure 8.4 and Figure 8.5 we can analyze the absolute error in translation (longitudinal and lateral) and rotation for each frame of sequences 05 and 07. These values are calculated by finding the difference between the global accumulated translation and

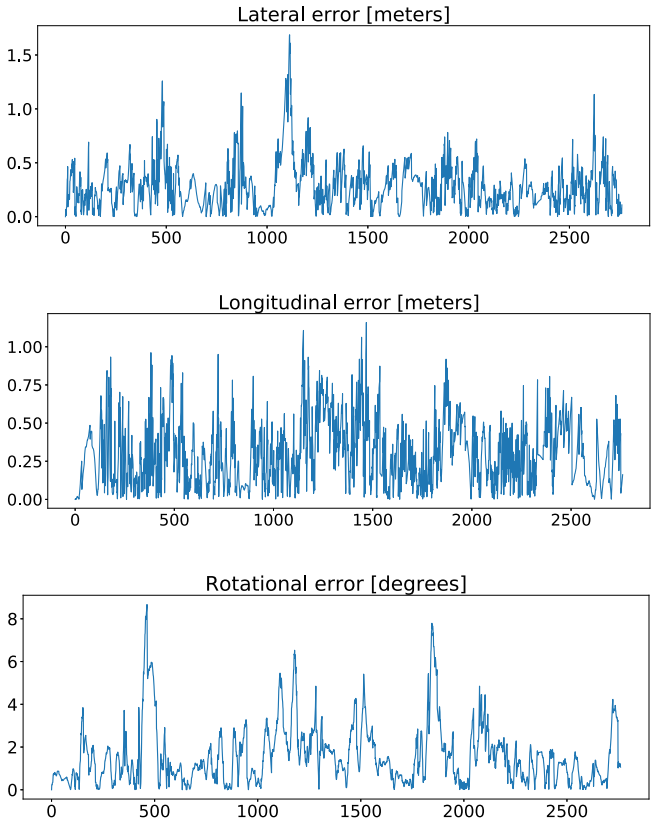


Figure 8.4: Lateral, longitudinal and rotational error for each frame of sequence 05.

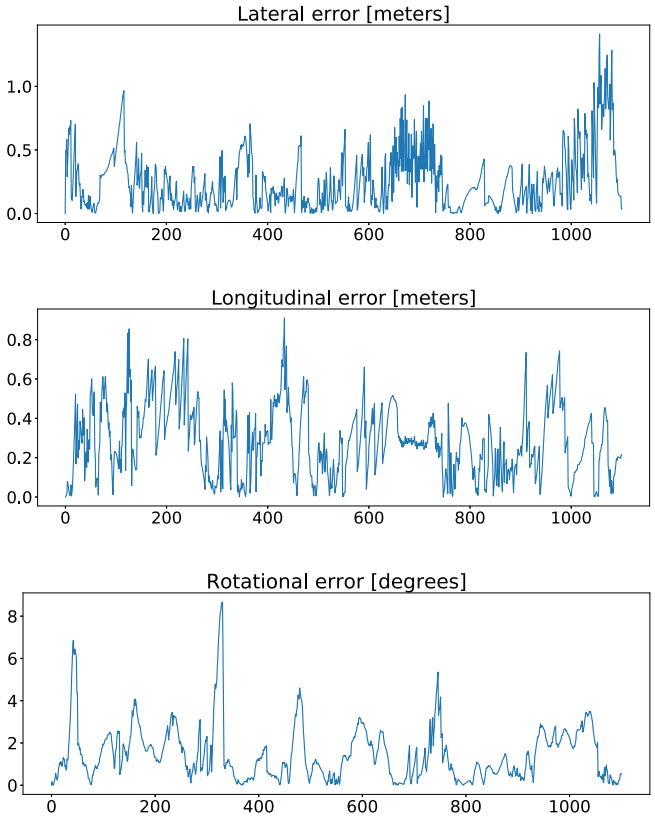


Figure 8.5: Lateral, longitudinal and rotational error for each frame of sequence 07.

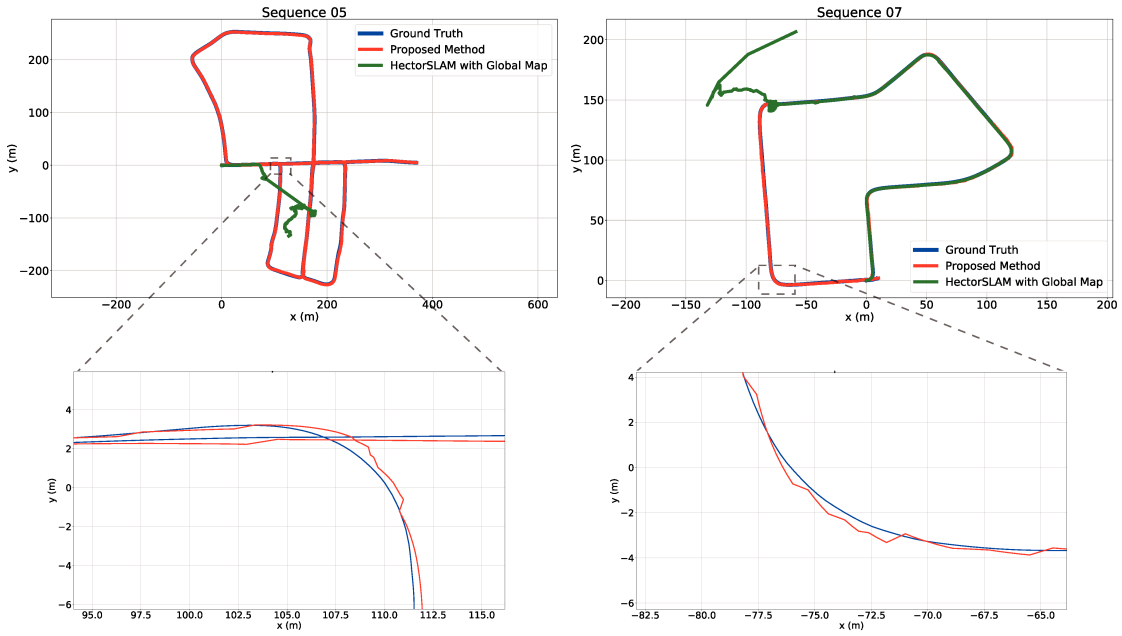


Figure 8.6: Trajectory results for sequence 05 and 07. We can observe that HectorSLAM is not able to follow the whole trajectory and it gets loss using a previouly mapped global map, while the proposed approach predicts all the global trajectory close to the ground truth. The extracts from the trajectories shows how the method is able to correct the localization once it starts to drift away from the right position.

rotation values and their ground truth counterparts. The mean absolute error was 0.25 meters for translation and 1.4 degrees for rotation. Considering that we use a resolution of 0.2 meters with only the sparse information of the simulated 2D laser scanner, the network gets a median translation error of only around 1 cell.

