UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

FILIPE WALL MUTZ

NOVEL TECHNIQUES FOR MAPPING AND LOCALIZATION OF SELF-DRIVING CARS USING GRID MAPS

VITÓRIA

2019

FILIPE WALL MUTZ

NOVEL TECHNIQUES FOR MAPPING AND LOCALIZATION OF SELF-DRIVING CARS USING GRID MAPS

Tese apresentada ao Programa de Pós-Graduação em Informática do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Doutor em Ciência da Computação.

VITÓRIA

2019

NOVEL TECHNIQUES FOR MAPPING AND LOCALIZATION OF SELF DRIVING CARS USING GRID MAPS

Filipe Wall Mutz

Tese submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Aprovada em 02 de setembro de 2019:

Alberto Ferreira De Souza
Orientador(a)

Prof. Dr. Thiago Oliveira dos Santos
Membro Interno

Profª. Drª Claudine Santos Badue Gonçalves
Membro Interno

Profª. Drª. Karin Satie Komati
Membro Externo

Prof. Dr. Felipe Maia Galvão França
Membro Externo, participação remota

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
Vitória-ES, 02 de setembro de 2019.

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Mutz, Filipe Wall, 1990-
M992n Novel techniques for mapping and localization of self driving cars using grid maps / Filipe Wall Mutz. - 2019.
117 f. : il.

Orientador: Alberto Ferreira De Souza.
Tese (Doutorado em Informática) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Computação. 2. Robótica. 3. Inteligência artificial. 4. Mapeamento digital. I. De Souza, Alberto Ferreira. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

DEDICATÓRIA

À todos que me deram suporte nessa árdua jornada.

AGRADECIMENTOS

Aos meus pais, Elvira Wall (*in memoriam*) e Alfredo Mutz, por terem me ensinado o valor da educação e por terem me amado incondicionalmente. Considero-me abençoado por ter tido pais tão maravilhosos. À minha esposa Ana Paula Martins Guimarães Mutz por seu amor, suporte, companheirismo, e compreensão ao longo desta jornada. Ao professor doutor Raul Henriques Cardoso Lopes por me apresentar a carreira acadêmica e por me dizer que eu era capaz. Aos professores do departamento de informática da Universidade Federal do Espírito Santo por me proporcionarem uma ótima formação. Ao professor doutor Elias de Oliveira pela paciência e dedicação durante a iniciação científica. Ao professor doutor Alberto Ferreira De Souza por sua orientação e paciência ao longo do mestrado e doutorado. Você é uma das pessoas mais inteligentes que conheci e conviver com você me fez crescer como profissional e como pessoa. Ao professor doutor Thiago Oliveira-Santos por estar sempre presente e disposto a ajudar. Obrigado pela honestidade, pelas críticas construtivas, e por ser uma figura que me inspira. À professora doutora Claudine Badue pela motivação nos momentos difíceis e por mostrar a importância de buscarmos ser sempre melhores. Ao professor doutor Juergen Schmidhuber e ao doutor Paulo Rauber por seus ricos ensinamentos durante meu doutorado sanduíche no *Istituto Dalle Molle di Studi Sull'Intelligenza Artificiale* (IDSIA), na Suíça. Aos meus colegas do laboratório de computação de alto desempenho (LCAD) pelas discussões filosóficas e por todo carinho. Agradeço em especial ao doutor Lucas de Paula Veronese por ser tão grande amigo, por ser compreensivo e honesto com minhas falhas e continuar presente mesmo em face delas. Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo suporte financeiro na forma de bolsas de doutorado e financiamento via programa de doutorado sanduíche no exterior (PSDE, processo nº 88881.133206/2016-01). Por fim, agraço ao Instituto Federal do Espírito Santo (IFES; Campus Serra) pela concessão de afastamento integral remunerado para realização do doutorado (portarias nº 208/2015 e nº 147/2017).

EPÍGRAFE

“A ciência se compõe de erros que, por sua vez, são os passos até a verdade.”

Julio Verne

RESUMO

Este trabalho propõe novas técnicas para construção de mapas de grade de ambientes de grande escala, e para estimativa da localização de carros autônomos nestes mapas. A técnica de mapeamento é utilizada para criar mapas de grade de ocupação, de refletividade, coloridos e semânticos. A localização é baseada em filtros de partículas. Novos métodos para cálculo dos pesos das partículas usando informações semânticas e coloridas são apresentados. A rede neural profunda DeepLabv3+ é usada para segmentar semanticamente imagens capturadas por uma câmera frontal. A estimativa das poses do veículo para o mapeamento é modelada como um problema de Localização e Mapeamento Simultâneos (*Simultaneous Localization and Mapping* – SLAM). Valores iniciais das poses são obtidos usando o algoritmo GraphSLAM para fundir dados de odometria e GPS. Esses valores são refinados usando informações de fechamento de circuito. As poses otimizadas são utilizadas para construir mapas do ambiente. As localizações dos carros autônomos são calculadas em relação a estes mapas. As técnicas de localização e mapeamento foram avaliadas em vários ambientes complexos e de larga escala usando o automóvel robótico autônomo e inteligente (*Intelligent and Autonomous Robotic Automobile* – IARA). O impacto de usar diferentes tipos de mapas de grade na acurácia da localização assim como sua robustez a condições adversas de operação (e.g., iluminação variável, e tráfico intenso de veículos e pedestres) foram avaliadas quantitativamente. Até onde sabemos, as técnicas de mapeamento e localização, a metodologia para produção dos valores de referência para os experimentos, e a avaliação da acurácia da localização para diferentes mapas de grade são novidades.

Palavras-chaves: Robótica; Carros Autônomos; Localização; Mapas de Grade;

ABSTRACT

This work proposes novel techniques for building grid maps of large-scale environments, and for estimating the localization of self-driving cars in these maps. The mapping technique is employed for creating occupancy, reflectivity, colour, and semantic grid maps. The localization is based on particle filters. New methods for computing the particles' weights using semantic and colour information are presented. The deep neural network DeepLabv3+ is used for visual semantic segmentation of images captured by a camera. The estimation of the vehicle poses for mapping is modelled as a Simultaneous Localization and Mapping (SLAM) problem. The values of the poses are obtained by using the GraphSLAM algorithm to fuse odometry and GPS data. These values are refined using loop-closure information. The optimized poses are used for building maps of the environment. The self-driving cars localizations are computed in relation to these maps. The mapping and localization techniques were evaluated in several complex and large-scale environments using a real self-driving car – the Intelligent and Autonomous Robotic Automobile (IARA). The impact of using different types of grid maps in the localization accuracy as well as its robustness to adverse conditions of operation (e.g., variable illumination, and intense traffic of vehicles and pedestrians) were evaluated quantitatively. As far as we know, the mapping and localization techniques, the methodology for producing the localization ground truth, and the evaluation of which type of grid map leads to more accurate localization are novelties.

Keywords: Robotics; Self-driving Cars; Localization; Grid Maps;

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUCTION

The self-driving cars technology has the potential to change several aspects of the society, ranging from economy, to city organization and life style. If the technology succeeds to be adopted, jobs, businesses, and even the climate may be affected in the long term. One of the main positive impacts expected from self-driving cars is the reduction of accidents. According to the National Highway Traffic Safety Administration of the United States of America (USA), 94% of all accidents are caused by human error [1]. Several of these are due to tiredness, lack of attention, and emotional aspects of human drivers. Several of these accidents could have been prevented by the use of self-driving cars since they do not suffer from these conditions.

According to a report from Securing America's Future Energy, a non-profit organization from Washington D.C., USA, the widespread adoption of self-driving cars could lead to nearly $800 billion in annual social and economic benefits to the the country by 2050 [2]. The authors of the report argue that most of this benefit will come from reducing the toll of vehicle crashes, giving productive time back to commuters, and improving USA’s energy security by reducing dependence on oil (according to the report, autonomous vehicles are likely to accelerate the transition to advanced fuels such as electricity and natural gas).

Self-driving cars also have the potential to reduce costs of businesses that depend on the transportation of goods and people. According to the Bureau of Labor Statistics from the USA, nearly 4.2 million jobs existed in their country for driving-related tasks in 2016 (bus driver, truck driver, and delivery truck driver) [3]. The costs related to these jobs are likely to be significantly reduced after the introduction of the self-driving technology. It is expected that the transportation time also will decrease since self-driving cars do not require sleep, bathroom breaks or stops for food. Besides that, it is possible to program self-driving cars for optimizing the use of their resources and by doing so expenses due to maintenance could be alleviated.

Besides businesses, self-driving cars can also benefit people that own cars for daily activities such as going to work and taking kids to school. For these users, cars are an expensive good that is most of time idle. And more, a complex “ecosystem” is maintained in houses (e.g., garages), and cities (e.g., large avenues, parking lots, overpasses) to support the use of cars. By using self-driving cars, the costs of car sharing businesses can be reduced which could make the service cheaper for the end user. With low cost car sharing services available, consumers will not need to acquire cars. With fewer cars, houses and cities can be restructured. Home driveways and garages may shrink in size or be repurposed. Existing garages could even be converted to additional living space. Parking lots that takes up considerable space in any city, could also be converted for other uses, possibly making cities more human-friendly.

In order to advance the development of self-driving cars, the Defense Advanced Research Projects Agency (DARPA) organized three competitions in the last decade. The first, named DARPA Grand Challenge, was realized at the Mojave Desert, USA, in 2004, and required driverless cars to navigate a 142 mi long course throughout desert trails within a 10h time limit [4]. All competing cars failed within the first few miles. The DARPA Grand Challenge was repeated in 2005 and required robotic cars to navigate a 132 mi long route through flats, dry lake beds, and mountain passes, including three narrow tunnels and more than 100 sharp left and right turns [4]. This competition had 23 finalists and 4 cars completed the route within the time limit, with the Stanford University’s car, “Stanley” [5], claiming first place.

The third competition, known as the DARPA Urban Challenge [6], was held at the former George Air Force Base, California, USA, in 2007, and required self-driving cars to navigate a 60mi long route throughout a simulated urban environment together with other self-driving and human driven cars, within a 6h time limit. The cars had to obey California traffic rules. This competition had 11 finalists and 6 cars completed the route in time. The Carnegie Mellon University’s car, “Boss” [7], claimed first place.

After the DARPA challenges, research on driverless cars has accelerated in both academia and industry around the world. Although most of the university research on

self-driving cars has been conducted in the USA, Europe and Asia, some relevant investigations have been carried out in China, Brazil and other countries. The IARA self-driving car [8, 9, 10, 11] developed at the Universidade Federal do Espírito Santo (UFES), for instance, was the first Brazilian driverless car to travel autonomously 74 km on urban roads and highways.

In order to standardize the evaluation of self-driving cars, the SAE International (formerly simply SAE, or Society of Automotive Engineers) published a classification system to assess the level of autonomy of self-driving cars [12]. According to their system, the level of autonomy may range from level 0 to level 5. This classification is based on the amount of intervention and attentiveness required from the human driver. In level 0, the car is equipped with assistance systems capable of issuing warnings and momentarily intervening on the car. In this level, however, the system has no sustained control over the car. In level 5, on the other hand, the cars' autonomy system has continuous and complete control of the car and no human intervention is required and even allowed in any circumstance.

Nowadays, there are already several solutions for achieving the first levels of autonomy. Among others, we can cite the vehicles with lane keeping functionalities, or frontal collision avoidance. Most of these solutions identify elements of the traffic infra-structure such as lane marks, and traffic signs, without using and storing information about them [13, 14, 15, 16].

For achieving level 5, however, self-driving cars must be able of operating in complex and dynamic environments such as city centres, residential areas, and suburbs. Currently, the existence of proper traffic infra-structure cannot be guaranteed in all of these scenarios. Unless governments and private organizations commit to structure cities to support self-driving cars, the solutions based on the online detection of traffic elements are insufficient. An additional challenge for achieving level 5 of autonomy is dealing with the limited field-of-view of sensors. Elements necessary for a proper decision-making such as traffic signs, and the geometry of the road ahead may be located beyond the range observable by the sensors. Moreover, even if a proper infra-structure exists and their elements are in

the sensors' field-of-view, obstacles around the vehicle may occlude parts of the environment hindering their identification.

These limitations of systems based on the online identification of traffic elements can be overcome using *a priori* information about the environment, such as a map. This map could store the geometry of roads as well as annotations regarding traffic rules, and the positions of traffic signs and traffic lights. In areas where the traffic infra-structure is lacking or cannot be directly observed by sensors, the necessary information could be accessed in the map.

In order to use this information, the localization of the self-driving car in relation to the map needs to be known. One could think of Global Positioning System (GPS) as a solution for this localization problem. However, the accuracy and availability of GPS data cannot be guaranteed, especially in urban areas. Although the accuracy of GPSs can be improved by using base stations with known position such as Real-Time Kinematic GPS, availability remains an issue. Satellite signals are affected by atmospheric conditions and tall objects such as buildings and trees that block the direct reception of signals [17]. These factors may cause position "jumps" ranging from a few meters to dozens of meters as well as deviations from the correct GPS position that change slowly over time.

Most of the previous works addressed the localization problem using probabilistic approaches [18, 19, 17, 20, 21, 22, 16, 23, 24, 25]. Besides the pose (position and orientation) of the vehicle in the map, these approaches can be used for estimating additional parameters such as the current velocity of the car, the steering wheel angle, and hidden (unobserved) biases and gains. The set of parameters estimated during localization will be referred as the vehicle state. The probabilistic approaches for localization usually consist of iterating prediction and correction steps. These steps are repeated continually one after another. In the prediction step, the state estimate is updated to account for changes in the state since last localization step, e.g., due to motion. In the correction step, the likelihoods of the states are estimated by comparing the sensors' data collected online with the *a priori* map. Assuming that the vehicle is in a given state and using the map as a model of the environment, it is possible to predict the sensors' measurements. States are considered more likely if

the measurements predicted using the map are consistent with the data captured by the sensors.

In order to use the probabilistic approaches for localization, the map must contain the information necessary for predicting the sensors' observations. Different types of information can be used for localization of self-driving cars. Among others, we can cite visual landmarks [26], poles and trees [27, 28], occupancy structure [19, 20, 29, 30], reflectivity from **Li**ght **D**etection **A**nd **R**anging sensors (LiDAR) [23, 24, 25, 31, 32], heights of objects [25, 32], satellite and aerial images [31], road intersections [33, 34], and high-level road maps [34, 35].

Most of the localization techniques in the literature employ landmarks maps or metric maps. Landmarks maps can be efficiently stored and transmitted if the average number of landmarks per meter is small. On the other hand, using these maps require detecting landmarks in online data and matching them with the landmarks from the map. Both, the detection and matching problems can be challenging. Among the metric maps, the most commonly used are the grid maps. In grid maps, the environment is discretized in squared cells with fixed predefined size. These cells usually store statistics representing features of the objects contained in the cell.

In this work, we propose techniques for creating grid maps of complex large-scale environments (the term large-scale environment is not precisely defined in the literature, however it usually refers to outdoor/urban environments) and for estimating the localization of self-driving cars in these maps. This mapping technique can be used for creating occupancy, reflectivity, color, and semantic grid maps. Occupancy and reflectivity grid maps are the most commonly used types of maps for localization in urban environments. Autonomous vehicles can profit from the use of semantic grid maps since they can be compacted and used for perception and planning. In semantic grid maps, cells are classified according to the type of objects they contain. Although there is a substantial body of literature proposing techniques for creating semantic maps [36, 37, 38, 39, 40, 41, 42, 43], as far as we know the feasibility of using them for estimating the localization of self-driving cars was not evaluated before. In order to create semantic grid maps, the data from the sensors must be

segmented in different classes of objects. In this work, a pre-trained deep neural network – the DeepLabv3+ [44] – is used for visual semantic segmentation of images captured by a camera.

Vision is one of the most important human senses. However, using the colours of objects for localization can be challenging due to variations in the illumination and due to the presence of shadows. To try and minimize the negative impact of these factors, the entropy correlation coefficient (ECC) is employed for comparing online colour data with the information stored in colour grid maps. Precise range data from a LiDAR are fused with the images (colour and semantic) and used for projecting their information into the grid map.

The accuracy achieved by the proposed localization technique when using each type of grid map (occupancy, reflectivity colour, and semantic) is evaluated. Experiments are performed in several large-scale environments using a real self-driving car, the Intelligent and Autonomous Robotic Automobile (IARA; Figure 1), developed in the **L**aboratório de **C**omputação de **A**lto **D**esempenho (LCAD, http://www.lcad.inf.ufes.br/) at **U**niversidade **F**ederal do **E**spírito **S**anto (UFES; http://ufes.br/). Localization systems for self-driving cars must be robust to challenging conditions of operation such as variable illumination, intense traffic of vehicles and pedestrians, unexpected changes in the environment, and outdated maps. All of these challenging conditions and others are present in the performed experiments.