To the best of our knowledge, there is no method that does relocalization inside 2D maps with the KITTI dataset. However, HectorSLAM [Kohlbrecher et al., 2011] is a popular SLAM solution based on 2D grid maps that has the option, in their open source implementation, to use a previous map of the region to perform relocalization. Therefore, we tested it using the same test sequences, creating first a global map with the ground truth positions and then running the HectorSLAM method for relocalization inside the map. The first problem encountered was that the method did not work with our chosen resolution (0.2m). We tested different resolutions and the best resolution where the method could work was 0.5m. We believe this is due to the sparse data encountered in urban environments using a 2D laser scanner. However, the main problem was that even with lower resolution the relocalization gets lost in some situations, as for example when there are a lot of dynamic obstacles or very sparse scans. Most of the time, after the algorithm loses the localization it is not able to reduce the drift and it ends up going out of the global map. This can be observed in Figure 8.6 along with the resulting trajectories of our proposed approach for the two testing sequences.

In Figure 8.6 we can also analyze how the drift correction works. In the two extracts bellow the trajectories we can find moments where the accumulated error in the localization starts to increase and we can observe a big correction being made. This

correction is possible because of the output for drift correction in the network. Once the global map starts to drift from the scan image at $t-1$, the network detects that the two images are not completely aligned and find the difference in translation and rotation to estimate the drift. This is how we correct the localization in an iterative way at each new laser acquisition and we are able to overall maintain a precise global localization.

8.4 Conclusion

In this chapter, we presented a novel approach based on CNNs to perform real-time relocalization inside a map using only the data of a 2D laser scanner. The network is able to understand not only how to estimate the odometry between sequential scan information, but also to predict the drift between the current position and the global map. This drift is generated by the small errors in odometry during the trajectory and can be corrected at each frame using this approach. The proposed approach was tested using the ground truth (with noise) to create a global map of the environment, and then to perform relocalization in this map. We can assume that this method could be used in any vehicle system that can perform offline a high precision SLAM method, that can run not in real-time, and with the addition of other localization sensors. Another option would be of creating a map with a vehicle equipped with a GPS RTK (not necessarily autonomously). Once this map is created, the vehicle can use our approach to obtain a reliable and fast relocalization inside that previously mapped region. As can be observed in Figure 8.3, the map does not need to be perfect and the proposed solution can work in the presence of noise.

The results were evaluated using the KITTI odometry dataset which has long and challenging sequences, where there are other vehicles that could make the localization task difficult. Results showed that the proposed approach is robust to dynamic environments and can maintain a precise global localization throughout the entire sequence. In Figure 8.6 we can observe that the network corrects the position of the vehicle once it starts to drift from the correct localization. We can also observe from Figure 8.4 and Figure 8.5 that the vehicle position is never more than 1.5 meters from the correct localization, and it has a mean error of only 0.25 meter. Considering this, we can conclude that the network is able to provide an accurate relocalization inside simple 2D laser maps.

In the future work, we could get better results by training with a larger dataset and with a real 2D laser scanner located in a good position in the vehicle. Another interesting improvement could be the addition of different sensors, such as cameras, GPS and IMU, that could provide more information for the localization. Even the use of 3D laser scanners could be explored by this approach. If the ground is extracted from the 3D point cloud and the remaining of the points are projected to a 2D grid, the proposed network could be trained in the same configuration for this data format. Additionally, if the odometry results alone are precise enough, this approach could be tested to perform online SLAM.

Part IV

Conclusion of the Thesis

In this part we first summarize and discuss the results presented in the previous chapters. Sequentially, we introduce how the work of this thesis could be further developed with possible hints for future research.

Chapter 9

Conclusions and Perspectives

Résumé du chapitre 9

Les travaux présentés dans cette thèse ont proposé différentes solutions pour la cartographie et la localisation de véhicules autonomes équipés de capteurs à faible coût, tels que des scanners laser 2D, des caméras et des odomètres. Nous avons montré que, même avec une configuration simple de capteurs, nous sommes en mesure d'obtenir des solutions robustes pour la compréhension de l'environnement du véhicule et pour sa localisation dans cet environnement. Nous avons également détaillé comment les solutions de Deep Learning peuvent être appliquées dans ce contexte, constituant ainsi un complément intéressant aux algorithmes classiques. Dans ce chapitre de conclusion, les résultats de cette thèse sont discutés et résumés, avec des conseils et pistes pour de futures recherches dans ce domaine.

In the near future autonomous vehicles are expected to become popular and be part of our everyday life. The use of this type of technology can bring several advantages for the users and the environment, such as the reduction of car accidents, the increase of car sharing and even the augmentation of the user's productivity. For this to happen, an autonomous vehicle needs to perform the driving task at least as good as a normal driver. However, since it is a machine, we expect that normal human errors should be avoided. There are several challenges still unsolved for the current autonomous vehicles to arrive at this point. The reliable understanding of the environment and the precise localization of the vehicle are two crucial problems. This thesis contributes to these topics by proposing novel mapping and localization systems from classic model-based solutions to learning-based approaches.

The first part of our research was dedicated to classic mapping solutions using evidential grid maps, which can be a powerful tool to manage highly dynamic environments such as the one of an autonomous vehicle. First, we introduced a new way of performing sensor fusion, between a stereo camera and a 2D laser scanner, using this kind of grid maps. The fusion of these two sensors increased the environment representation because of the complementary characteristics of them. Moreover, we introduced a new map layer that was able to distinguish different states for the obstacles detected, where we could determine if they are dynamic or static. Sequentially, we proposed a new localization system using this map layer and the evidential grid maps. The differentiation between static and dynamic obstacles allowed us to create a novel method based on grid matching that gives more weight to static obstacles which are more reliable to estimate the localization of the vehicle. The method proved to be robust to challenging road scenarios using only a 2D laser scanner and an odometer.

The second part of this thesis focused on the use of Neural Networks to solve the localization problem for autonomous vehicles equipped with low cost sensors. First, we introduced a novel odometry estimation method using 2D laser scanners as only inputs. The proposed network showed impressive results considering its simplicity and the small dataset provided. However, it still suffered from problems such as the lack of drift correction and the sparsity of the 2D laser data in outdoor scenarios. For example, the method had bad results in highways where the laser scanner detected almost no obstacles for long periods. Considering this problem, we presented the first neural network that was able to perform the fusion of camera images and 2D laser scanner data for odometry estimation. We proved that the network was able to consider the information of both sensors, increasing the accuracy of both camera-only and laser-only networks. However, the lack of drift correction continued to be a problem, since small errors were accumulated overtime and after a long period we could lose completely the localization of the vehicle. Finally, we presented in the last chapter a method to address this challenge. We propose a network that takes, besides the sensor information, also the global map of the environment. By the addition of this input, the model is able to estimate not only the odometry of the vehicle at each timestamp, but also the drift correction, if there is any drift related to the global map. Results show that it is possible to maintain a precise localization of the vehicle even for large sequences in challenging urban environments.

Considering these two parts, we are able to analyze some of the advantages and disadvantages of using learning-based method compared to model-based approaches. We could observe that in the first part we had to tackle the difficulties of modelling the sensors, the environment and the movement of the vehicle. However, we could have a better understanding and definition of the actions being performed at each stage along with their results. This is not possible in an end-to-end deep learning method. On the other hand, we can observe that the odometry problem along with the sensor fusion was easily addressed using neural networks without any complicated pre-processing or modelling. At the same time, we could observe that the lack of understanding of the results makes it hard to know how reliable they are, in a way that they could not be directly used in a vehicle. Finally, we showed in the last chapter how hybrid methods could be a better way of tackling this problem. We can take the advantage of modelling the environment with maps using classical methods, while using deep learning techniques to perform fast re-localization inside these maps. We believe that hybrid methods are most likely how neural networks can be safely applied in the future of autonomous vehicles.

In general, the work presented in this thesis proposed different solutions for mapping and localization of autonomous vehicles equipped with low-cost sensors, such as 2D laser scanners, cameras and odometers. We showed that even with a simple configuration of sensors, we are able to obtain robust solutions for the understanding of the surroundings of the vehicle and for the localization inside this environment. We also introduced how Deep Learning solutions can be applied in this context and is an interesting complement to classic algorithms used in the systems of AVs.

Several perspectives of future work can be generated from what we presented. First, we presented a new layer in order to maintain a life-long grid that could classify different types of obstacles. An improvement could be the addition of a semantic value to these classifications, such as classify if the obstacle is a car, a pedestrian or a building. This classification could be done in a separate Deep Learning method that is fused to our classic evidential map algorithm. Moreover, the localization algorithm based on grid matching could take the advantage of obstacle classification to improve what information should be used during the grid matching. For example, we could consider only buildings and fixed traffic signs to perform the localization. Also, the grid matching localization could be improved with the addition of a loop closure method. There are several types approaches that could be explored for this goal, such as graph-based optimization considering different submaps.

There are also different perspectives when it comes to the Deep Learning solutions presented in this thesis. We presented a method that proved that the fusion of different sensors for localization was possible. Considering this, the use of other sensors could be explored to improve the results, such as radars, IMUs and GPS. In addition, the problem of loop closure could be also explored by the use of neural networks for localization. The use of temporal neural networks to detect previously visited locations could be an interesting way of approaching this problem. Finally, it would be interesting in future work to investigate how the uncertainty can be represented using deep learning techniques for odometry estimation.

Appendices

Appendix A

Complements on Chapter 4

A.1 Combination rules for evidential grid maps

In chapter 4 we introduced the three most popular combination rules using the evidential model: conjunctive, disjunctive and dempster combination. We generalized these operators for any frame of discernment. In this appendix, we present how these combinations work for the specific case of evidential grid maps with $2^{\Theta} = \{\emptyset, F, O, \{F, O\}\}$.

A.1.1 Conjunctive combination

The conjunctive combination is defined as:

$$\forall A \in 2^{\theta} m_1 \textcircled{\cap} m_2(A) = \sum_{B \cap C = A} m_1(B) \cdot m_2(C) \tag{A.1}$$

Considering the evidential grid map frame of discernment, the result of the combination between m_1 and m_2 is:

$$m(F) = m_1(F) \cdot m_2(F) + m_1(F) \cdot m_2(\Theta) + m_1(\Theta) \cdot m_2(F) \tag{A.2}$$

$$m(O) = m_1(O) \cdot m_2(O) + m_1(O) \cdot m_2(\Theta) + m_1(\Theta) \cdot m_2(O) \tag{A.3}$$

$$m(\Theta) = m_1(\Theta) \cdot m_2(\Theta) \tag{A.4}$$

$$\begin{aligned} m(\emptyset) = & m_1(F) \cdot m_2(O) + m_1(F) \cdot m_2(\emptyset) + \\ & m_1(O) \cdot m_2(F) + m_1(O) \cdot m_2(\emptyset) + \\ & m_1(\Theta) \cdot m_2(\emptyset) + m_1(\emptyset) \cdot m_2(O) + \\ & m_1(\emptyset) \cdot m_2(F) + m_1(\emptyset) \cdot m_2(\emptyset) + \\ & m_1(\emptyset) \cdot m_2(\Theta) \end{aligned} \tag{A.5}$$

We can analyze from the equations how the conjunctive combination rule is permissive and merges both uncertain and certain information. The belief of conflict increases

as soon as there is a mismatch in the evidential masses. At the same time, the belief of unknown only increases if both of the sources are initially unknown. This happens because in this combination we assume that the pieces of evidence are both reliable.

A.1.2 Disjunctive combination

The disjunctive combination is defined as:

$$\forall A \in 2^{\theta} m_1 \textcircled{\cup} m_2(A) = \sum_{B \cup C = A} m_1(B) \cdot m_2(C) \tag{A.6}$$

Considering the evidential grid map frame of discernment, the result of the combination between m_1 and m_2 is:

$$m(F) = m_1(F) \cdot m_2(F) + m_1(F) \cdot m_2(\emptyset) + m_1(\emptyset) \cdot m_2(F) \tag{A.7}$$

$$m(O) = m_1(O) \cdot m_2(O) + m_1(O) \cdot m_2(\emptyset) + m_1(\emptyset) \cdot m_2(O) \tag{A.8}$$

$$\begin{aligned} m(\Theta) =& m_1(F) \cdot m_2(O) + m_1(F) \cdot m_2(\Theta) + \\ & m_1(O) \cdot m_2(F) + m_1(O) \cdot m_2(\Theta) + \\ & m_1(\Theta) \cdot m_2(F) + m_1(\Theta) \cdot m_2(O) + \\ & m_1(\Theta) \cdot m_2(\emptyset) + m_1(\Theta) \cdot m_2(\Theta) + \\ & m_1(\emptyset) \cdot m_2(\Theta) \end{aligned} \tag{A.9}$$

$$m(\emptyset) = m_1(\emptyset) \cdot m_2(\emptyset) \tag{A.10}$$

In contrast to the conjunctive combination rule, in this combination we assume that only one source of information is reliable. For this reason, only the unknown belief is increased if there is a mismatch in the evidential masses. The conflict information only exists if there was a belief of conflict since the beginning in both sources.