1.1. Hypothesis

The main hypothesis that guides this research is that it is possible to estimate the localization of a self-driving car in large-scale environments (even in challenging conditions of operation) with an average accuracy smaller or equals than 0.2m using data from odometer, IMU, camera, and LiDAR, some type of *priori* grid-map among occupancy, reflectivity, colour, and semantic grid maps, and an algorithm based on particle filters. In order to evaluate this hypothesis, the grid maps have to be built. Hence, we also evaluate the hypothesis that it is possible to create occupancy,

reflectivity, colour, and semantic grid maps of large-scale environments using the previously mentioned sensors.

1.2. Objective

Our main goal is to design, develop, and evaluate novel techniques for building grid maps of large-scale environments and for estimating the localization of self-driving cars in relation to these maps. Furthermore, for the sake of completeness, we describe the specific objectives of this research:

- Provide a grid-mapping technique that can be used for building occupancy, reflectivity, colour, and semantic grid-maps of large-scale environments (even in challenging conditions of operation).
- Provide techniques for accurately estimating the localization of self-driving cars in relation to grid-maps.
- Explore different types of grid maps for localization.
- Apply and evaluate the proposed techniques in real world scenarios using a self-driving car.

Figure 1. Intelligent and Autonomous Robotic Automobile (IARA) that was the first Brazilian self-driving car to travel autonomously 74 km on urban roads and highways. A video that shows IARA operating autonomously is available at https://youtu.be/iyKZV0ICysc.

1.3. Contributions

The main contributions of this work are the novel mapping and localization techniques and their evaluation in diverse large-scale environments. The mapping technique improves the method presented in a previous work from the same author [8]. The main novelties in relation to the previous work are the following:

- In the previous work, loop closures were detected using GPS which can present high level of noise. Detecting and handling incorrect GPS measurements can be challenging. Hence, in this work, an alternative approach is proposed. Loop closures are detected using an initial guess for the car poses obtained by fusing GPS and odometry data in a first step.

- In the previous work, the Generalized Iterative Closest Point algorithm (G-ICP) [45] is used for estimating the car poses in subsequent visits to loop closure regions. Although the G-ICP frequently produces good results, it eventually produces highly incorrect outcomes. These incorrect outcomes are hard to detect and they cause divergence in the pose estimation process. In this work, the proposed localization system is employed for computing the car poses in loop closure regions.

- In the previous work, the mapping technique is applied for building occupancy and reflectivity grid maps. In this work, semantic and colour grid maps are also built. To do so, data from LiDAR and camera are fused together.

The localization technique is based on particle filters [17, 20, 21, 22, 16]. Occupancy and reflectivity grid maps were employed in previous works for localization [23, 24, 25, 34, 29]. As far as we know, semantic grid maps and the ECC measure for comparing colour data have not been used for localization before. Moreover, a comparison of which type of grid map leads to more accurate self-driving car localization is lacking in the literature. This is a novel and important contribution of this work. A key challenge of this evaluation is the production of a satisfactory ground truth. We demonstrate that the same infra-structure used for handling loop closures during mapping can be successfully applied to produce the ground truth.

Besides the previously mentioned contributions, the following papers were published during the Ph.D.:

- Lucas de Paula Veronese, José Guivant, Fernando A. Auat Cheein, Thiago Oliveira-Santos, **Filipe Mutz**, Edilson de Aguiar, Claudine Badue, and Alberto F. De Souza. “A Light-Weight Yet Accurate Localization System for Autonomous Cars in Large-Scale and Complex Environments”. In Proceedings of IEEE International Conference on Intelligent Transportation Systems (ITSC). 2016.

- **Filipe Mutz**, Lucas de Paula Veronese, Thiago Oliveira-Santos, Edilson de Aguiar, Fernando A. Auat Cheein, Alberto Ferreira de Souza. “Large-scale Mapping in Complex Field Scenarios using an Autonomous Car”. Expert Systems with Applications, v. 46, pp. 439-462. Pergamon. 2016.

- Alberto Ferreira De Souza, Jacson Rodrigues Correia da Silva, **Filipe Mutz**, Claudine Badue, Thiago Oliveira-Santos. “Simulating Robotic Cars Using Time-Delay Neural Networks”. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN). 2016.

- Vinicius Cardoso, Josias Oliveira, Thomas Teixeira, Claudine Badue, **Filipe Mutz**, Thiago Oliveira-Santos, Lucas Veronese, Alberto F. De Souza. “A Model-Predictive Motion Planner for the IARA Autonomous Car”. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). 2016.

- **Filipe Mutz**, Vinicius Cardoso, Thomas Teixeira, Luan F. R. Jesus, Michael A. Golçalves, Rânik Guidolini, Josias Oliveira, Alberto F. De Souza. “Following the Leader using a Tracking System based on Pre-trained Deep Neural Networks”. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN). 2017.

- Rânik Guidolini, Alberto F. De Souza, **Filipe Mutz**, Claudine Badue. “Neural-Based Model Predictive Control for Tackling Steering Delays of Autonomous

Cars”. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN). 2017.

- Paulo Rauber, **Filipe Mutz**, Jürgen Schmidhuber. “Hindsight Policy Gradients”. In Proceedings of the Hierarchical Reinforcement Learning Workshop – Neural Information Processing Systems (NIPS). 2017.

- Rafael Correia Nascimento, Claudine Badue, Thiago Oliveira-Santos, Alberto Ferreira De Souza, **Filipe Mutz**, Fábio Daros Freitas, Christian Daros Freitas. “Avaliação de Oportunidades de Investimentos no Mercado Futuro Brasileiro na Escala de Dezenas de Segundos”. In Proceedings of the Simpósio Brasileiro de Pesquisa Operacional. 2018.

- Vagner Gon Furtado, Eduardo Max Amaro Amaral, **Filipe Mutz**, Flavio Severiano Lamas de Souza, Karin Satie Komati. “Conversão Semi-automática de Algoritmos Sequenciais de Processamento Digital de Imagens para Algoritmos Paralelos na Arquitetura CUDA”. Revista de Empreendedorismo, Inovação e Tecnologia (REIT-IMED). 2018.

- Paulo Rauber, Avinash Ummadisingu, **Filipe Mutz**, Jürgen Schmidhuber. “Hindsight Policy Gradients”. In Proceedings of the International Conference on Learning Representations (ICLR). 2018.

- Thomas Teixeira, **Filipe Mutz**, Vinicius B. Cardoso, Lucas Veronese, Claudine Badue, Thiago Oliveira-Santos, Alberto F. De Souza. “Map Memorization and Forgetting in the IARA Autonomous Car”. 2018. Arxiv preprint: https://arxiv.org/abs/1810.02355.

- Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius Brito Cardoso, Avelino Forechi, Luan Ferreira Reis Jesus, Rodrigo Ferreira Berriel, Thiago Meireles Paixão, **Filipe Mutz**, Thiago Oliveira-Santos, Alberto Ferreira De Souza. “Self-Driving Cars: A Survey”. 2019. Arxiv preprint: https://arxiv.org/abs/1901.04407.

- Lucas Amorim, **Filipe Mutz**, Claudine Badue, Alberto De Souza, Thiago Oliveira-Santos. “Simple and Effective Load Volume Estimation in Moving

Trucks using LiDARs". In Proceedings of SIBGRAPI – Conference on Graphics, Patterns and Images. 2019.

- Rupesh Srivastava, Pranav Shyam, Wojciech Jaśkowski, **Filipe Mutz**, Jürgen Schmidhuber. "Upside-Down Reinforcement Learning: Don't Predict Rewards – Just Map them to Actions". *Under review* in the Neural Information Processing Systems (NeurIPS). 2019.

1.4. Outline

After this introduction, the work is organized as follows.

- Chapter 2 reviews the literature related to Simultaneous Localization and Mapping (SLAM), large-scale mapping, and localization of self-driving cars.
- Chapter 3 describes the IARA self-driving car which is used for evaluating the proposed techniques. The architectures of hardware and software are presented in details.
- Chapter 4 introduces the methods used for pre-processing the sensors' data. Among others, the chapter presents the methods for calibrating odometry and reflectivity measurements, and for fusing data from LiDAR and camera.
- Chapter 5 presents the main contributions of our work: the novel techniques for building grid maps and for estimating the localization of a self-driving car in them. In particular, occupancy, reflectivity, colour, and semantic grid maps are described, along with the techniques for using them for localization and for building them.
- In Chapter 6, we present the experimental methodology to evaluate the proposed techniques and the results achieved. In particular, the technique to produce the ground truth to evaluate the localization is described.

- In Chapter 7, the conclusions of the work are presented along with a set of potential directions for future works.

2. RELATED WORK

This chapter discusses previous works on mapping, localization, and simultaneous localization and mapping (SLAM). In particular, we emphasize methods that can be applied for mapping large-scale environments and for estimating the localization of self-driving cars. The remaining of the chapter is organized as follows. Section 2.1 surveys seminal and current works on Simultaneous Localization and Mapping (SLAM). Section 2.2 discusses methods that used semantic information for building maps. Finally, Section 2.3 presents methods from previous works for estimating the localization of self-driving cars.

2.1. Simultaneous Localization and Mapping (SLAM)

The problem of mapping unknown environments is usually addressed using probabilistic approaches. These approaches can be classified into online and offline. Online algorithms use the Markov assumption, i.e., they estimate the most recent state of the self-driving car and the map assuming that the past (previous data and estimates) is irrelevant. Offline SLAM algorithms, on the other hand, try and solve the Full-SLAM problem which consists of estimating all of the vehicle states along a path and the map using for it the complete set of data available.

The SLAM algorithms based on Kalman filters such as the Extended Kalman Filter [46], the Unscented Kalman Filter SLAM [47], and Iterated Sigma Point Kalman Filter SLAM [48] are examples of online algorithms. These methods update the state estimate, the map and their uncertainties using a sequence of prediction and correction steps. During the prediction step, a motion model is used to expand the parameters search space by raising their covariance. In the correction step, on the other hand, a measurement model is used to enhance the probability of likely regions of the search space and to shrink the covariance around these values [21, 20].

Even being the standard solution to the SLAM problem for several years, the EKF-SLAM algorithm has drawbacks that prevent its use in outdoor and/or large-

scale environments [49, 21, 22, 16, 50, 51]. Once the poses and the map are represented in a joint state, adding new poses and new map variables (e.g., landmarks) increases the dimensionality of the state's mean and covariance matrices. This unbound growth of the state matrices makes the solution unfeasible to large-scale applications. Additionally, the simple act of visiting poses and observing landmarks leads to a computationally expensive update of the full mean and covariance matrices. Besides these drawbacks, the EKF-SLAM can also diverge due to several conditions, such as the incorrect matching of landmarks, and the presence of strong nonlinearities or discontinuities in the motion model and in the measurement model.

Montemerlo et al. introduced the FastSLAM and the FastSLAM 2.0 algorithms [50][51]. FastSLAM uses Monte Carlo Sampling techniques (Particle Filters) to factorize the SLAM posterior into a product of conditional map distributions and a distribution over robot paths. This factorization (called Rao–Blackwellization) enables the introduction of nonlinear motion models and notably speeds up the SLAM algorithm. In the FastSLAM framework, each particle represents a possible path travelled by the robot and a map.

Thrun and Montemerlo state that the main disadvantage of online SLAM algorithms is their inability to revisit sensors data, in special to handle loop closures [52]. Lu and Milios introduced offline techniques for estimating the localization of a robot when performing SLAM [53]. Their work as well as subsequent works [54, 55, 56] showed that it is possible to increase the accuracy of the localization estimates by memorizing the data until the estimation process is complete and only then building the map. Golfarelli, Maio, and Rizzi showed that the SLAM posterior probability can be modeled as a sparse graph [57]. Moreover, the optimization of this sparse graph leads to a sum of nonlinear quadratic factors. When optimizing this objective function (the sum of nonlinear quadratic factors), it is possible to obtain the most likely map and poses given the sensor data.

Thrun and Montemerlo proposed the GraphSLAM algorithm [52] which builds on top of these works. In the GraphSLAM, nodes represent model variables (the vehicle states and the map), and edges represent sensors measurements and their

covariances. In outdoor applications, the number of map variables (e.g., landmarks) tends to be large. Attempting to reduce the computational cost of the method, the authors propose to marginalize the map variables out of the posterior distribution. They do so by transforming the restrictions imposed by the map in relationships between poses. In particular, the paths of the vehicle in different visits to loop closure regions become dependent. These dependencies are modelled as edges in the graph. In order to add these edges, the displacement between the paths must be estimated along with the covariance of the estimates.

Levinson, Montemerlo, and Thrun propose to solve the displacement estimation problem using map-matching. They assume that the orientation of the vehicle is known, and estimate the displacement in the longitudinal and lateral directions [23]. Although it is a valid procedure when there are orientation measurements with almost zero error, the method would require a limiting amount of computation if the orientation has to be calculated. Levinson and Thrun, and posteriorly Mutz et al. used the Generalized Iterative Closest Point algorithm [45] for estimating the poses of a vehicle in subsequent visits to loop closure regions.

A common assumption in probabilistic approaches to SLAM is that sensors' data follow Gaussian distributions. In some cases, scaling factors and additive offsets can be used for transforming biased sensors' measurements into Gaussian distributed data. Perera et al. and posteriorly Kummerle propose the estimation of sensors' biases using a probabilistic framework [58][59]. They augmented the vehicle state by adding bias variables. In this case, the poses and biases are calculated together during the estimation process. An important feature of their approaches is the ability to change (even dramatically) the biases on-the-fly. This feature guarantees recoverability to unpredicted events. Inspired by these works, we also add parameters representing biases into the state of our system.

Given the sensors' data and the poses of the self-driving car, the mapping problem reduces to projecting the data into the map. The choice of which information is stored in the map and how it will be represented depends on the application. Metric representations usually decompose the state space into regularly spaced cells. For self-driving cars, the most common representation of the state space are the grid

maps [16]. In grid maps, the environment is discretized into squared cells with fixed predefined size (usually of the order of decimeters). An alternative metric representation is the Octree-based map proposed by Hornung et al. [60], which stores information with varied 3D resolutions. Compared to grid maps with varied 3D resolutions, OctoMaps (octree-based map) store only the space that is observed and, consequently, are more efficient in terms of memory consumption. However, OctoMaps treat updates of sensor data and estimation of obstacles' occupancy in a uniform and discrete manner. Consequently, they are slower than grid maps with uniform resolution [61]. Although OctoMaps represent a significant advantage in terms of memory consumption, intense computational complexity makes them unfeasible for applications that require real-time operations.

Another alternative metric representation is a hybrid map proposed by Droeschel et al. which stores occupancy and distance measurements with varied resolutions [62]. For this, measurements are stored in grid cells of increasing size from the center of the car. Thus, computational efficiency is gained by having a high resolution in the proximity of the sensor and a lower resolution as the distance from the sensor grows. This follows characteristics of some sensors with respect to distance and measurement density (e.g., angular resolution of laser sensors).

2.2. Mapping with Semantic Information

The creation of semantic grid maps has been an active area of research [36]. Zhao and Chen used RGB-D sensors for creating 3D semantic maps of indoor scenes [37]. In the work, semantic segmentations are computed by a support vector machine (SVM) classifier using features extracted from RGB-D data. The segmentations are then integrated using dense conditional random fields (CRF).

Vineet et al. used a stereo camera for creating 3D semantic grid maps of large-scale environments [38]. The camera path is estimated using visual odometry. The visual data are projected in the map using the depth maps computed by stereo-matching algorithms. The authors claim that algorithms based on stereo cameras produce data that are noisier than LiDARs' measurements, but they are suitable for both, large robots, and wearable glasses / headsets. McCormac et al. applied a

similar approach posteriorly [39], but using convolutional neural networks (CNNs) for semantic segmentation, and the ElasticFusion algorithm for path estimation.

Yang, Huang, and Scherer used CNNs and CRFs for semantic segmentation of sensors' data which are integrated in 3D semantic grid maps [40]. Similarly, Sengupta, Sturgess, and Torr used two CRFs to build 2D semantic grid maps of urban areas [63]. The first one is used for instantaneous visual semantic segmentation, while the second one is used for aggregating the segmentations in the map.

Neither of the previous works evaluate the feasibility of using the semantic grid maps for localization. Semantic landmarks maps have already been used for localization [64, 65]. Landmarks maps consist of a point cloud of landmarks, each of which containing a 3D position, and, potentially, additional features (e.g., the visual appearance in the landmark neighborhood). Atanasov et al. proposed augmenting landmarks with semantic labels to try and improve the localization performance [64, 65]. Using maps based on landmarks for localization requires solving the landmark matching problem which is particularly challenging in large-scale environments. Inspired by the benefits of using semantic information in landmarks maps for localization, in this work we evaluate the feasibility of using semantic grid maps for localization.

2.3. Localization

Levinson and Thrun [24] proposed a localization method that uses reflectivity grid maps for estimating the localization of self-driving cars. They used the multi-layer LiDAR Velodyne HDL-64E LIDAR in their work. An unsupervised calibration method is proposed for calibrating the Velodyne HDL-64E' laser beams so that they all respond similarly to the objects with the same brightness. A 2-dimension histogram filter [20] is employed to estimate the self-driving car position. Their method has shown a root mean squared (RMS) lateral error of 9 cm and a RMS longitudinal error of 12 cm.