A.1.3 Dempster combination

The dempster combination is defined as:

$$\begin{aligned} m_{1 \oplus 2}(A) &= \frac{m_{1 \textcircled{\cap} 2}(A)}{1 - m_{1 \textcircled{\cap} 2}(\emptyset)} \\ m_{1 \oplus 2}(\emptyset) &= 0 \end{aligned} \tag{A.11}$$

Considering the evidential grid map frame of discernment, the result of the combination between m_1 and m_2 is:

$$m(\emptyset) = 0.0 \tag{A.12}$$

$$m(F) = \frac{m_1(F) \cdot m_2(F) + m_1(F) \cdot m_2(\Theta) + m_1(\Theta) \cdot m_2(F)}{1 - m_{1\textcircled{\cap}2}(\emptyset)} \tag{A.13}$$

$$m(O) = \frac{m_1(O) \cdot m_2(O) + m_1(O) \cdot m_2(\Theta) + m_1(\Theta) \cdot m_2(O)}{1 - m_{1\textcircled{\cap}2}(\emptyset)} \tag{A.14}$$

$$m(\Theta) = \frac{m_1(\Theta) \cdot m_2(\Theta)}{1 - m_{1\textcircled{\cap}2}(\emptyset)} \tag{A.15}$$

In the conjunctive combination rule, we observed that the conflict belief can increase fast with the number of combined mass functions. The normalization process added in the Dempster combination has the effect of distributing the belief from the conflict to the other states. Therefore, this combination rule can be useful to eliminate conflict information from the evidential grid maps (usually caused by false alarms or dynamic obstacles).

Bibliography

E Ackerman. Quanergy announces 250 solid-state lidar for cars, robots, and more. *IEEE Spectrum*, 7, 2016a. 15

E Ackerman. Velodyne says it's got a" breakthrough" in solid state lidar design. *IEEE Spectrum*, 12, 2016b. 15

Ali Akbar Aghamohammadi, Hamid D Taghirad, Amir Hossein Tamjidi, and Ehsan Mihankhah. Feature-based laser scan matching for accurate and high speed mobile robot localization. In *EMCR*, 2007. 35

AliAkbar Aghamohammadi, Amir H Tamjidi, and Hamid D Taghirad. Slam using single laser range finder. *IFAC Proceedings Volumes*, 41(2):14657–14662, 2008. 35

Juan Andrade-Cetto, Teresa Vidal-Calleja, and Alberto Sanfeliu. Unscented transformation of vehicle states in slam. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 323–328. IEEE, 2005. 38

Qadeer Baig and Olivier Aycard. Low level data fusion of laser and monocular color camera using occupancy grid framework. In *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, pages 905–910. IEEE, 2010. 21

Tim Bailey. *Mobile robot localisation and mapping in extensive outdoor environments*. PhD thesis, Ph. D. dissertation, Univ. Sydney, Australian Ctr. Field Robotics, 2002. 39

Simon Baker and Iain Matthews. Equivalence and efficiency of image alignment algorithms. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–1090. Citeseer, 2001. 75

Th Bayes. An essay towards solving a problem in the doctrine of chances. 1763. *MD computing: computers in medical practice*, 8(3):157, 1991. 52

José M Bernardo and Adrian FM Smith. *Bayesian theory*, volume 405. John Wiley & Sons, 2009. 21

Fernando Bernuy and Javier Ruiz-del Solar. Topological semantic mapping and localization in urban road scenarios. *Journal of Intelligent & Robotic Systems*, 92(1): 19–32, 2018. 24

Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992. 34

Neal E. Boudette. Fatal tesla crash raises new questions about autopilot system, 2018. URL https://www.nytimes.com/2018/03/31/business/tesla-crash-autopilot-musk.html. 4

Guillaume Bresson, Yu Li, Cyril Joly, and Fabien Moutarde. Urban localization with street views using a convolutional neural network for end-to-end camera pose regression. 2019. 42

José A Castellanos, José Neira, and Juan D Tardós. Limits to the consistency of ekf-based slam. *IFAC Proceedings Volumes*, 37(8):716–721, 2004. 38

H Jacky Chang, CS George Lee, Y Charlie Hu, and Yung-Hsiang Lu. Multi-robot slam with topological/metric maps. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1467–1472. IEEE, 2007. 24

R Omar Chavez-Garcia. *Multiple Sensor Fusion for Detection, Classification and Tracking of Moving Objects in Driving Environments*. PhD thesis, Université de Grenoble, 2014. 20, 21, 22

Ricardo Omar Chavez-Garcia and Olivier Aycard. Multiple sensor fusion and classification for moving object detection and tracking. *IEEE Transactions on Intelligent Transportation Systems*, 17(2):525–534, 2015. 21

Jaebum Choi. Hybrid map-based slam using a velodyne laser scanner. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 3082–3087. IEEE, 2014. 24

Ronald Clark, Sen Wang, Andrew Markham, Niki Trigoni, and Hongkai Wen. Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6856–6864, 2017. 42

Joachim Clemens, Thomas Reineking, and Tobias Kluth. An evidential approach to slam, path planning, and active exploration. *International Journal of Approximate Reasoning*, 73:1–26, 2016. 51, 155

Andrew I Comport, Ezio Malis, and Patrick Rives. Accurate quadrifocal tracking for robust 3d visual odometry. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 40–45. IEEE, 2007. 36

Gabriele Costante and Thomas Alessandro Ciarfuglia. Ls-vo: Learning dense optical subspace for robust visual odometry estimation. *IEEE Robotics and Automation Letters*, 3(3):1735–1742, 2018. 42

Christophe Coué, Cédric Pradalier, Christian Laugier, Thierry Fraichard, and Pierre Bessière. Bayesian occupancy filtering for multitarget tracking: an automotive application. *The International Journal of Robotics Research*, 25(1):19–30, 2006. 24

Andrew J Davison. Real-time simultaneous localisation and mapping with a single camera. In *null*, page 1403. IEEE, 2003. 36

Andrew J Davison and David W Murray. Simultaneous localization and map-building using active vision. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):865–880, 2002. 36

Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007. 36

Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006. 40

Arthur P Dempster. A generalization of bayesian inference. *Journal of the Royal Statistical Society: Series B (Methodological)*, 30(2):205–232, 1968. 54

Jean-Emmanuel Deschaud. Imls-slam: scan-to-model matching based on 3d data. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2480–2485. IEEE, 2018. 35

Arnaud Doucet, Nando De Freitas, Kevin Murphy, and Stuart Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 176–183. Morgan Kaufmann Publishers Inc., 2000. 39

Hugh Durrant-Whyte and Thomas C Henderson. Multisensor data fusion. *Springer handbook of robotics*, pages 585–610, 2008. 21

Ahmed El-Rabbany. *Introduction to GPS: the global positioning system*. Artech house, 2002. 19

Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. 51, 52

Alberto Elfes. Occupancy grids: a probabilistic framework for robot perception and navigation. 1991. 24

Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014. 36

Olivier D Faugeras, Q-T Luong, and Stephen J Maybank. Camera self-calibration: Theory and experiments. In *European conference on computer vision*, pages 321–334. Springer, 1992. 17

Sergio Alberto RODRÍGUEZ FLÓREZ and Christoph Stiller. Contributions by vision systems to multi-sensor object localization and tracking. 2011. 17

Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014. 36

Udo Frese. A discussion of simultaneous localization and mapping. *Autonomous Robots*, 20(1):25–42, 2006. 38

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11): 1231–1237, 2013. 76, 78, 98, 111, 113, 126

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016. 92

Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010. 40, 94

Giorgio Grisettiyz, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2432–2437. IEEE, 2005. 35

Jose E Guivant and Eduardo Mario Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE transactions on robotics and automation*, 17(3):242–257, 2001. 39

Dirk Hahnel, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 1, pages 206–211. IEEE, 2003. 35

Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1106–1112. IEEE, 1997. 17

Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278. IEEE, 2016. 32

Seungpyo Hong, Heedong Ko, and Jinwook Kim. Vicp: Velocity updating iterative closest point algorithm. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1893–1898. IEEE, 2010. 35

Zhencheng Hu, Francisco Lamosa, and Keiichi Uchimura. A complete uv-disparity study for stereovision based 3d driving environment analysis. In *Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM'05)*, pages 204–211. IEEE, 2005. 57

Katarzyna Janocha and Wojciech Marian Czarnecki. On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*, 2017. 88

Maximilian Jaritz, Raoul de Charette, Marin Toromanoff, Etienne Perot, and Fawzi Nashashibi. End-to-end race driving with deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2070–2075. IEEE, 2018. 86

Simon J Julier and Jeffrey K Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Int. symp. aerospace/defense sensing, simul. and controls*, volume 3, pages 182–193. Orlando, FL, 1997. 38

Simon J Julier and Jeffrey K Uhlmann. A counter example to the theory of simultaneous localization and map building. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 4238–4243. IEEE, 2001. 38

Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Fast incremental smoothing and mapping with efficient data association. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1670–1677. IEEE, 2007. 40

Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960. 37

Juho Kannala and Sami S Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE transactions on pattern analysis and machine intelligence*, 28(8):1335–1340, 2006. 17

Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE international conference on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE, 2016. 42

Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5974–5983, 2017. 42

Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015. 42

Christian Kerl, Jurgen Sturm, and Daniel Cremers. Dense visual slam for rgb-d cameras. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2100–2106. IEEE, 2013. 36

Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007. 36

Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 155–160. IEEE, 2011. 32, 128

Marek Kurdej, Julien Moras, Veronique Cherfaoui, and Philippe Bonnifait. Map-aided evidential grids for driving scene understanding. *IEEE Intelligent Transportation Systems Magazine*, 7(1):30–41, 2015. 62

Raphael Labayrade, Didier Aubert, and J-P Tarel. Real time obstacle detection in stereovision on non flat road geometry through" v-disparity" representation. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 646–651. IEEE, 2002. 57

Richard B Langley. Rtk gps. *GPS World*, 9(9):70–76, 1998. 19

John J Leonard and Hugh F Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on robotics and Automation*, 7(3):376–382, 1991. 30

John J Leonard and Hans Jacob S Feder. A computationally efficient method for large-scale concurrent mapping and localization. In *Robotics Research*, pages 169–176. Springer, 2000. 39

Jiaxin Li, Huangying Zhan, Ben M Chen, Ian Reid, and Gim Hee Lee. Deep learning for 2d scan matching and loop closure. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 763–768. IEEE, 2017a. 43, 93, 94

Ling Li and Hsuan-Tien Lin. Ordinal regression by extended binary classification. In *Advances in neural information processing systems*, pages 865–872, 2007. 109

Ruihao Li, Qiang Liu, Jianjun Gui, Dongbing Gu, and Huosheng Hu. Indoor relocalization in challenging environments with dual-stream convolutional neural networks. *IEEE Transactions on Automation Science and Engineering*, 15(2):651–662, 2017b. 42

Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7286–7291. IEEE, 2018. 42, 86

Kai Lingemann, Andreas Nüchter, Joachim Hertzberg, and Hartmut Surmann. High-speed laser localization for mobile robots. *Robotics and autonomous systems*, 51(4): 275–296, 2005. 35

Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, 4(4):333–349, 1997. 40

Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1): 3–15, 2017. 63

Raj Madhavan and Hugh F Durrant-Whyte. Natural landmark-based autonomous vehicle navigation. *Robotics and Autonomous Systems*, 46(2):79–95, 2004. 35

Agostino Martinelli, Nicola Tomatis, and Roland Siegwart. Some results on slam and the closing the loop problem. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2917–2922. IEEE, 2005. 38

Ruben Martinez-Cantin and Jose A Castellanos. Unscented slam for large-scale outdoor environments. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3427–3432. IEEE, 2005. 38

Christopher Mei and Patrick Rives. Calibration between a central catadioptric camera and a laser range finder for robotic applications. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 532–537. IEEE, 2006. 18

Jilin Mei, Biao Gao, Donghao Xu, Wen Yao, Xijun Zhao, and Huijing Zhao. Semantic segmentation of 3d lidar data in dynamic scene using semi-supervised learning. *IEEE Transactions on Intelligent Transportation Systems*, 2019. 86

Ellon Mendes, Pierrick Koch, and Simon Lacroix. Icp-based pose-graph slam. In *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 195–200. IEEE, 2016. 32

Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949. 39

Javier Minguez, Florent Lamiraux, and Luis Montesano. Metric-based scan matching algorithms for mobile robot displacement estimation. In *ICRA*, pages 3557–3563, 2005. 35

TM Mitchell. Machine learning, mcgraw-hill higher education. *New York*, 1997. 86

Kenro Miyamoto. Fish eye lens. *JOSA*, 54(8):1060–1061, 1964. 17

Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Aaai/iaai*, pages 593–598, 2002. 32, 39

Frank Moosmann and Christoph Stiller. Velodyne slam. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 393–398. IEEE, 2011. 35

Julien Moras, Véronique Cherfaoui, and Philippe Bonnifait. Credibilist occupancy grids for vehicle perception in dynamic environments. In *2011 IEEE International Conference on Robotics and Automation*, pages 84–89. IEEE, 2011. 24, 53, 56, 67, 121

Julien Moras, Véronique Cherfaoui, and Philippe Bonnifait. Evidential grids information management in dynamic environments. In *17th International Conference on Information Fusion (FUSION)*, pages 1–7. IEEE, 2014. 53, 62

Melvin M Morrison. Inertial measurement unit, December 8 1987. US Patent 4,711,125. 18

Philippe Moutarlier and Raja Chatila. Stochastic multisensory data fusion for mobile robot location and environment modeling. In *5th Int. Symposium on Robotics Research*, volume 1. Tokyo, 1989. 37