Veronese et al. [31] proposed a localization method based on particle filters that compares satellite aerial maps with re-emission maps. Aerial maps are downloaded offline from sources in the Internet, like OpenStreetMap, and remission maps are built online from LIDAR reflectance intensity data. A particle filter algorithm is employed to estimate car's pose by matching remission maps to aerial maps using the normalized mutual information (NMI) measure to calculate the particles likelihood. The method was evaluated on a 6.5 km dataset collected by the IARA self-driving car and achieved position estimation accuracy of 89 cm. One advantage of this method is that it does not require building a map specifically for the method.

Hata and Wolf [66] proposed a localization method based on road feature detection. Their curb detection algorithm uses ring compression analysis and least trimmed squares [66] to analyze the distance between consecutive concentric measurements (or rings) formed by a multilayer LiDAR (Velodyne HDL-32E) scan. The road marking detection algorithm uses Otsu thresholding [67] to analyze LiDAR reflectance intensity data. Curb and road marking features are stored in a grid map. A particle filter algorithm is employed to estimate the car pose by matching road features extracted from multilayer LIDAR measurements to the grid map. The method was evaluated on the autonomous vehicle "CARINA" [68], and has shown lateral and longitudinal localization estimation errors of less than 30cm.

Veronese et al. [29] proposed a localization method based on particle filters that computes the particles' likelihood by matching 2D online occupancy grid-maps and 2D offline occupancy grid-maps. Two map-matching distance functions were evaluated: an improved version of the traditional Likelihood Field distance between two grid-maps, and an adapted standard Cosine distance between two high-dimensional vectors. An experimental evaluation on the IARA self-driving car demonstrated that the localization method is able to operate at about 100 Hz using the Cosine distance function, with lateral and longitudinal errors of 13cm and 26cm, respectively.

Wolcott and Eustice [25] proposed a probabilistic localization method that models the world as a multiresolution grid map. Each cell of the map stores the parameters of mixtures of Gaussians representing the height and reflectivity of the

environment. An extended Kalman filter (EKF) localization algorithm is used to estimate the car's pose by registering 3D point clouds against the multiresolution-maps. The method was evaluated on two driverless cars in adverse weather conditions and presented localization estimation errors of about 15cm.

3. THE IARA SELF-DRIVING CAR

This chapter presents the IARA self-driving car. The Intelligent and Autonomous Robotic Automobile (IARA, Figure 1) is a research autonomous vehicle developed at the Laboratório de Computação de Alto Desempenho (LCAD) of the Universidade Federal do Espírito Santo (UFES). IARA was the first Brazilian driverless car to travel autonomously 74 km on urban roads and highways. The novel techniques presented in this thesis are integrated in IARA's architecture of software, and evaluated in the real world using the self-driving car. The remaining of the chapter is organized as follows. The architectures of hardware and software of IARA are described in Sections 3.1 and 3.2, respectively. IARA's software architecture is organized in perception and decision-making modules which are detailed in Sections 3.2.1 and 3.2.2. Finally, Section 3.3 introduces IARA's motion model.

3.1. Hardware

IARA is based on a 2011 Ford Escape Hybrid, which was adapted for autonomous driving by Torc Robotics (https://torc.ai/) and by LCAD's team. To power IARA's computers and sensors, we take advantage of the 330 Volts battery used by the electric powertrain of the Ford Escape Hybrid. For running its autonomy software pipeline, IARA is equipped a workstation Dell Precision R5500, with 2 Xeon X5690 six-core 3.4GHz processors and one NVIDIA GeForce GTX-1030.

IARA's sensors comprise wheel's odometers, two multi-layer LiDARs (a Velodyne HDL-32E and a SICK LD-MRS), three stereo cameras (two PointGrey Bumblebee XB3, and a StereoLabs ZED), an IMU (Xsens MTi), and a dual RTK GPS (based on the Trimble BD982 receiver). The sensors used in this work are the Velodyne HDL-32E LiDAR, the front-facing Bumblebee XB3 camera, the Xsens MTi IMU, the Trimble RTK GPS, and the wheels' odometers. For conciseness, these sensors will be referred just by Velodyne, camera, IMU, GPS, and odometer from now on. The arrangement of the sensors in IARA as well as their coordinate systems are illustrated in Figure 2. The coordinate system of IARA, located in the centre of the

rear axle is also depicted in this figure. In all coordinate systems except the camera's one, the x-axis points forward, the y-axis points to the left, and the z-axis upwards. The camera coordinate system is rotated in relation to the others, with the x-axis pointing to the right, y-axis downwards, and z-axis pointing forward. Some of IARA's modules require knowing the relative poses among the coordinate systems. These relative poses were measured manually.

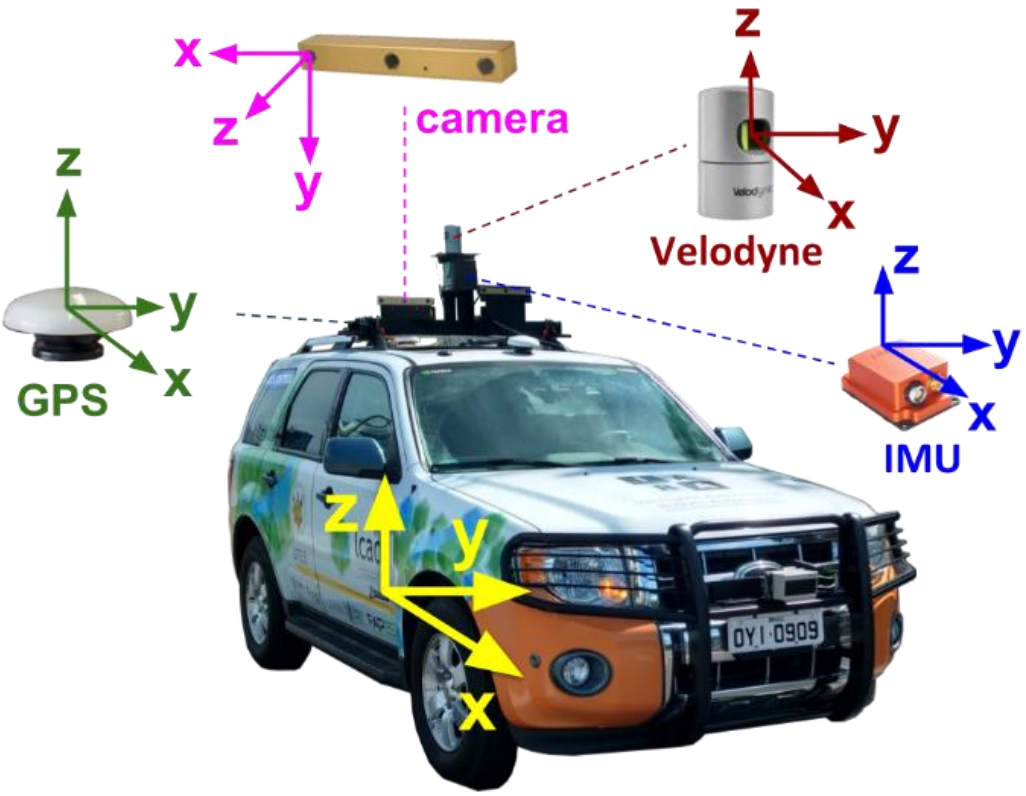


Figure 2. Sensors arrangement and their coordinate systems in the IARA self-driving car.

The data produced by the sensors is as follows. The GPS measures the position of the sensor in the world coordinate system. These measurements have variable, but bounded, error. If the car velocity is higher than a predefined threshold, the car orientation is estimated using consecutive GPS positions. Else, only the positioning data are considered. The odometer measures the car's linear velocity and the steering wheel angle. The IMU provides acceleration and angular velocity

measurements in the directions of the x, y, and z axes, as well as an orientation measurement in the world coordinate system obtained using a magnetometer. The stereo camera system captures simultaneously two images using a pair of cameras. These cameras are mounted side-by-side horizontally with an offset between them. In this work, only the right image of the pair is used. The Velodyne is composed of 32 lasers assembled one above the other. Each of the lasers measures the distance to the nearest object and the reflectivity of its surface. The set of 32 range and reflectivity readings are assumed to be obtained at the same time and it is referred as a shot. A rotating motor turns the 32 lasers around the vertical axis. For each shot, the sensor also returns the horizontal angle of the rotating motor. By turning around the vertical axis, the sensor is capable of providing a 3D view of the environment around the car. Figure 3 presents examples of Velodyne point clouds captured in different environments.

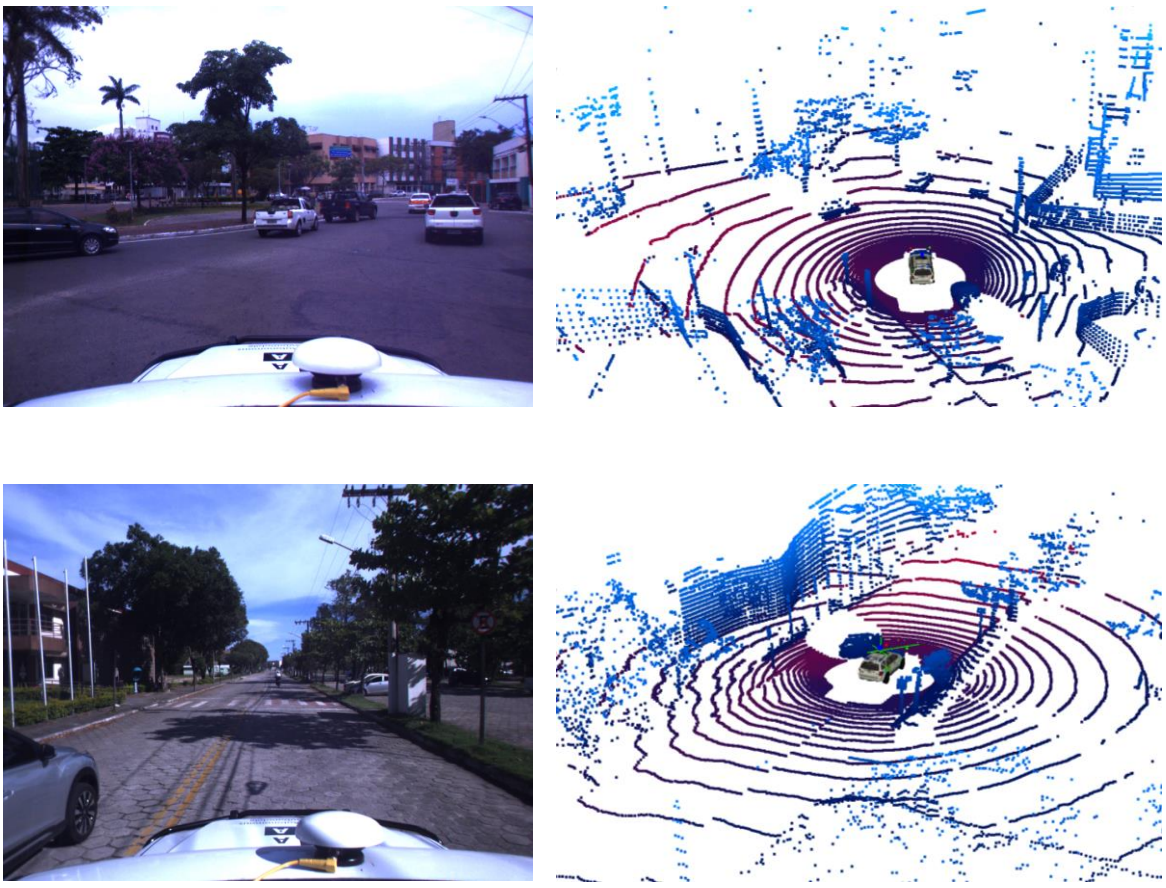


Figure 3. Examples of camera images and point clouds captured by Velodyne in different environments.

3.2. Software

IARA's software architecture is organized into perception and decision-making systems. The perception system is responsible for estimating the state of the car and for creating a representation of the environment internal to the self-driving car. This representation contains among other things, the approximate positions of static obstacles in the environment, the states of moving vehicles and pedestrians, and the rules of the road. The decision-making system is responsible for planning and executing commands to drive the car from its initial position to the final goal defined by the user. The commands are chosen considering the car state, the internal representation of the environment, traffic rules and the comfort of the passengers.

Figure 4 shows a block diagram of the main subsystems of IARA's software architecture. The figure depicts how each subsystem connects with the others. The perception system is organized in modules for mapping, localization, moving object tracking, and traffic sign detection. The decision-making system is subdivided into path planning, behaviour selection, trajectory planning, obstacle avoidance, and control subsystems.

3.2.1. Perception

The mapper module [69, 29] has two modes of operation, the offline mode and the online mode. The offline mode is employed before autonomous operation for building maps of previously unseen environments. The maps created in this mode are called offline maps and they are intended to represent the static structure of the environment as realistically as possible. During autonomous operation, the mapper module works in online mode. In this mode, instantaneous data are used for creating online maps that represent the part of the environment observed by sensors, including dynamic obstacles. The online maps are merged with the offline maps for producing merged maps which integrates the instantaneous and a priori information about the environment. The merged maps contain information regarding both static and dynamic objects. Examples of instantaneous, offline, and merged occupancy grid maps are presented in Figure 5 (a), Figure 5 (b), and Figure 5 (c), respectively.

The localizer module [29] is responsible for estimating the car's current state. The state contains the car's pose (position and orientation) in relation to the offline map. It may also contain additional information, such as, linear velocities, angular velocities, and/or calibration parameters. The localizer module estimates the car state using a probabilistic approach.

For detecting and tracking moving objects, the moving objects tracking module, or MOT, is employed. Horizontal (lane markings) and vertical (i.e. speed limits and traffic lights) traffic signalization must be recognized and obeyed by self-driving cars. The traffic signalization detection module, or TSD, is responsible for the detection and recognition of traffic signalization.

3.2.2. Decision-making

Given the information obtained by the perception subsystem, the decision-making subsystem plan and execute commands for driving the self-driving car in direction to a user-defined final goal. The route planner module computes a route, W, in the offline maps, from the current state to the final goal. A route is a sequence of way points, i.e. $W = \{w_1, w_2, \dots, w_{|W|}\}$, where each way point, w_i, corresponds to a position, i.e. $w_i = (x_i, y_i)$, in the offline maps.

Given a route, the path planner module computes an odd set of paths, $\mathrm{P} = \{P_1, P_2, \dots, P_c, \dots, P_{|\mathrm{P}|}\}$ taking in consideration the car state and the internal representation of the environment as well as traffic rules. A path is a sequence of poses, i.e., $P_j = \{p_1, p_2, \dots, p_{|P|}\}$**.** The central path, P_c, is aligned as best as possible with W. The paths at its left side ($\mathrm{P}^l = \{P_1, P_2, \dots, P_{c-1}\}$) and at its right side ($\mathrm{P}^r = \{P_{c+1}, \dots, P_{|\mathrm{P}|}\}$) have the same initial pose of P_c, but they depart from P_c to the left and to the right with different levels of aggressiveness.