Philippe Moutarlier and Raja Chatila. An experimental system for incremental environment modelling by an autonomous mobile robot. In *Experimental Robotics I*, pages 327–346. Springer, 1990. 37

Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31 (5):1147–1163, 2015. 36

Shree K Nayar. Omnidirectional video camera. In *Proc. DARPA Image Understanding Workshop*, volume 1, pages 235–241, 1997. 18

José Neira and Juan D Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on robotics and automation*, 17(6):890–897, 2001. 39

Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011. 36

Austin Nicolai, Ryan Skeele, Christopher Eriksen, and Geoffrey A Hollinger. Deep learning for laser based odometry estimation. In *RSS workshop Limits and Potentials of Deep Learning in Robotics*, 2016. 43, 86, 94

Feiping Nie, Zhanxuan Hu, and Xuelong Li. An investigation for loss functions widely used in machine learning. *Communications in Information and Systems*, 18(1):37–52, 2018. 88

Zhenxing Niu, Mo Zhou, Le Wang, Xinbo Gao, and Gang Hua. Ordinal regression with multiple output cnn for age estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4920–4928, 2016. 109, 111

Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018. 88

Bradford W Parkinson and Per K Enge. Differential gps. *Global Positioning System: Theory and applications.*, 2:3–50, 1996. 19

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013. 96

Naman Patel, Anna Choromanska, Prashanth Krishnamurthy, and Farshad Khorrami. Sensor modality fusion with cnns for ugv autonomous driving in indoor environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1531–1536. IEEE, 2017. 22

Mathias Perrollaz, Anne Spalanzani, and Didier Aubert. Probabilistic representation of the uncertainty of stereo-vision and application to obstacle detection. In *2010 IEEE Intelligent Vehicles Symposium*, pages 313–318. IEEE, 2010a. 58

Mathias Perrollaz, John-David Yoder, Anne Spalanzani, and Christian Laugier. Using the disparity space to compute occupancy grids from stereo-vision. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2721–2726. IEEE, 2010b. 57

Sam T Pfister, Kristo L Kriechbaum, Stergios I Roumeliotis, and Joel W Burdick. Weighted range sensor matching algorithms for mobile robot displacement estimation. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1667–1674. IEEE, 2002. 35

Oliver Pink. Visual map matching and localization using a global feature map. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–7. IEEE, 2008. 23

RH Rasshofer, M Spies, and H Spies. Influences of weather phenomena on automotive laser radar systems. *Advances in Radio Science*, 9(B. 2):49–60, 2011. 15

Thomas Reineking and Joachim Clemens. Dimensions of uncertainty in evidential grid maps. In *International Conference on Spatial Cognition*, pages 283–298. Springer, 2014. 66

Bruno Ricaud, Cyril Joly, and Arnaud de La Fortelle. Nonurban driver assistance with 2d tilting laser reconstruction. *Journal of Surveying Engineering*, 143(4):04017019, 2017. 15

Fabrizio Russo and Giovanni Ramponi. Fuzzy methods for multisensor data fusion. *Ieee transactions on instrumentation and measurement*, 43(2):288–294, 1994. 22

Glenn Shafer. *A mathematical theory of evidence*, volume 42. Princeton university press, 1976. 53, 56, 121

Geraldo Silveira, Ezio Malis, and Patrick Rives. An efficient direct approach to visual slam. *IEEE transactions on robotics*, 24(5):969–979, 2008. 36

Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *IJCAI*, volume 95, pages 1080–1087, 1995. 24

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 123

Santokh Singh. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Technical report, 2015. 4

Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990. 30, 37

Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986. 37

Hartmut Surmann, Kai Lingemann, Andreas Nüchter, and Joachim Hertzberg. A 3d laser range finder for autonomous mobile robots. In *Proceedings of the 32nd ISR (International Symposium on Robotics)*, volume 19, pages 153–158, 2001. 15

Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual slam algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):16, 2017. 35

Sebastian Thrun and John J Leonard. Simultaneous localization and mapping. In *Springer handbook of robotics*, pages 871–889. Springer, 2008. 37

Sebastian Thrun and Michael Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006. 40

Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Iccv*, volume 98, page 2, 1998. 122

Guillaume Trehard, Zayed Alsayed, Evangeline Pollard, Benazouz Bradai, and Fawzi Nashashibi. Credibilist simultaneous localization and mapping with a lidar. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2699–2706. IEEE, 2014. 78

Guillaume Trehard, Evangeline Pollard, Benazouz Bradai, and Fawzi Nashashibi. On line mapping and global positioning for autonomous driving in urban environment based on evidential slam. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 814–819. IEEE, 2015. 76, 78

Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999. 40

Michelle Valente, Cyril Joly, and Arnaud de La Fortelle. Fusing laser scanner and stereo camera in evidential grid maps. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 990–997. IEEE, 2018a. 21

Michelle Valente, Cyril Joly, and Arnaud De La Fortelle. Grid matching localization on evidential slam. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1477–1483. IEEE, 2018b. 34

Martin Velas, Michal Spanel, Michal Hradis, and Adam Herout. Cnn for imu assisted odometry estimation using velodyne lidar. In *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 71–77. IEEE, 2018. 43, 94, 96, 97, 98, 99, 159

Daisuke Wakabayashi. Self-driving uber car kills pedestrian in arizona, where robots roam, 2018. URL https://www.nytimes.com/2018/03/19/technology/uber-driverless-fatality.html. 4

Florian Walch, Caner Hazirbas, Laura Leal-Taixe, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization using lstms for structured feature correlation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 627–637, 2017. 42

Hongjian Wang, Guixia Fu, Juan Li, Zheping Yan, and Xinqian Bian. An adaptive ukf based slam method for unmanned underwater vehicle. *Mathematical Problems in Engineering*, 2013, 2013. 38

Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050. IEEE, 2017. 41, 42, 86, 94, 96, 98, 107, 111, 113, 155, 159

Pan Wei, Lucas Cagle, Tasmia Reza, John Ball, and James Gafford. Lidar and camera detection fusion in a real-time industrial multi-sensor collision avoidance system. *Electronics*, 7(6):84, 2018. 21

Huadong Wu, Mel Siegel, Rainer Stiefelhagen, and Jie Yang. Sensor fusion using dempster-shafer theory [for context-aware hci]. In *IMTC/2002. Proceedings of the 19th IEEE Instrumentation and Measurement Technology Conference (IEEE Cat. No. 00CH37276)*, volume 1, pages 7–12. IEEE, 2002. 22

Kai M Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, volume 2, 2010. 24

Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2018. 22

Ronald R Yager. Entropy and specificity in a mathematical theory of evidence. *International Journal of General System*, 9(4):249–260, 1983. 66

Chunlei Yu, Veronique Cherfaoui, and Philippe Bonnifait. Evidential occupancy grid mapping with stereo-vision. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 712–717. IEEE, 2015. 66, 67

Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, 2014. 35, 43

Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 666–673. Ieee, 1999. 17

List of Figures

1.1 Components of an autonomous robotic system along with their most common tasks. 5

2.1 An example of sensor configuration for an autonomous vehicle 15
2.2 Pinhole camera model 16
2.3 Pespective, diotropic and catadioptric cameras 17

3.1 SLAM problem 31
3.2 Architecture in a classic model-based SLAM 33
3.3 Comparison between a classic Visual SLAM algorithm and an end-to-end Deep Learning method. Image from [Wang et al., 2017]. 41

4.1 Schematics of the proposed system. 50
4.2 Comparison between the Evidential and the Bayesian approaches for occupancy grid maps. F, O, $\emptyset$ and Θ represent the free, occupied, conflict and unknown states respectively. We can observe how in the probabilistic grid map the lack of information and the conflict information end up having the same representation ($P(O) = P(F) = 0.5$), while in the evidential grid map we can analyze them separately ($m(\Theta) = 1$, $m(O) = m(F) = 0.5$ and $m(\emptyset) = 1$). Image from [Clemens et al., 2016]. 51
4.3 Top: original image from one of the cameras. Middle: ground disparity map. Bottom: obstacle disparity map with the V-disparity map on the right and the U-disparity map on bottom. 58
4.4 Geometrical configuration of the stereo camera's coordinate system. . . 59
4.5 Plot of the activation function $\tanh(\alpha_a C_{TOTAL}(c_k))$ when $\alpha_a = 0.1$. . . 60
4.6 Illustration of the multi-sensor and temporal fusion steps. 61
4.7 Diagram illustrating how the states of the life-long grid can change. 1 represents the timeout parameters and 2 the accumulator. The lines represent how the states can be updated considering the new information in the fusion grid. The dash line represent a change in state that can happen in case a cell was wrongly classified as FO. 62
4.8 Examples of spatial multi-sensor grid fusion at three different time slots. 64
4.9 Global map after performing temporal fusion. The colors represent the different states of the life-long grid: CF (green), CU (gray), CO (red), FO (blue) and U (black). 65
4.10 Average entropy (top) and specificity (bottom) for each frame 66

5.1 Schematics of the proposed system. 71
5.2 The three steps of the grid matching process for localizing the vehicle. In the pre-processing step, the local grid represents the current scan grid map, while the recent grid stores the last 3 seconds of the life-long grid and the complete grid all the information gathered in the life-long grid since the beginning. 73
5.3 Example of trajectories from the KITTI odometry dataset applying the proposed method. 77
5.4 Life-long grid of sequence 06 of the KITTI odometry database. The map shows all five states: free space currently free (green), free space currently unknown (gray), currently occupied space (red), fixed occupied space (blue) and unknown (black). 77
5.5 Translation and rotation errors for 10 sequences of KITTI odometry database applying the proposed method. 78
5.6 Translation and rotation errors for 10 sequences of KITTI odometry database without the weight parameter. 79
5.7 Trajectory and global map for sequence 07 without using the time window in the grid matching step. 79

6.1 Overview of the proposed system. The complete Recurrent Convolutional Neural Network takes a sequence of 2D laser scanner measurements as input, learns its features by a sequence of CNNs, which are used by the RNN to estimate the poses of vehicle. The output is a 2D pose of the vehicle composed by two values, one for translation and another one for rotation. 85
6.2 Illustration to compare classification and regression in supervised training. 87
6.3 An example of the different layers in a Feed-Forward neural network. . 88
6.4 Example of a network composed by convolutional, pooling and dense layers. 89
6.5 Recurrent Neural Network architecture. 90
6.6 Long-Short Term Memory architecture. 91
6.7 Data encoding for the 2D laser scanner with a 360° rotation range. At each time step the raw data from the laser scanner is separated into 0.1° bins. The average depth of all the points in a specific bin is calculated and stored into a vector. The result is a 3601 size vector that stores the depth values for each bin. 94
6.8 Architecture of the proposed RCNN. Each block of the illustration presents the size of the tensors considering that the input is two sequences of laser scanners concatenated after being encoded as presented in subsection 6.3.1. 95
6.9 Trajectories of the two test sequences (05 and 07) applying the proposed method. The blue lines represent the ground truth trajectory, while in red the predicted one. 99

6.10 Results of rotation and translation estimation for the two testing sequences. The ground truth values are presented in blue and in red the output of the network. 100

7.1 Overview of the proposed system. Consecutive laser scans and camera images are used, first in two separate networks and sequentially in a common network, in order to estimate the pose of the vehicle between consecutive frames. 105
7.2 Architecture of the proposed network. Each block of the illustration presents the size of the tensors considering that the input is the raw sensor data pre-processed as presented in subsection 7.2.1. 108
7.3 Summary of the ordinal classification labeling with an example for the angle classification. 110
7.4 Trajectories of two test sequences (07 and 10) applying the proposed CNN-Fusion. The blue lines represent the ground truth trajectory, while in red the predicted one. 112
7.5 Results of rotation and translation estimation for the two testing sequences. The ground truth values are presented in blue and in red the output of the network. 114

8.1 Overview schematics of the proposed system. At each timestamp the current laser scans and the previous one are encoded along with the accumulated global map to be fed into a neural network that will estimate the localization of the vehicle inside the global map. The pose of the previous frame is used in the data encoding to define the region of the global map based on the estimated location of the vehicle up to that moment. 119
8.2 Architecture of the proposed network. 124
8.3 Global map of Sequence 07. 126
8.4 Lateral, longitudinal and rotational error for each frame of sequence 05. 127
8.5 Lateral, longitudinal and rotational error for each frame of sequence 07. 127
8.6 Trajectory results for sequence 05 and 07. We can observe that HectorSLAM is not able to follow the whole trajectory and it gets loss using a previouly mapped global map, while the proposed approach predicts all the global trajectory close to the ground truth. The extracts from the trajectories shows how the method is able to correct the localization once it starts to drift away from the right position. 128

List of Tables

2.1 Each of the three main perception sensors has their own advantages and disadvantages. In this table we can observe that radar, LiDAR and cameras are more complementary than competitive, making their fusion necessary for autonomous vehicles. 20

4.1 Fusion of two cells using different combination rules. 55

6.1 Configuration of the convolutional layers in the proposed network. . . . 96

6.2 RMSE translation drift results for the two testing sequences along with the computation time per frame without GPU acceleration. We show the difference between the use of only CNNs and RCNN, the results for both of the RCNNs are using the pre-trained CNN-Classification network. In addition, we present the error for the different training configurations presented in subsection 6.3.3. We also compare the proposed approach with two other Deep Learning odometry estimation methods, one using as the sensor a monocamera [Wang et al., 2017] and the second using a 3D LiDAR [Velas et al., 2018]. However, the result presented for the 3D LIDAR method is for their training dataset, since they chose different sequences for testing. 98