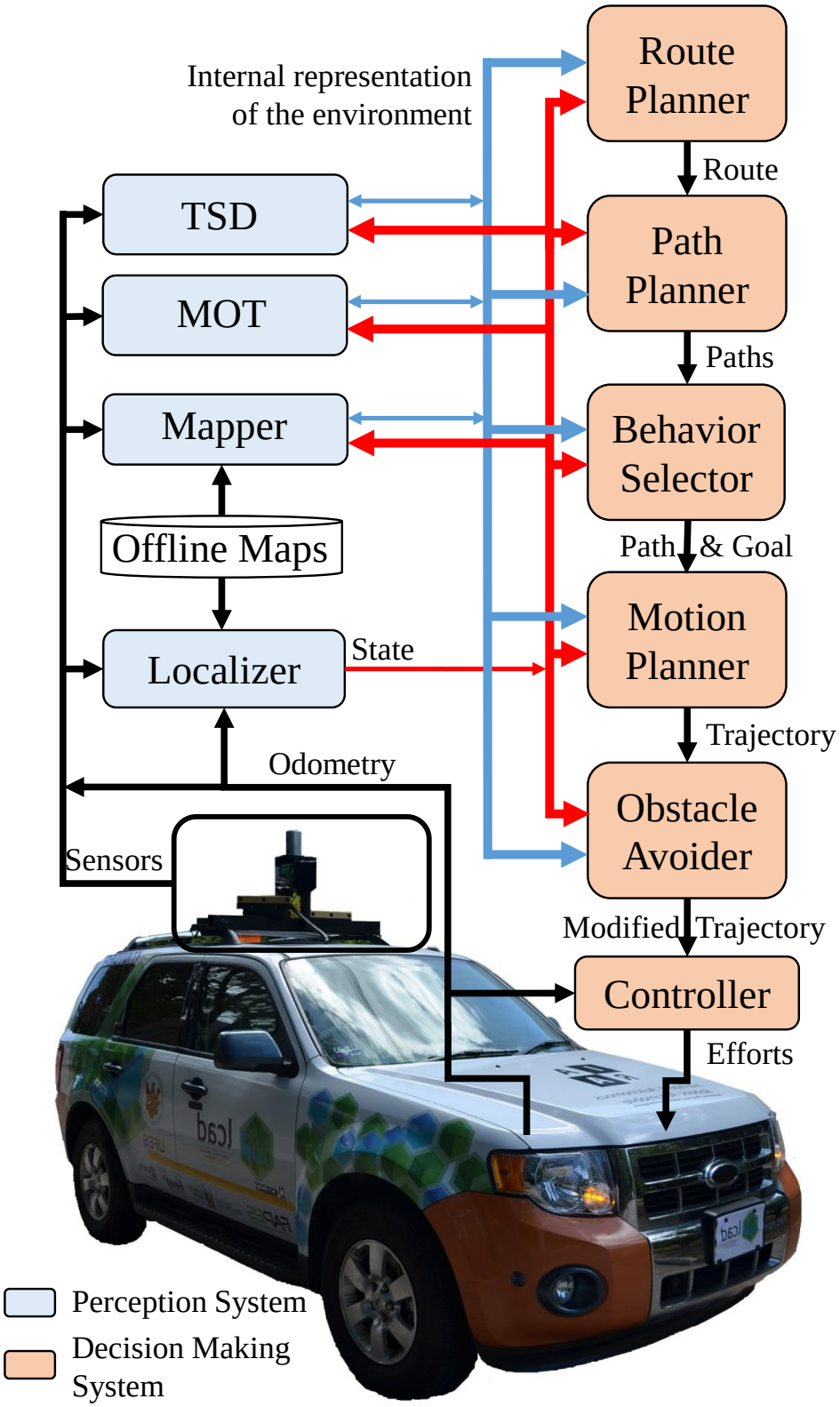
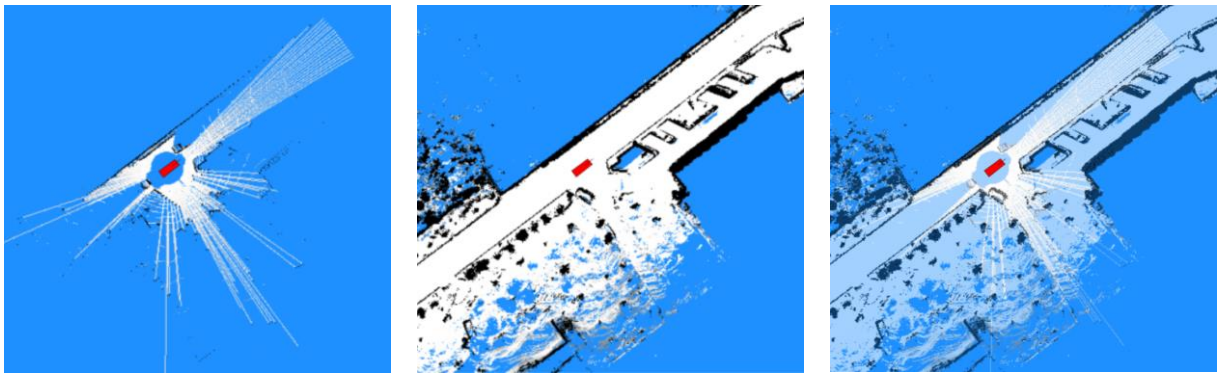


Figure 4. Overview of the software architecture of the IARA self-driving car. TSD denotes traffic signalization detection and MOT, moving objects tracking.

IARA's path planner employ the same approach used in the 2005 DARPA Grand Challenge. In this approach, instead of using an algorithm (e.g., A* search) for computing a path online, we assume that a set of paths obtained *a priori* is given as input for the module. These paths are represented by a road definition data file (RDDF). The RDDF is composed of a sequence of car poses, which were stored while IARA was conducted by a driver along a path of interest. The path planner extracts a path from a subset of the RDDF. This path is composed by a sequence of equally spaced poses starting from the car's current pose to a goal pose some meters ahead.

The behaviour selector module is responsible for choosing the current driving behaviour, such as lane keeping, overtaking, and traffic light handling. It does so by selecting a path, P_j in P, a pose in P_j a few seconds (about five seconds, usually) ahead of the current state – the decision horizon – and the desired velocity at this pose. The composition of a pose in P_j and its desired velocity results in a goal. The behaviour selector chooses a goal in the path considering the current driving behaviour, and avoiding collisions with static and moving obstacles within the time frame defined by the decision horizon. It receives as input the online map, IARA's current state, the path, and a set of annotations, such as the positions of speed bumps, security barriers, crosswalks and speed limits, and the positions and states of traffic lights. Then, it defines a goal state in the path some seconds ahead of IARA's current state, and adjusts the goal state's pose and velocity, in order to make the car behave properly according to the scenario, such as stopping on red traffic lights, reducing the velocity to avoid collisions, or stopping in busy crosswalks.

The motion planner module [9] is responsible for computing a trajectory, T, from IARA's current state to the goal state chosen by the behaviour selector. The trajectory is composed by a sequence of control commands, each one comprised of linear velocity, steering wheel angle and execution time. Besides the car state and the goal state, the module also takes in consideration the path, and the online map for computing the trajectory. It employs a model-predictive approach that (i) follows the path given by the behaviour selector, (ii) satisfies the car's kinematic and dynamic constraints, (iii) avoids obstacles in the map, and (iv) provides comfort for the passengers.



(a) **(b)** **(c)**

Figure 5. Examples of maps created by the mapper module. (a) Online map built using instantaneous data observed by the sensors during autonomous operation. (b) Offline map built before autonomous operation. (c) Merged map in which instantaneous data are fused with a priori data.

The obstacle avoider module [10] receives as input the trajectory computed by the motion planner and updates it, if necessary, to avoid collisions. The updates typically consist of reducing the velocity of the car. The module receives as input the online map, IARA's current state, and the trajectory. Then, it simulates the trajectory execution along the online map. If the trajectory intercepts an obstacle, the module decreases the linear velocity of the commands to prevent the collision.

Finally, the controller module [11] receives as input the motion planner trajectory, eventually updated by the obstacle avoider, and computes efforts commands for the steering wheel, throttle and brakes actuators. These efforts are sent directly to the car, and they are chosen so that the car executes the modified trajectory as best as the physical world allows. In IARA, a Proportional Integral Derivative (PID) approach is employed. Using the trajectory and odometry data, the controller module estimates an error measurement that accounts for how far IARA's current steering wheel angle and velocity are from those specified in the trajectory. Actuation commands are, then, computed to try and minimize this error.

3.3. Motion Model

In this Section, we describe IARA's motion model. The motion model provides a way of estimating the car movement in an interval of time given the car's linear velocity and steering wheel angle. We assume that the car moves in the ground plane ($z = 0$), and that the pitch and roll angles are perfectly measured by the IMU. These values (z, pitch, roll) are not updated by the motion model.

Given the previous assumptions, the car pose can be represented by a triplet (x, y, θ), in which the first two coordinates represent the car position in the ground plane, and the third coordinate represents its heading direction. Assume that in instant t, the car pose is (x_t, y_t, θ_t), and, after that, it moves with linear velocity v and steering wheel angle φ for Δ_t seconds. According to the Ackermann motion model [70], the car pose after the movement $(x_{t+1}, y_{t+1}, \theta_{t+1})$ is given by:

$$x_{t+1} = x_t + v \cos\theta_t \ \Delta_t \quad (1)$$

$$y_{t+1} = y_t + v \sin\theta_t \ \Delta_t \quad (2)$$

$$\theta_{t+1} = \theta_t + \frac{v \tan(\varphi)}{L} \Delta_t \quad (3)$$

where L is the distance between the front and rear axles.

The car path can be estimated using the measurements provided by the odometer and the motion model. The path obtained by this procedure is called the cars' dead-reckoning [17]. Note that the instantaneous noise in the odometry measurements impact all future poses of the path. This fact implies that the error in the poses accumulates over time causing a divergence in relation to the true path of the vehicle.

4. Sensor Data Pre-Processing

This chapter describes the steps for pre-processing the sensors' data. In Section 4.1, the methodology for synchronizing data from different sensors is presented. Then, in Sections 4.2 and 4.3, respectively, the techniques for calibrating odometry measurements and the reflectivity of Velodyne are described. Section 4.4 briefly introduces the deep neural network used for visual semantic segmentation, and the steps for pre-processing images before using them as input for the neural network. Finally, in Section 4.5, the method for fusing images and point clouds from Velodyne is presented.

4.1. Sensors Synchronization

Sensors capture data samples simultaneously and at different rates. For each data sample, the sensors provide a timestamp representing the instant in which the data was captured. The timestamp of the less frequent sensor is chosen as the reference for synchronization. A synchronization step is performed whenever a sample from the less frequent sensor is received. Since the remaining sensors capture data with higher frequency, we can assume that at least one sample from each sensor will be available between two synchronization steps.

A queue is maintained for each sensor, except for the less frequent one. In these queues, we store all samples received since the last synchronization step along with their timestamps. When a sample from the less frequent sensor is received, we search in the queues for the most synchronized samples. One sample is returned from each sensor queue. The returned sample is the one in which the difference between the sample timestamp and the reference timestamp is the smallest. The samples returned from the queues and the sample from the less frequent sensor are grouped together in a data package.

In this work, a log is defined as the set of data packages recorded while driving the self-driving car throughout an environment. For building a map, the mapping

module receives as input the whole log at once. The localization module, on the other hand, estimates the current car state using only the most recent data package. The camera and the Velodyne are the less frequent sensors, respectively. The camera is not necessary for mapping and localizing using occupancy grid maps or reflectivity grid maps. Thus, in this case, the Velodyne is the reference sensor for synchronization. When using the colour and semantic grid maps, however, the sensors' data are synchronized using the camera's timestamps.

Figure 6 presents a visualization of the method for synchronizing sensors' data assuming that the camera is the slowest sensor. The number of rows in the queues represents the frequency of the sensors. Queues with more rows represent more frequent sensors. The data packages are built by searching in each queue for the message with the nearest timestamp in relation to the timestamp of the slowest sensor (the camera in the example).

4.2. Odometry Calibration

During initial experiments, we observed that the linear velocity and the steering wheel angle values measured by the car odometer, as well as the GPS timestamps are biased. The biases in the GPS timestamps are likely to be caused by latency in the process of acquiring data. An odometry calibration subsystem was developed to try and correct the biases in the odometry measurements and in the timestamps of GPS. The subsystem is an improvement of the method presented by Mutz et al. [8]. It is based on the idea that if the odometry measurements are correct, then the dead-reckoning [71] path estimated using these measurements will be consistent with the path measured by the GPS. The calibration parameters are estimated using an optimization approach in which we search for the parameters values that minimize the distance between the GPS path and the dead-reckoning path.

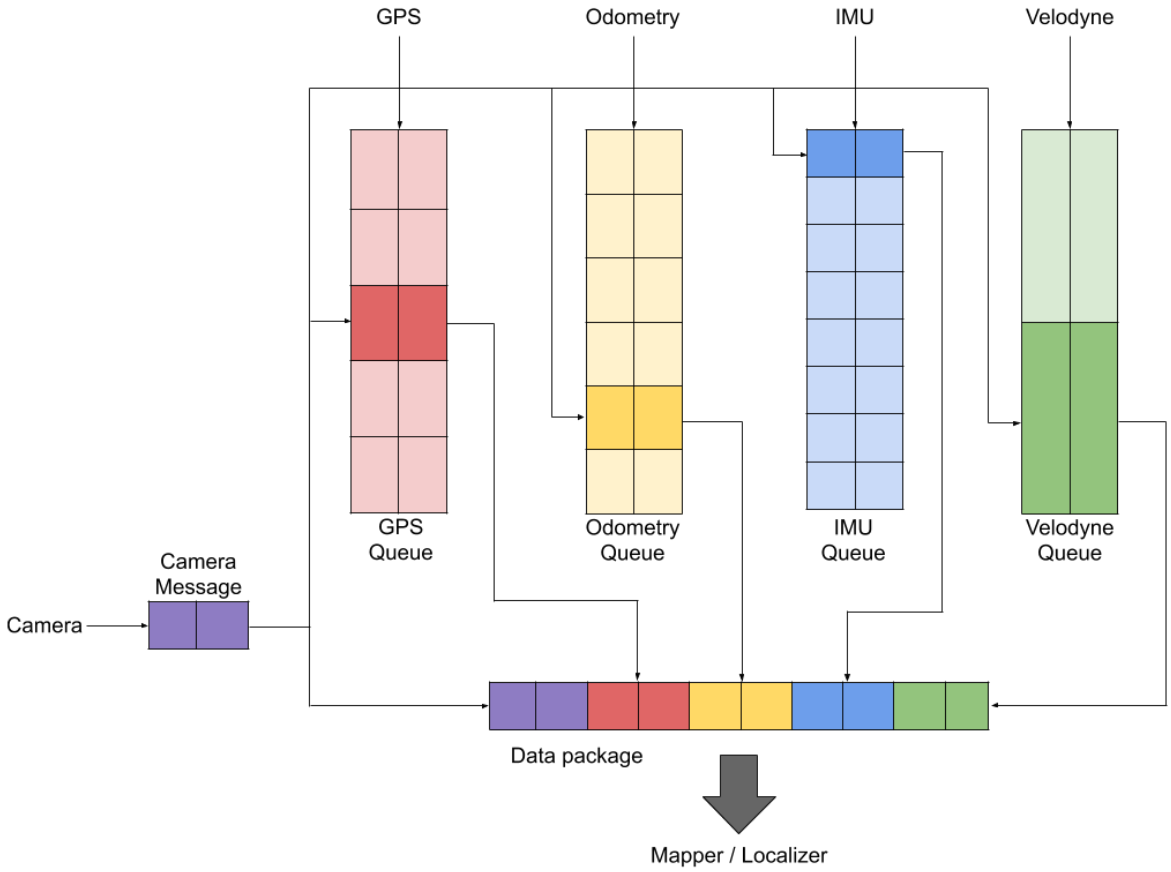


Figure 6. Visualization of the method for synchronizing sensors' data assuming that the camera is the slowest sensor. The number of rows in the queue represents the frequency of the sensors. Queues with more rows represent more frequent sensors. The columns in the queues represent the data sample and the timestamp associated with the sample.

A block diagram of the subsystem is presented in Figure 7. In the figure, blue boxes represent the inputs and outputs of the subsystem, yellow boxes represent the calibration parameters that we want to estimate, and the white circles and boxes represent transformations in the data. The subsystem receives as input odometry and GPS measurements from a log along with their timestamps. It outputs the calibration parameters, the corrected odometry measurements, and an error value that is related to the quality of the calibration.

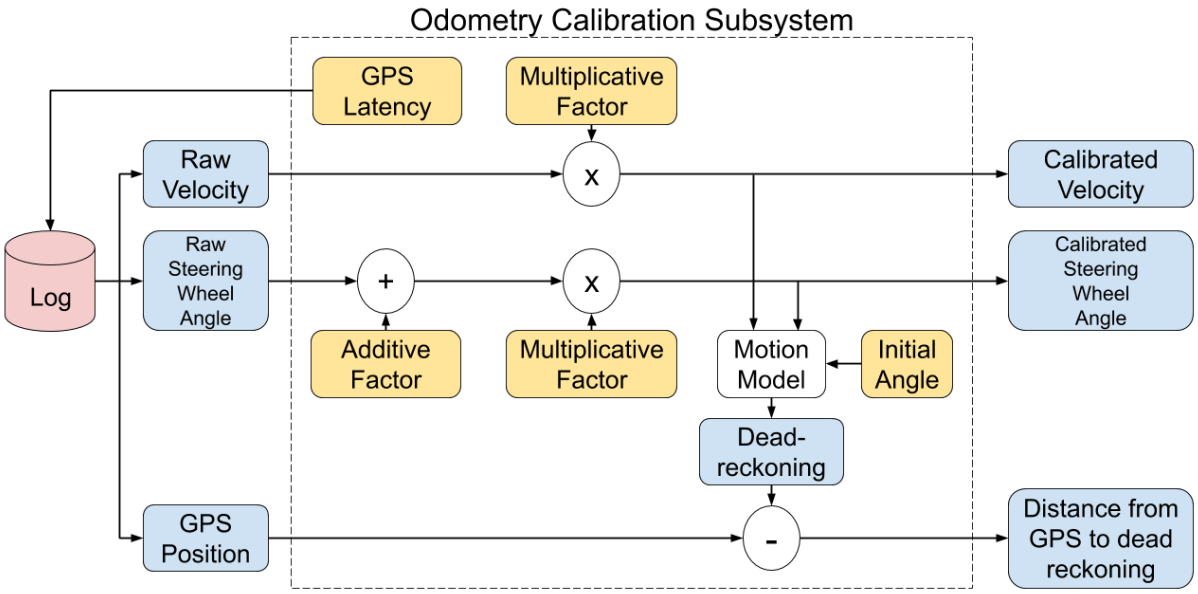


Figure 7. Block diagram of the odometry calibration process. Blue boxes represent the inputs and outputs. Yellow boxes represent parameters that are optimized to minimize the distance between dead-reckoning and GPS positions. White elements represent transformations of the data.

Before presenting a formal description of the subsystem, it is worth making a few definitions. Let:

- $g_t = \left(g_t^x, g_t^y, g_t^\theta\right)$ be the gps position associated with the timestamp t + L_{gps}, where L_{gps} is the bias in the GPS' timestamps. An offset is subtracted from the gps positions to avoid numerical issues. The offset is chosen so that origin of the world coordinate system is translated to g_0. During synchronization, GPS measurements may be assigned to different data packages depending on the value of L_{gps};

- v_t and φ_t be the raw linear velocity and steering wheel angle measured by the odometer at timestamp t;

- $\overline{\varphi_t} = m_\varphi \left(\varphi_t + a_\varphi\right)$ be the corrected steering wheel angle, where a_φ and m_φ are additive and multiplicative correction factors, respectively.

- $\overline{v_t} = v_t\, m_v$ be the corrected linear velocity, where m_v is a multiplicative correction factor. The velocity component has no additive bias because the

odometer provides a precise zero measurement. An additive factor could lead to an illusion of motion when the vehicle is not moving;

- $o_t = \left(o_t^x, o_t^y, o_t^\theta\right)$ be the car pose at instant *t* estimated by dead-reckoning. This pose is given by $o_t = o_{t-d_t} \oplus M(\overline{v_t}, \overline{\varphi_t}, d_t)$, where d_t is the time elapsed since the previous data package, *M* represents the motion model introduced in Section 3.3, and $\oplus$ is the operation of composition of poses [72]. The value of $M(\overline{v_t}, \overline{\varphi_t}, d_t)$ is an estimate of the car movement in d_t seconds, assuming linear velocity $\overline{v_t}$, and steering wheel angle $\overline{\varphi_t}$.

- $o_0 = \left(o_0^x, o_0^y, o_0^\theta\right)$ be the initial dead-reckoning pose. The orientation component, o_0^θ, is not known and its value is estimated together with the calibration parameters. The position in which the GPS is mounted in the car is known. Hence, for any GPS sample and given the vehicle orientation, the rear axle position (which corresponds to the car position) can be computed. In particular, the values of o_0^x, o_0^y can be computed from g_0 (that is measured by the GPS) and o_0^θ (that is optimized by the system).

- the position components, o_0^x and o_0^y are computed using the fact that the pose, and (ii) g_0 corresponds to the origin of the world coordinate system.

The parameters estimated by the odometry calibration subsystem are $L_{\text{gps}}, m_v, m_\varphi, a_\varphi, o_0^\theta$. These values are obtained by solving the following optimization problem:

$$x^* = \text{argmin}_x \, \text{J}(x) \tag{4}$$

where x represents a set of values for the parameters $L_{\text{gps}}, m_v, m_\varphi, a_\varphi, o_0^\theta$, x^* represents the optimal values of the parameters, and J is the objective function which is defined as the sum of the squared differences between the GPS positions and the respective dead-reckoning positions:

$$J(x) = \sum_t (g_t^x - o_t^x)^2 + \left(g_t^y - o_t^y\right)^2 \tag{5}$$

The Global Best Particle Swarm Optimization algorithm (GBEST-PSO) [73, 74, VER09] is employed to search for a solution for the optimization problem. The GBEST-PSO is an evolutionary algorithm inspired by the movement of a flock of birds through the environment as they search for places with abundance of resources.

The algorithm assumes a fixed-size population of *n* individuals. Each individual is located in a position of the search space. The position of the individual defines a potential solution (values for the parameters $L_{\text{gps}}, m_v, m_\varphi, a_\varphi, o_0^\theta$) for the optimization problem. The individuals are initialized at random positions of the search space, assuming a uniform distribution. After initialization, the individuals "move" over the search space (their positions and velocities are updated) for a number of iterations. As birds in an environment, the individuals tend to move towards the most promising regions, i.e., the positions of the search space in which the value of the objective function is the smaller. After the pre-defined number of iterations, the solution for the optimization problem is given by the position with the smallest value of the objective function considering all individuals and all iterations.

More formally, let $P^t = [p_1^t, \dots, p_n^t]$ be the positions of the individuals at iteration t, where p_i^t is the position of the i-th individual, and $V^t = [v_1^t, \dots, v_n^t]$ be their respective velocities. In all iterations of the algorithm, the velocities and positions of the particles are updated. The velocities are updated considering the current position of the particle, the best position of the particle over all the iterations, and the best position considering all particles and all iterations. The best position of the particle is called the particle personal best position and for the *i-th* particle in the *t-th* iteration it is represented by b_i^t. The best position over all particles is referred as the global best and it is represented by g^t in the t-th iteration. The velocity is updated to encourage the movement of the particles from their current positions in the direction of their personal bests and the global best positions. The velocity is updated by a weighted sum of two vectors, the first one connecting the current position and the personal best position, and the second one connecting the current position and the global best position:

$$v_i^{t+1} = \lambda \left[v_i^t + c_1 r_1 (b_i^t - p_i^t) + c_2 r_2 (g^t - p_i^t) \right] \quad (6)$$

where r_1 and r_2 are random numbers sampled from $Unif(0,1)$, c_1 and c_2 are pre-defined positive constants, and λ is a constriction coefficient given by:

$$\lambda = \frac{2}{\left|2 - \rho - \sqrt{\rho^2 - 4\rho}\right|} \tag{7}$$

where $\rho = c_1 + c_2$.

The factors $c_1 r_1$ and $c_2 r_2$ control the particles' tendencies of moving in direction to the personal or the global best positions, respectively. The constants c_1 and c_2 are fixed hyperparameters used to favour the personal or the global best directions. The values r_1 and r_2 add randomness to the particles' movements which is positive for exploration. The constriction coefficient can be viewed as a recommendation to the particle to "take smaller steps" [74].

After updating the velocities of the particles, their positions are updated by adding the velocities:

$$p_i^{t+1} = p_i^t + v_i^{t+1} \tag{8}$$

Clerc and Kennedy analysed the impact of the hyperparameters in the performance of the algorithm and they found that choosing $c_1 = c_2 = 2.05$ leads to a good performance on average and these values are used in accordance to them [74].

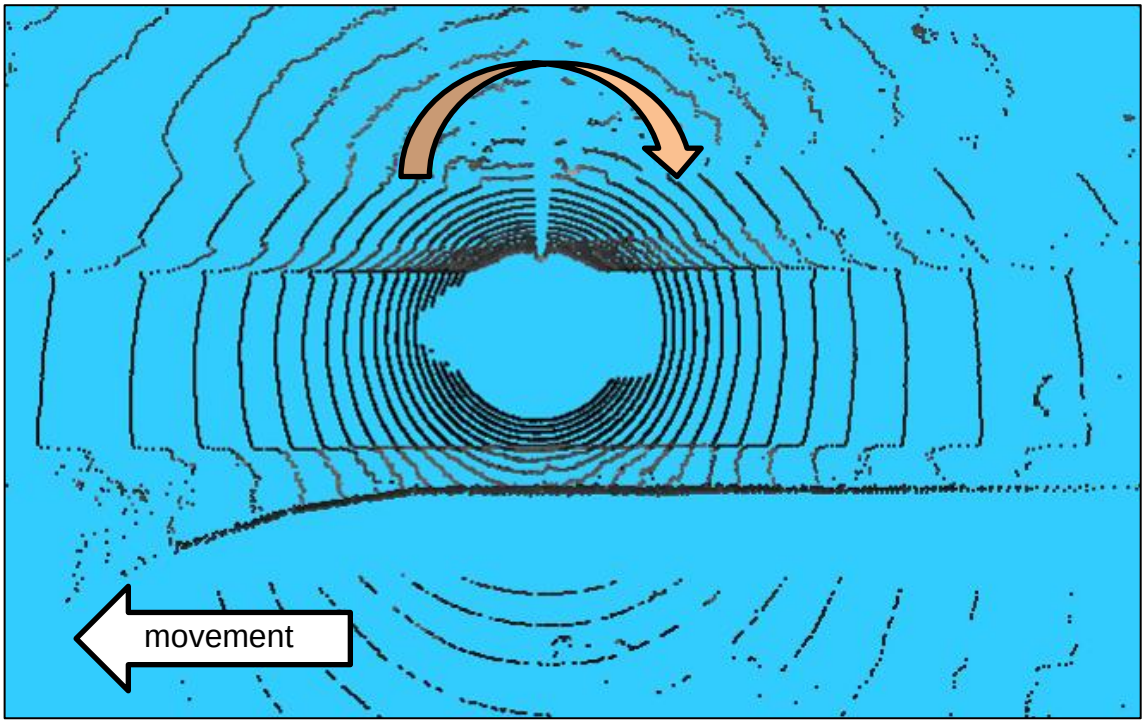
4.3. Velodyne Pre-processing

The pre-processing of Velodyne point clouds consists of eliminating invalid readings, correcting the points' positions considering the movement of the vehicle, and calibrating the reflectivity measurements. The points of Velodyne's point clouds are described in spherical coordinates, i.e., they are represented by a vertical angle, a horizontal angle, and a range value. Points whose ranges are equal to zero or bigger than 70 meters are considered invalid. These measurements usually correspond to incorrect readings due to refraction of rays. Likewise, points that hit the car are also discarded since they do not contain useful information for neither localization, mapping, nor decision making.

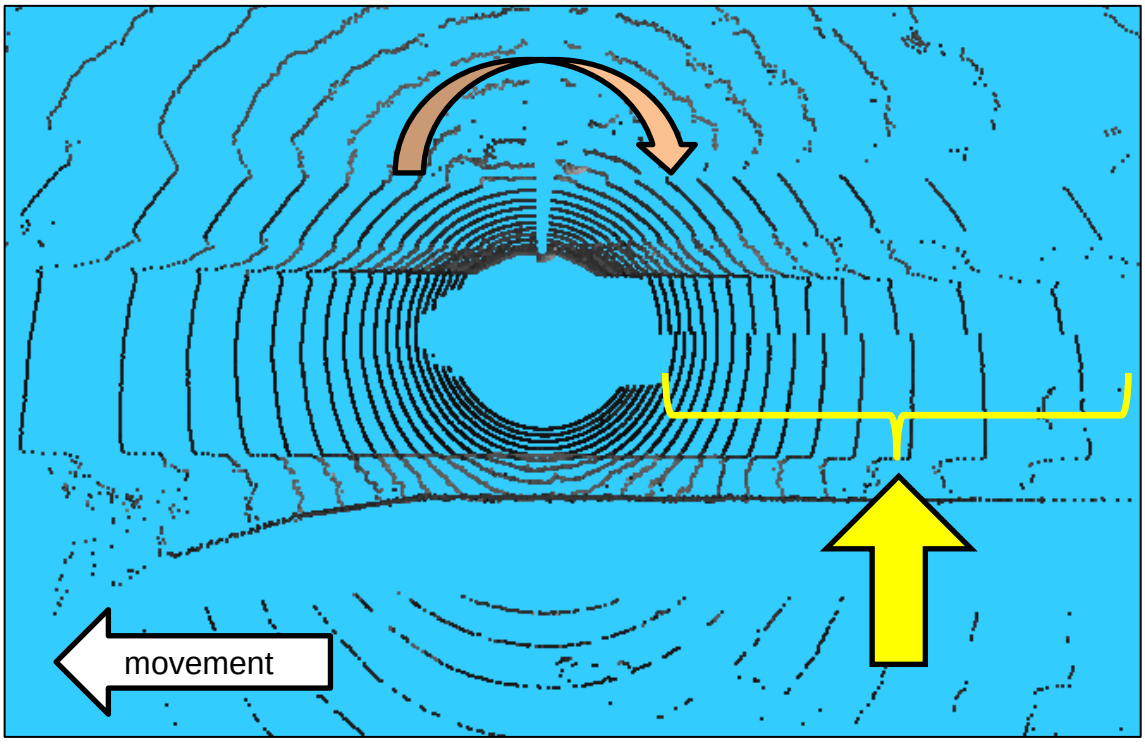
The point clouds are described in relation to the sensors' coordinate system. In order to project the points to the world coordinate system, the pose of the sensor in the car, as well as the pose of the car in the world have to be known. However, if the car is moving, the car pose in the beginning of the sensor revolution around the z-axis will be different from the car pose in the end of the revolution. To account for this fact, the car movement since the beginning of the revolution is estimated for each shot using the motion model presented in Section 3.3. This estimate is used to "correct" the car pose before projecting the points of the shot to the world coordinate system. A visualization of point clouds from Velodyne without considering the car movement, and considering it are presented in Figure 8 (a) and in Figure 8 (b). Note that because the car is moving the start and end of the revolution do not match in Figure 8 (b).

Due to physical properties of the 32 lasers of Velodyne such as their power and the sensibility of their sensors, the lasers can return different reflectivity values when they hit objects with the same brightness. This lack of consistency among the lasers is problematic for both mapping and localization [24]. We employed a novel method for calibrating the reflectivity of the different lasers. The method is similar to the one proposed by Levinson and Thrun [24]. Visualizations of point clouds before and after calibration are presented in Figure 10 (a) and (b), respectively.

Our method receives as input a log and the car poses associated with the data packages, and it outputs a calibration table. Each position of the calibration table stores a calibrated reflectivity value. Given a measurement from a laser (point in spherical coordinates and uncalibrated reflectivity value), a triple (l, d, r) is built, where $l \in [0, 31]$ is the index of the laser that performed the measurement, $d \in [0, 10]$ is a class based on the distance from the sensor to the object hit by the laser, and $r \in [0, 255]$ is the uncalibrated reflectivity. The values l, d, and r are used to access a position of the calibration table (see Figure 9). The calibration table can be seen as a (discrete) function mapping triples in the format presented above to calibrated reflectivity values.

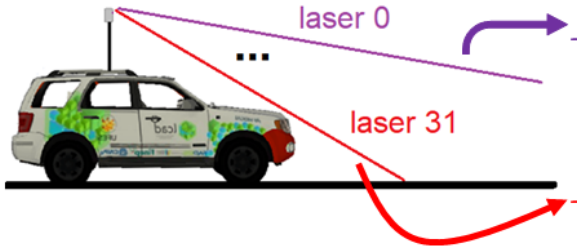


(a)



(b)

Figure 8. Visualization of point clouds from Velodyne (a) without considering the car movement (before calibration), and (b) considering the car movement (after calibration). Note that because the car is moving the start and end of the revolution do not match in (b). The orange arrow represents the sensors' rotation direction and the yellow arrow in figure (b) highlights the part of the point cloud that does not match after calibration.



Laser Id	Raw Reflectivity	Distance Index	Calibrated Reflectivity
0	0	0	5
0	10	5	12
0	5	1	0
…	…	…	…
31	42	9	30
31	255	2	245
31	0	9	5

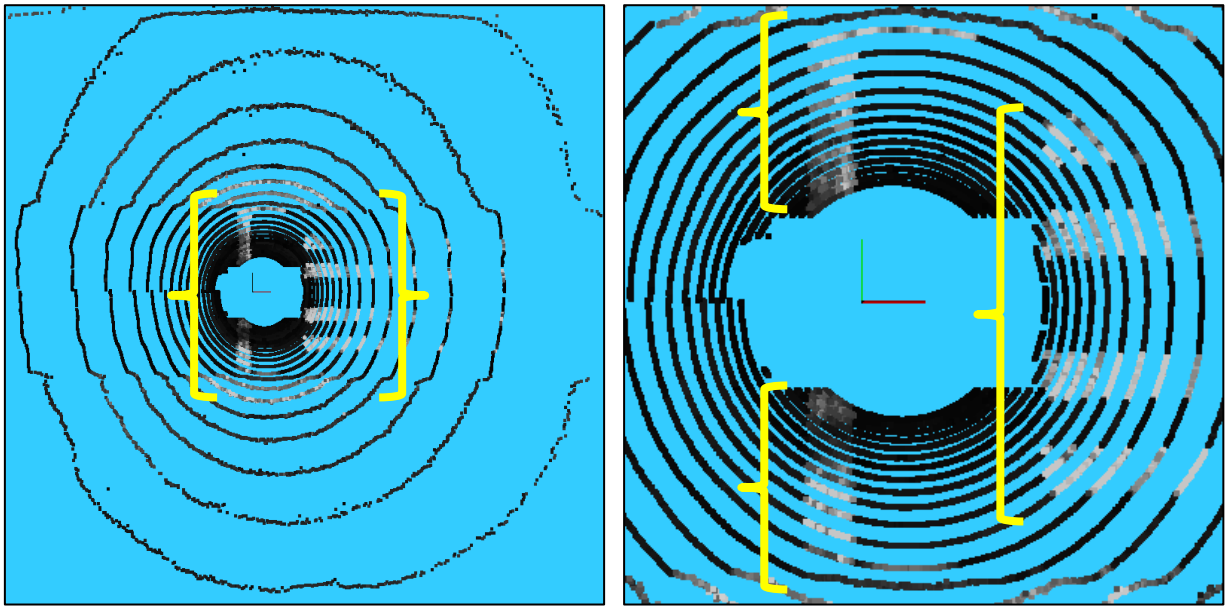
Figure 9. Illustration of the reflectivity calibration table. The sensor is composed of 32 rays, two from which are presented in the figure. The indices of the lasers index the table along with the raw reflectivity reading and the index computed from the distance.