7.1 Average rotation absolute error σ_r (degrees) and the average translation absolute error σ_t (meters) results for the CNN-Laser trained with Standard Classification and with Ordinal Classification 112

7.2 Average rotation absolute error σ_r (degrees) and the average translation absolute error σ_t (meters) results for the single sensor CNNs, CNN-Cam and CNN-Laser, and the result after CNN-Fusion. 113

7.3 Average translation (%) and rotation (degree/100m) RMSE drift on trajectory lengths of 100 to 800 meters, along with the computation time per frame, for the proposed approach and the DeepVO method. . 115

RÉSUMÉ

L'arrivée des voitures autonomes va provoquer une transformation très importante de la mobilité urbaine telle que nous la connaissons, avec un impact significatif sur notre vie quotidienne. En effet, elles proposent un nouveau système de déplacement plus efficace, plus facilement accessible et avec une meilleure sécurité routière. Pour atteindre cet objectif, les véhicules autonomes doivent effectuer en toute sécurité et de manière autonome trois tâches principales: la perception, la planification et le contrôle. La perception est une tâche particulièrement difficile en milieu urbain, car elle se doit d'être suffisamment précise pour assurer à la fois la sécurité du conducteur et celle des autres. Il est décisif d'avoir une bonne compréhension de l'environnement et de ses obstacles, ainsi qu'une localisation précise, afin que les autres tâches puissent être performantes.

L'objectif de cette thèse est d'explorer différentes techniques pour la cartographie et la localisation des voitures autonomes en milieu urbain, en partant des approches classiques jusqu'aux algorithmes d'apprentissage profond. On s'intéresse plus spécifiquement aux véhicules équipés de capteurs bon marché avec l'idée de maintenir un prix raisonnable pour les futures voitures autonomes. Dans cette optique, nous utilisons dans les méthodes proposées des capteurs comme des scanner laser 2D, des caméras et des centrales inertielles à bas coût.

Dans la première partie, nous introduisons des méthodes classiques utilisant des grilles d'occupation évidentielles. Dans un premier temps, nous présentons une nouvelle approche pour faire de la fusion entre une caméra et un scanner laser 2D pour améliorer la perception de l'environnement. De plus, nous avons ajouté une nouvelle couche dans notre grille d'occupation afin d'affecter un état à chaque objet détecté. Cet état permet de suivre l'objet et de déterminer s'il est statique ou dynamique. Ensuite, nous proposons une méthode de localisation s'appuyant sur cette nouvelle couche ainsi que sur des techniques de superposition d'images pour localiser le véhicule tout en créant une carte de l'environnement.

Dans la seconde partie, nous nous intéressons aux algorithmes d'apprentissage profond appliqués à la localisation. D'abord, nous introduisons une méthode d'apprentissage pour l'estimation d'odométrie utilisant seulement des données issues de scanners laser 2D. Cette approche démontre l'intérêt des réseaux de neurones comme un bon moyen pour analyser ce type de données, dans l'optique d'estimer le déplacement du véhicule. Ensuite, nous étendons la méthode précédente en fusionnant le laser scanner 2D avec une caméra dans un système d'apprentissage de bout-en-bout. L'ajout de cette caméra permet d'améliorer la précision de l'estimation d'odométrie et prouve qu'il est possible de faire de la fusion de capteurs avec des réseaux de neurones. Finalement, nous présentons un nouvel algorithme hybride permettant à un véhicule de se localiser dans une région déjà cartographiée. Cet algorithme s'appuie à la fois sur une grille évidentielle prenant en compte les objets dynamiques et sur la capacité des réseaux de neurones à analyser des images.

Les résultats obtenus lors de cette thèse nous ont permis de mieux comprendre les problématiques liées à l'utilisation de capteurs bon marché dans un environnement dynamique. En adaptant nos méthodes à ces capteurs et en introduisant une fusion de leur information, nous avons amélioré la perception générale de l'environnement ainsi que la localisation du véhicule. De plus, notre approche a permis d'identifier les avantages et inconvénients entre les différentes méthodes classiques et d'apprentissage. Ainsi, nous proposons une manière de combiner ces deux types d'approches dans un système hybride afin d'obtenir une localisation plus précise et plus robuste.

MOTS CLÉS

SLAM, Fusion des Capteurs, L'apprentissage en profondeur

ABSTRACT

Self-driving cars have the potential to provoke a mobility transformation that will impact our everyday lives. They offer a novel mobility system that could provide more road safety, efficiency and accessibility to the users. In order to reach this goal, the vehicles need to perform autonomously three main tasks: perception, planning and control. When it comes to urban environments, perception becomes a challenging task that needs to be reliable for the safety of the driver and the others. It is extremely important to have a good understanding of the environment and its obstacles, along with a precise localization, so that the other tasks are well performed.

This thesis explores from classical approaches to Deep Learning techniques to perform mapping and localization for autonomous vehicles in urban environments. We focus on vehicles equipped with low-cost sensors with the goal to maintain a reasonable price for the future autonomous vehicles. Considering this, we use in the proposed methods sensors such as 2D laser scanners, cameras and standard IMUs.

In the first part, we introduce model-based methods using evidential occupancy grid maps. First, we present an approach to perform sensor fusion between a stereo camera and a 2D laser scanner to improve the perception of the environment. Moreover, we add an extra layer to the grid maps to set states to the detected obstacles. This state allows to track an obstacle over time and to determine if it is static or dynamic. Sequentially, we propose a localization system that uses this new layer along with classic image registration techniques to localize the vehicle while simultaneously creating the map of the environment.

In the second part, we focus on the use of Deep Learning techniques for the localization problem. First, we introduce a learning-based algorithm to provide odometry estimation using only 2D laser scanner data. This method shows the potential of neural networks to analyse this type of data for the estimation of the vehicle's displacement. Sequentially, we extend the previous method by fusing the 2D laser scanner with a camera in an end-to-end learning system. The addition of camera images increases the accuracy of the odometry estimation and proves that we can perform sensor fusion without any sensor modelling using neural networks. Finally, we present a new hybrid algorithm to perform the localization of a vehicle inside a previous mapped region. This algorithm takes the advantages of the use of evidential maps in dynamic environments along with the ability of neural networks to process images.

The results obtained in this thesis allowed us to better understand the challenges of vehicles equipped with low-cost sensors in dynamic environments. By adapting our methods for these sensors and performing the fusion of their information, we improved the general perception of the environment along with the localization of the vehicle. Moreover, our approaches allowed a possible comparison between the advantages and disadvantages of learning-based techniques compared to model-based ones. Finally, we proposed a form of combining these two types of approaches in a hybrid system that led to a more robust solution.

KEYWORDS

SLAM, Sensor Fusion, Deep Learning