The distance-based class used to index the second dimension of the table is obtained as follows. First, the distance from the sensor to the object hit by the laser in the ground plane is computed. Then, we discretize the values of the distance in slots such that each slot is 50% bigger than the previous one. The class assigned to a measurement represents the slot that contains it. More formally, let g be the distance from the sensor to the object in the ground plane. Then, if g is smaller than a base threshold b, the class zero is assigned to the measurement. Else, the class is given by:

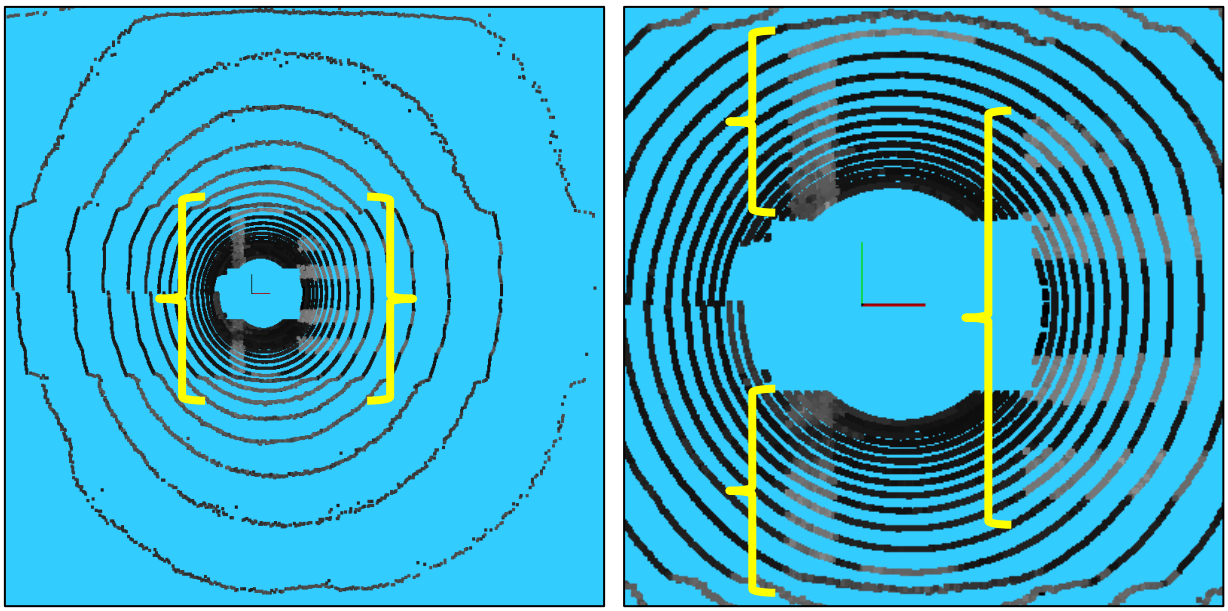
$$\text{class} = \text{truncate}\left(\frac{\log(g-b-1)}{\log(1.45)} + 0.5\right). \quad (9)$$

where 1.45 represents the size of the first slot. This value was chosen empirically by visually inspecting the result of the calibration. The class 9 is returned if the result of the truncation is bigger than nine, and the class 0 is returned if it is smaller than zero.

In order to create the calibration table, a grid map is built using the log and the car poses. Each cell of the grid map stores all Velodyne measurements that hit the cell, along with their respective lasers' indices. We assume that the measurements that hit the same cell should have returned the same reflectivity values. This value is estimated by the mean of the uncalibrated reflectivity values considering all measurements of all cells. We assume that the sensor calibration does not change over time. The calibration table is produced using one log and used for all other logs.



(a)



(b)

Figure 10. Example of point clouds from Velodyne before (a) and after (b) the calibration of the reflectivity measurements. The images on right are zoomed in views of the images in the left. The regions highlighted with yellow braces correspond to a crosswalk and they should present nearly the same reflectivity. As the reader can observe in (a), without the calibration the reflectivity values present significant variation. Considering calibration (b), the reflectivity values are more uniform.

4.4. Visual Semantic Segmentation

In order to obtain semantic information for mapping and localization, we performed visual semantic segmentation of the images captured by the front-facing camera. Visual semantic segmentation consists of assigning a class (e.g., building, pavement, tree, pedestrian, and car) to each pixel of an image. A deep neural network, in particular, the DeepLabv3+ model [44] pre-trained on the Cityscapes dataset[1] [75] is used for segmenting the images. The pre-trained model was chosen because of its high accuracy in preliminary qualitative analyses.

Before using the images as input to the deep neural network, they are pre-processed. Figure 11 illustrates the steps for using the neural network for visual semantic segmentation. The images are captured with resolution of 640 x 480 pixels, and rectified prior to use. They are cropped by 40 pixels in the top and 110 pixels in the bottom, and resized to 513 x 264 pixels. This crop removes regions in the images that correspond in most part to the sky and parts of the self-driving car. Besides that, the crop and resize make the image consistent with the input size expected by the neural network.

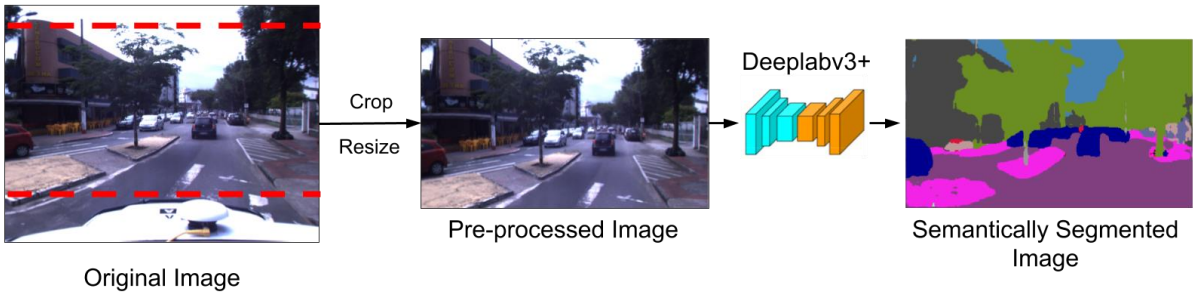


Figure 11. Steps used in our work for visual semantic segmentation of images using the deep neural network Deeplabv3+. The images are captured by the camera mounted on the self-driving car.

[1] The trained model was downloaded from https://github.com/rishizek/tensorflow-deeplab-v3-plus

4.5. Fusion of Velodyne and Images

Velodyne's point clouds can be fused with images for benefiting from both Velodyne's precise range measurements, and colours and/or semantic labels from images. This sensors fusion is performed by projecting each 3D point captured by Velodyne into an image pixel, and associating the point with the pixel value. Figure 12 illustrates the process of projecting points captured by Velodyne into the image plane.

Let $P = [X, Y, Z, 1]_T$ be a Velodyne point in the sensor coordinate system represented in homogeneous coordinates, and $p = [x, y, z]_T$ be the projection of P into the image plane, also in homogeneous coordinates. The transform that computes p from P is given by:

$$p = F \; T_v^i P \tag{10}$$

where T_v^i is a 4 x 4 matrix that projects points from the sensor coordinate system to the camera coordinate system, and F is a 3 x 4 matrix that projects points from the camera coordinate system into the image plane. The F matrix is given by [76, 77]:

$$F = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{11}$$

where f_x and f_y are the focal length in x and y directions, and c_x and c_y are the coordinates of the centre of the image. The pixels column and row can be obtained from *p* by diving the *x* and *y* coordinates by *z* [76], respectively.

The subset of the Velodyne points visible by the camera are associated with the values of the pixels they hit. The point cloud resultant from the fusion of Velodyne's data and colour images will be referred as a colour point cloud. Likewise, the expression segmented point cloud will be used to refer to the fusion of Velodyne's point cloud with semantically segmented images. Figure 13 illustrates the points visible by the camera projected into the image plane. Figure 14 presents a comparison of an instantaneous map built using the complete point cloud from Velodyne and an instantaneous map built using the respective colour point cloud. Similarly, Figure 15 shows the complete process of semantic segmentation of images

along with the projection of points into the segmented image, and Figure 16 compares the instantaneous maps built using the complete point cloud and the semantic point cloud.

It is worth noting that the fusion of LiDAR and camera is not a requirement of the proposed method. Stereo cameras or any other sensor capable of producing depth information could be used by the mapping and localization techniques. The fusion of LiDAR and camera is motivated by the accuracy of the range measurements provided by the LiDAR.

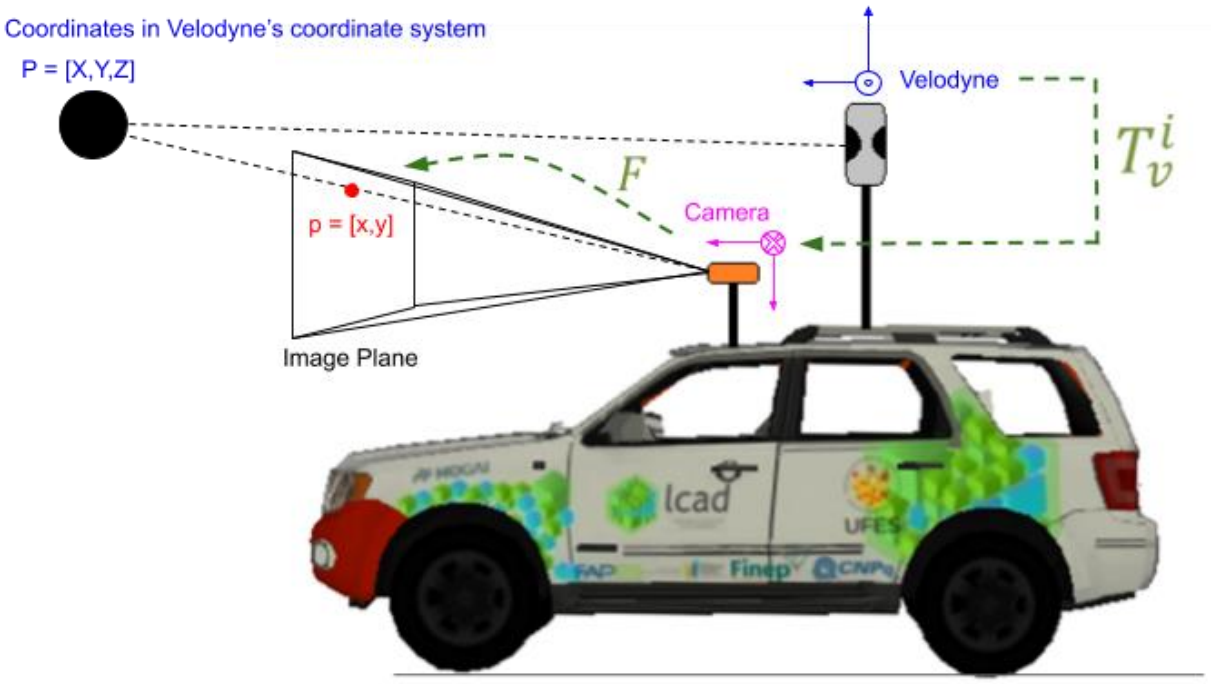
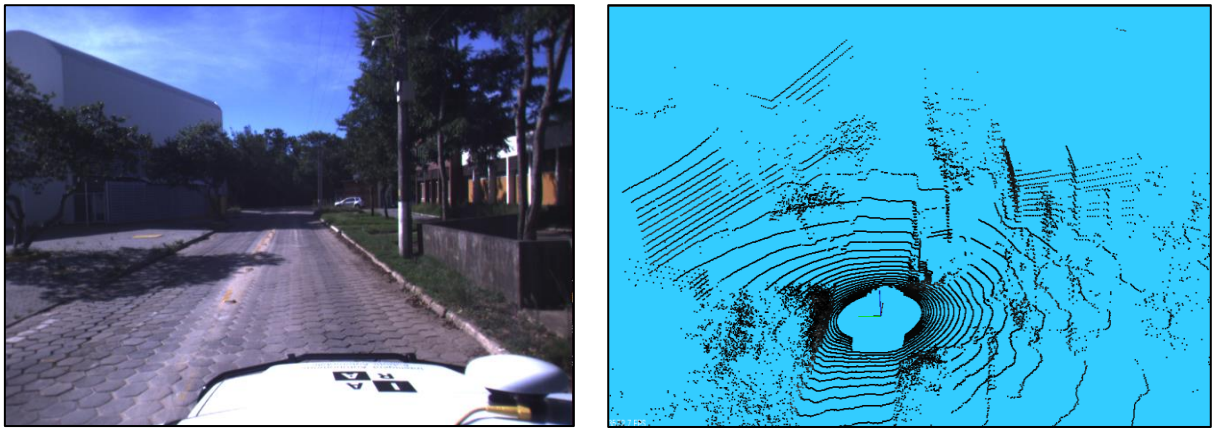
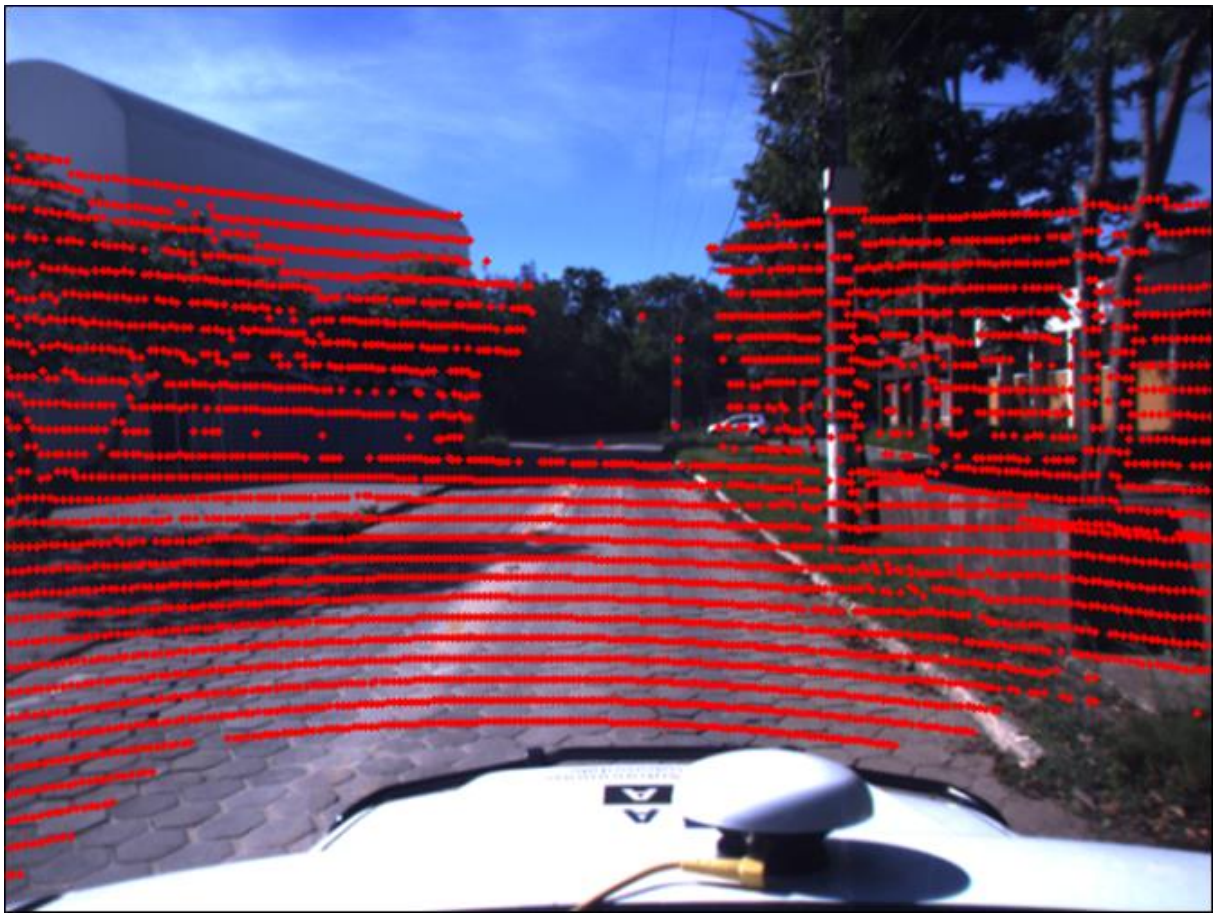


Figure 12. Projection of LiDAR points into the image plane. The matrix T_v^i transforms the point *P* from the LiDAR to the camera coordinate system. The matrix *F* projects *P* into the image plane.

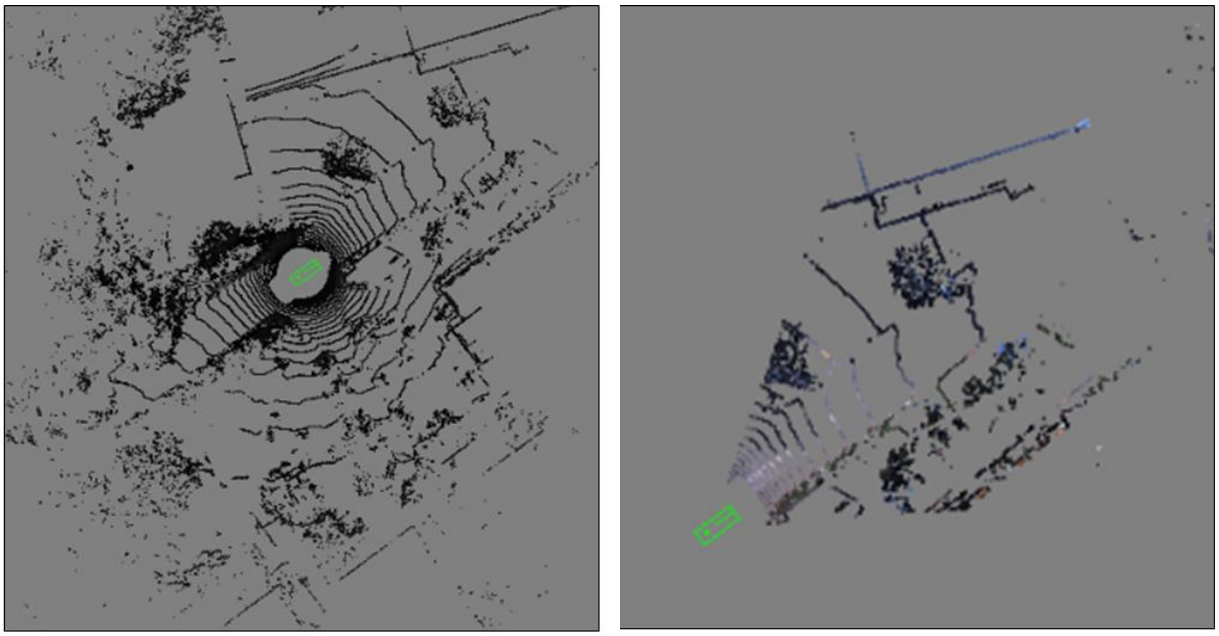


(a) **(b)**



(c)

Figure 13. Projection Velodyne points into the image plane. (a) The current image captured by the front-facing camera. (b) The point cloud captured at the same instant as the image. (c) Projection of the points into the image.



(a) (b)

Figure 14. Visualization of the instantaneous map created by fusing data from Velodyne and camera. Figure (a) presents the complete Velodyne point cloud, and figure (b) presents the subset of points visible by the camera with their associated colours.

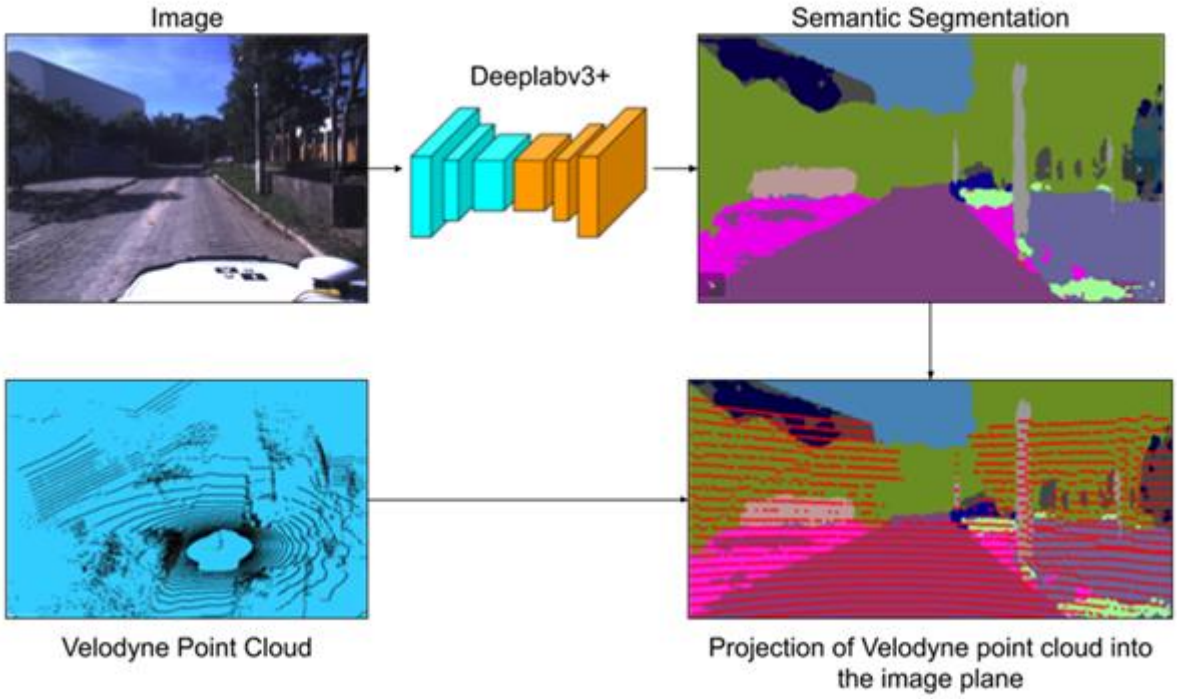
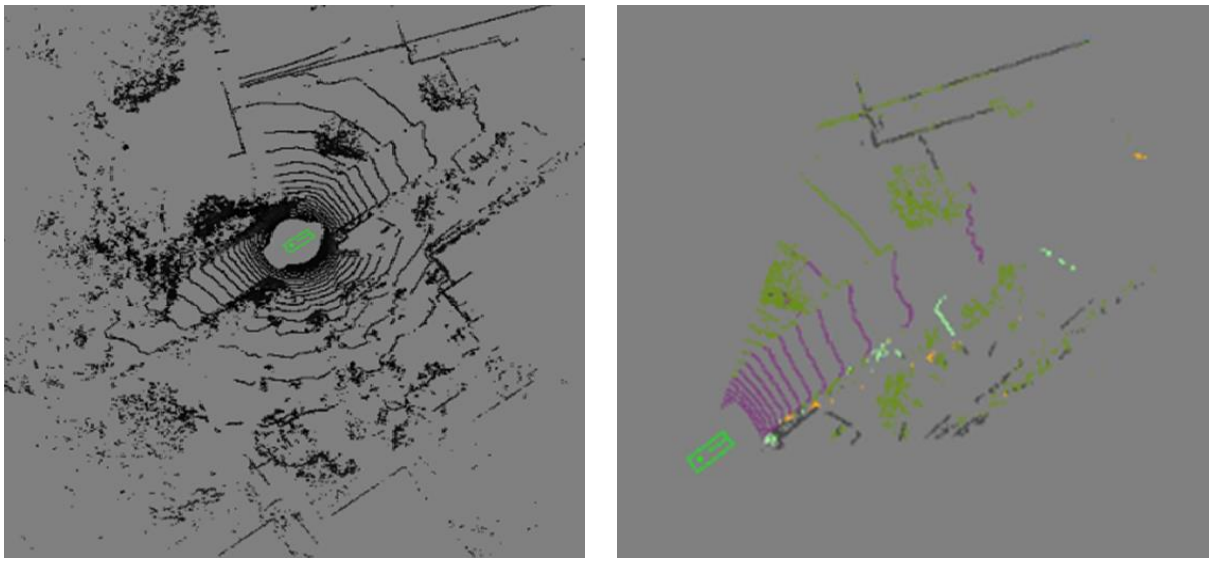


Figure 15. Projection of velodyne points into the image produced by the semantic segmentation deep neural network.



(a) (b)

Figure 16. Visualization of the instantaneous map created by fusing data from Velodyne and the image that results from the semantic segmentation. Similarly to Figure 14, figure (a) presents the complete Velodyne point cloud, and figure (b) presents the subset of points visible by the camera with their associated classes.

5. Novel Techniques for Mapping and Localization

This chapter describes the main contributions of this work: the novel techniques for building grid maps of large-scale environments and for estimating the localization of a self-driving car in them. Initially, Section 5.1 describes the types of grid maps considered in this work, i.e., occupancy, reflectivity, semantic, and colour grid maps are introduced. Then, the localization technique which employs these maps is presented in Section 5.2. Finally, the method employed for building grid maps is described in Section 5.3.

5.1. Grid Maps

In grid maps [78, 20], the environment is discretized in cells with fixed predefined size. Different types of information can be stored in the cells of the grid map depending on the application. In the context of self-driving cars, grid maps are commonly used for localization and decision making. Grid maps usually store statistics. The initial values of these statistics are chosen to reflect some prior knowledge about the environment. As sensors observe the actual state of the environment, the statistics are updated. If no prior knowledge is available, a uniform distribution is traditionally used for initializing the cells.

For localization, it is important to store in the maps discriminant features of the environment (e.g., lane marks and buildings) such that the car pose can be estimated by matching sensors' data and the map. For decision making, being able of easily classifying the cells as occupied by obstacles or free is important. Grid maps that can be used for both localization and decision making are preferable given that storing and operating with multiple maps of potentially large regions requires substantial computational resources.

In this work, we are interested in evaluating the impact of using different types of grid maps in the accuracy of a localization algorithm. Under the assumption that cars move in the ground plane and aiming at minimizing the use of computing and

memory resources, 2D grid maps are employed instead of 3D grid maps. The types of grid maps considered are occupancy grid maps, reflectivity grid maps, semantic grid maps, and colour grid maps. Occupancy and reflectivity grid maps are the most common types of grid maps used in self-driving cars applications [16, 17]. Semantic grid maps are invariant to illumination conditions and sensor modality given robust methods for extracting semantic information from sensors' data. Moreover, semantic grid maps can represent different information that might be useful for planning and decision making. Vision is one of the main senses used by humans, and colour data is rich, informative, and can aid the identification of features of the environment. For these reasons, colour and semantic grid maps are also considered. It is worth noting, however, that others types of grid maps exist in the literature and some of them were even used for localization. Examples are height grid maps [25], lane side grid maps [30], joint height and reflectivity grid maps represented as mixtures of gaussians [25].

In the following, we describe the contents of the grid maps used in this work and, in Chapter 5.3, the technique for creating grid maps is presented. An urban area (presented in Figure 17) is used for demonstrating the different types of grid maps. The self-driving car was driven through the path highlighted in blue and the data captured by the sensors were stored in a log file. The data was posteriorly processed and used for creating the grid maps presented in Figure 18. In particular, the occupancy, reflectivity, semantic, and colour grid maps are presented in Figure 18 (a), Figure 18 (b), Figure 18 (c), and Figure 18 (d), respectively. In Figure 18 (a), blue pixels correspond to unobserved cells, while in the other maps unobserved cells are represented in grey. A detailed view of the area highlighted in red in Figure 17 is presented in Figure 13. The different grid maps of the detailed area are presented in Figure 14.

Figure 17. An urban area used for demonstrating the different types of grid maps. The self-driving car was driven though the path highlighted in teal and the data captured by the sensors were stored. The data was posteriorly processed and used for creating the grid maps presented in Figure 18. A detailed view of the area highlighted in red is presented in Figure 19, and different grid maps of the detailed area are presented in Figure 20.

5.1.1. Reflectivity and Colour Grid Maps

Reflectivity grid maps [23, 24, 29, 25, 8] and colour grid maps [60, 79] store, for each cell, the mean and variance of a Gaussian distribution representing, respectively, the reflectivity and colour of the objects inside the area delimited by the cell. In reflectivity grid maps, cells with high variance represent parts of the environment that change often, e.g., cells occupied by moving obstacles. In colour grid maps, this analysis is less obvious since a cell may present high variance because of other factors, such as, abrupt changes in the illumination of the scene.

5.1.2. Occupancy Grid Maps

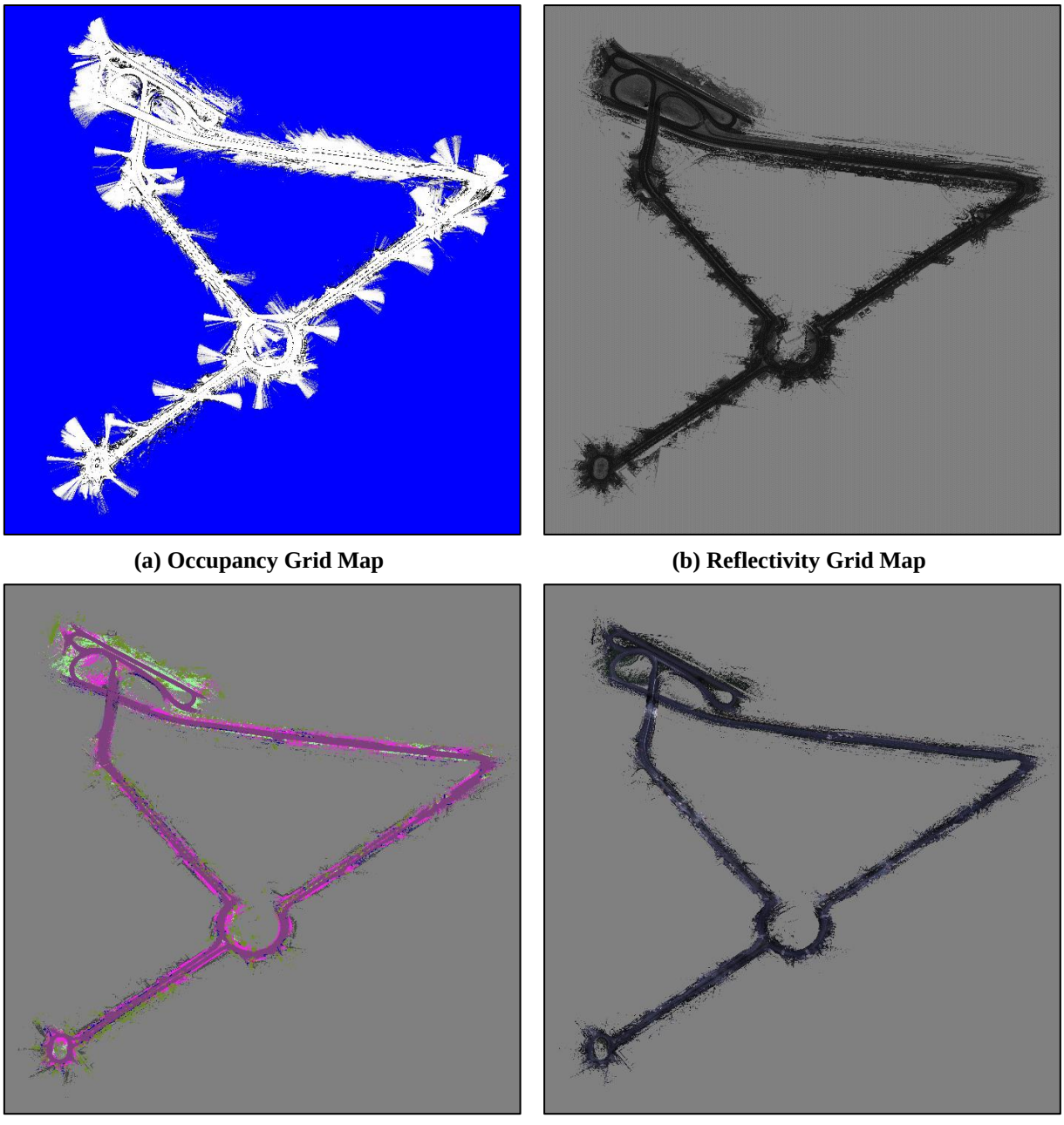
Occupancy grid maps [78, 20] store the probability of the cells being occupied by obstacles. Whenever a measurement is received from the LiDAR, two updates are performed in the map. First, the occupancy probability of the cell at the given range is updated according to the evidence that the ray hit an obstacle or not. The technique for computing the evidence that a laser ray hit an obstacle is illustrated in Figure 21. Assuming that the height of the sensor is known and that the car is moving in a plane, we can use the i-th laser from Velodyne for estimating the position in the floor that the next laser (the i+1-th laser) will hit if the area is free from obstacles. However, if an obstacle exists, then the distance in the floor between the sensor and the object hit by the laser will be smaller than expected. The difference between the expected distance in the floor, and the distance measured by the sensor is used for computing the evidence that the ray hit an obstacle.

Let δ_e be the expected distance between consecutive rays in the floor, and δ_m be the measured distance. Then, the evidence e that the laser i+1 hit an obstacle is given by:

$$e = \frac{\delta_e - \delta_m}{\delta_e} \tag{12}$$

If the evidence of obstacles is higher than a threshold, the point relative to laser *i+1* is classified as being an obstacle, else it is classified as a plane area. The cell hit by the laser is updated according to the evidence of obstacle.

The second update is a raycast technique which consists of decreasing the occupancy probabilities of all cells between the first ray that hit the floor and the first ray that hit an obstacle. This update is based on the idea that if obstacles were present in any of these cells, a smaller range would have been measured. The raycast technique also provides a way of “cleaning” cells that were momentarily occupied, e.g., by moving objects. The technique makes occupancy grid maps denser than the other types of maps since only the cells directly observed by sensors are updated in them. The instantaneous map presented in Figure 5 (a) employ the raycast technique. Note the “beams” connecting the points that hit the floor and the obstacles.

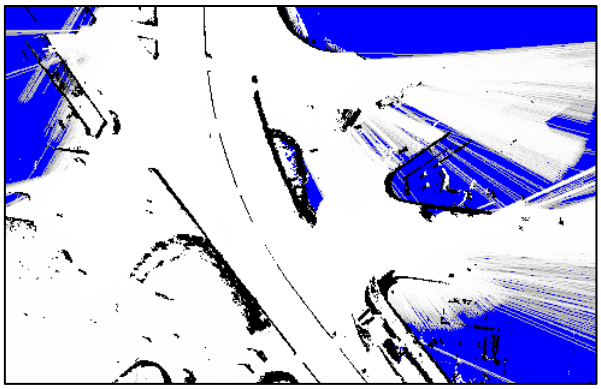


(a) Occupancy Grid Map **(b) Reflectivity Grid Map**

(c) Semantic Grid Map **(d) Colour Grid Map**

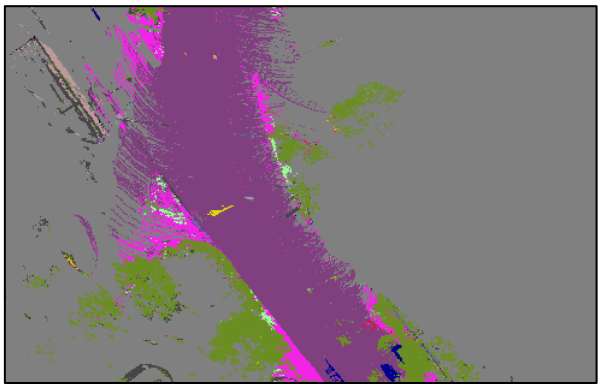
Figure 18: Visualization of the different types of maps from the area presented in Figure 17. (a) Occupancy grid map. (b) Reflectivity grid map. (c) Semantic grid map. (d) Colour grid map.

Figure 19. Detailed view of the region marked in red in Figure 17.

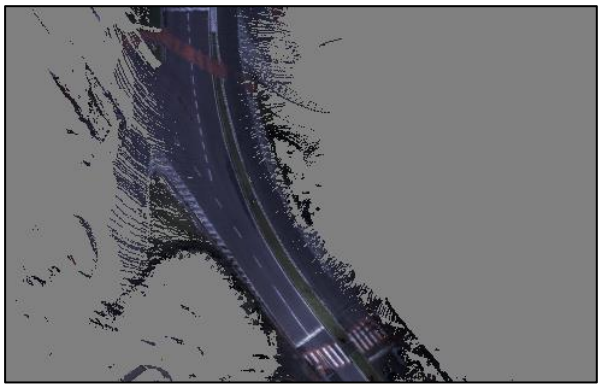


(a) Occupancy Grid Map

(b) Reflectivity Grid Map



(c) Semantic Grid Map



(d) Colour Grid map

Figure 20: Visualization of different types of maps of the region presented in Figure 19. (a) Occupancy grid map. (b) Reflectivity grid map. (c) Semantic grid map. (d) Colour grid map.

5.1.3. Semantic Grid Maps

In semantic grid maps [36, 37, 38, 39, 63, 40], cells are classified according to the types of objects they contain. Common cell classes in self-driving cars applications are road, building, tree, traffic sign, traffic light, and pedestrian. The semantic information is extracted from sensors data (for example using the technique presented in Section 4.4) and, then, projected into the grid map.

A map cell may receive several samples from different classes. In order to preserve all of this information, each cell of a semantic grid map stores a categorical distribution over classes of objects. This distribution is represented as a vector in which the i^{th} dimension represents the probability that objects of class i exist in the cell. Instead of storing the actual probabilities, in our implementation we store counters representing the number of times each class was observed in the cell. The probabilities can be derived from the counters by normalization. The process of updating the map when changes in the environment are observed is simplified if counters are used instead of the probabilities. The counters can be simply incremented according to the new classes of objects observed in the environment.

To initialize the map, we assume a uniform prior distribution and the classes counters of all cells are set to one. Although the initialization of the counters with zero seems more intuitive since they were not observed by sensors, it would lead to a division by zero when computing the probabilities from the counters. Semantic grid maps can be seen as a generalization of occupancy grid maps to more than two classes.

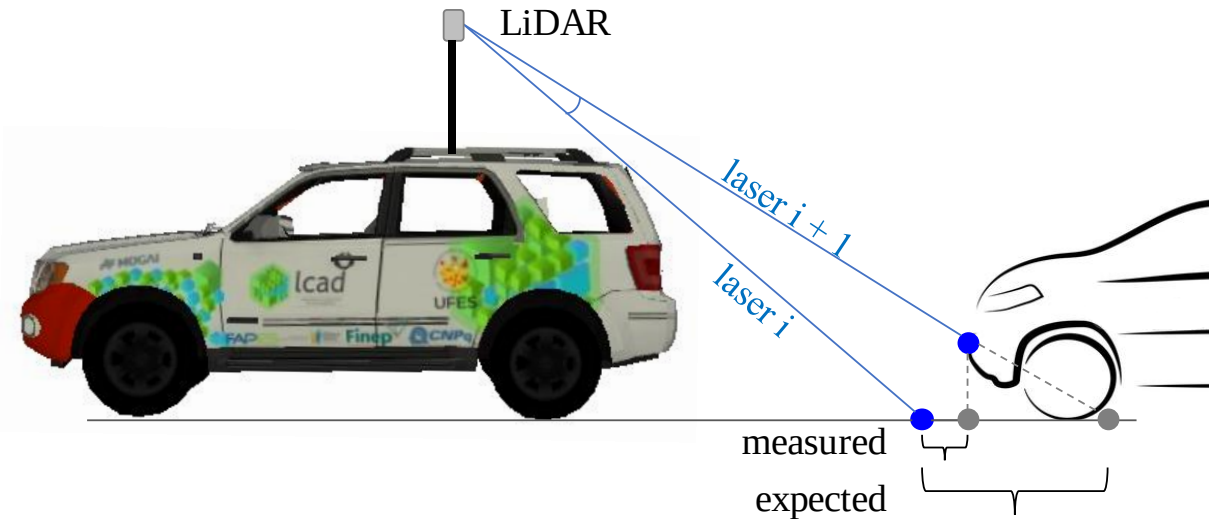


Figure 21. Technique for computing the evidence that a laser ray hit an obstacle. Assuming that the height of the sensor is known and that the car is moving in a plane, we can use the i-th laser from Velodyne for estimating the position in the floor that the next laser (the i+1-th laser) will hit if the area is free from obstacles. However, if an obstacle exists, then the ray size in the floor will be smaller than expected. The difference between the expected distance in the floor, and the distance measured by the sensor is used for computing the evidence that the ray hit an obstacle.

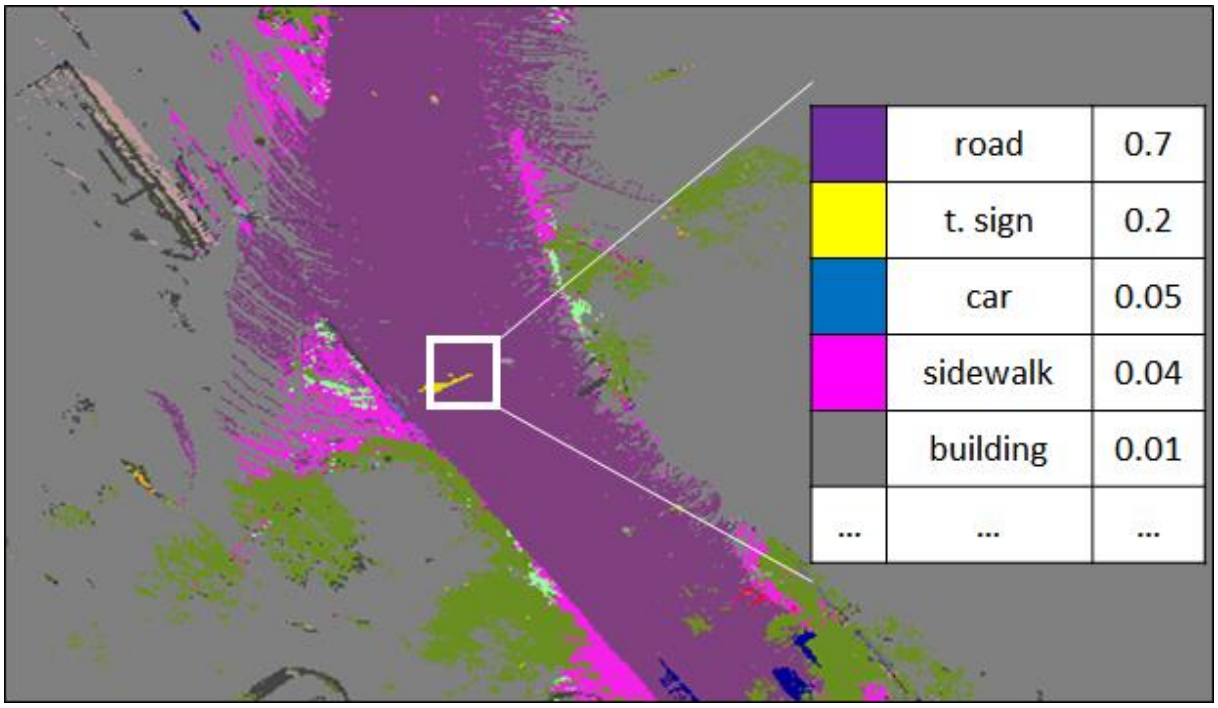


Figure 22. Illustration of semantic grid maps and the contents of their cells. In practice, instead of storing the probabilities of each class, the cells store counters representing the number of rays from each class that have hit the cell.

5.1.4. Comparison of Grid Maps

Each of the grid maps mentioned above have specificities that make them more or less adequate for different applications. Table 1 lists the advantages and disadvantages of the grid maps for self-driving cars localization. The sensitivity of the grid maps to different conditions of illumination and weather depends on the sensor used for mapping and localization. LiDARs and Sonars, for instance, are nearly invariant to illumination conditions. On the other hand, heavy rain and snow can impact the performance of LiDARs. Cameras are more sensitive to changes in illumination and weather. Besides the sensor being used, the sensitivity of semantic grid maps to illumination and weather conditions also depends on the semantic segmentation method.

Occupancy grid maps and semantic grid maps can be used for discriminating free areas from occupied areas, and they are suitable for both localization and motion planning. Extracting this information from reflectivity grid maps and colour grid maps requires additional processing and may be unreliable. On the other hand, because these maps store the “appearance” of the world as projected into the cells, they can lead to more accurate localization than occupancy grid maps in areas with ambiguous structure. Examples of such areas are corridor-like environments (e.g., tunnels, bridges), and roads with featureless surroundings. Since semantic grid maps are more informative than occupancy grid maps, they have the potential for producing better localization estimates in ambiguous environments.

Grid maps used for self-driving cars’ localization should reflect the static structure of the environment. Potentially dynamic objects such as parked vehicles, moving vehicles, and pedestrians, should not be part of the map, since they not necessarily will be present in future sessions of autonomous operation. Semantic grid maps can provide natural ways of handling these potentially dynamic objects if the semantic segmentation method is capable of discriminating dynamic and static objects. This capacity is also important if the map is intended to be used for decision making. Identifying which parts of the environment are dynamic or static in occupancy, colour, and reflectivity grid maps require additional post-processing and can be challenging.

Table 1. Advantages and disadvantages of different grid maps for self-driving cars localization.

Reflectivity Grid Maps	Occupancy Grid Maps
• Resilient to different illumination. • Sensitive to heavy rain and snow. • Represent the “visual” appearance of the environment. • Discriminating obstacles from free areas require post-processing. • Discriminating dynamic obstacles from static ones require post-processing. • Requires the use of LiDARs. • Require pre-processing for calibrating the reflectivity values.	• Resilience to different illumination and weather conditions depends on the sensor. • Represent the structure of the environment. • Discriminate obstacles from free areas. • Discriminating dynamic and movable obstacles from static ones require post-processing. • Can be used with different sensors.
Colour Grid Maps	**Semantic Grid Maps**
• Sensitive to different illumination. • Represent the visual appearance of the environment. • Discriminating obstacles from free areas require post-processing. • Discriminating dynamic obstacles from static ones require post-processing. • Requires the use of cameras. • Require pre-processing for obtaining depth information.	• Resilience to different illumination and weather conditions depends on the sensor and the semantic segmentation method. • Depending on the classes produced by the semantic segmentation method, can represent both the appearance and the structure of the environment. • Discriminate obstacles from free areas. • Discriminating movable obstacles from static ones. • Different sensors can be used. • Depending on the sensor used, it may require pre-processing for obtaining depth information and extracting semantic data (which is potentially expensive).

The costs of self-driving cars can determine their widespread adoption. Semantic and occupancy information can be extracted from different sensors. In principle, precise and potentially costly sensors can be equipped in specialized mapping platforms for building high quality maps, while low cost sensors can be equipped in the end users' self-driving cars for localization in occupancy and semantic grid maps. Reflectivity grid maps require the use of LiDARs which are currently expensive sensors. Colour grid maps rely on cameras whose costs can be lower. However, extracting depth information from cameras still challenging. Moreover, although there are promising works in this direction [80, 81, 82, 83, 84], depth measurements obtained from cameras are less precise than the measurements obtained using LiDARs.

5.1.5. Tilling

The grid maps are stored and used as a set of squared fixed-sized tiles as illustrated in Figure 23. Only the tile that currently contains the car along with the eight tiles around it are maintained in primary memory (see [8]). By doing so, the amount of primary memory used by the system is constant, even when operating in large-scale environments.

5.2. Localization

The proposed localization technique estimates the state of self-driving cars using offline maps (see Section 3.2.1). In our work, a state $s = (v, b, \varphi, p)$ is composed of a linear velocity v, a steering wheel angle bias b, a steering wheel angle φ, and a 2D pose $p = (x, y, \theta)$. A particle filter algorithm [19, 20, 21 22] is employed for localization. The particle filter maintains set of samples (the particles) representing possible values of the current car state. Each sample represents a potential value of the state. Weights are associated with the samples representing their probabilities.

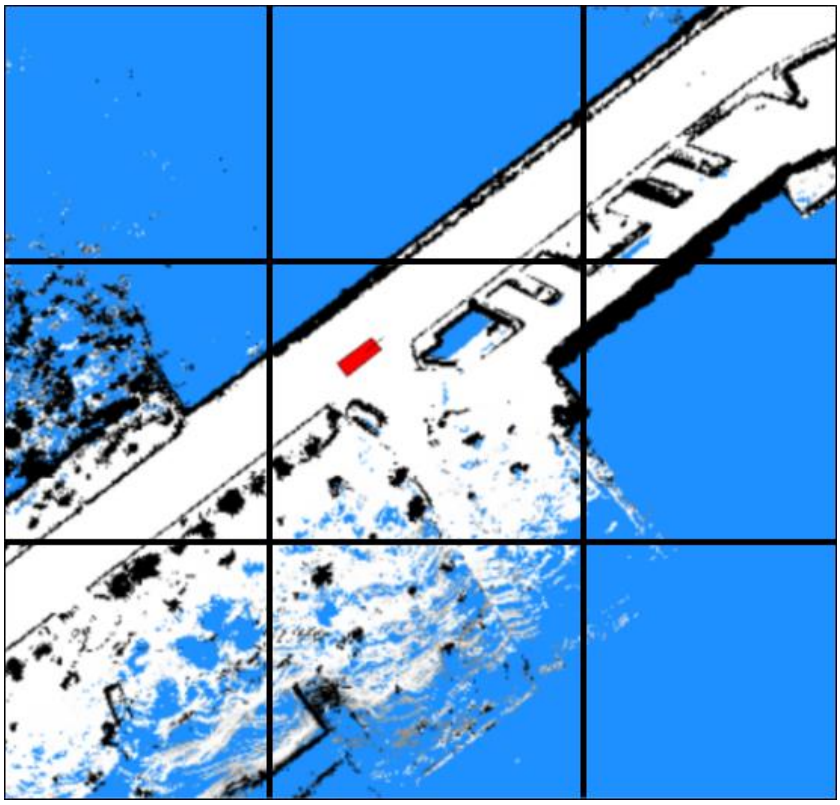


Figure 23. Visualization of the tilling scheme used for limiting the memory consumption of the system.

After initialization, the samples and their weights are iteratively updated as new data packages are observed. Each update consists of a prediction step, followed by a correction step. In the prediction step, the particles values are updated to account for changes in the state since the last data package (e.g., due to the movement of the car). In the correction step, the likelihoods of the particles are evaluated and a new set of particles is produced by resampling [20]. After the prediction and correction steps, a point estimate of the state is computed using the set of particles. The expected value of the particles (the mean) is commonly used for producing the estimate.

The remaining of the chapter is organized as follows. Sections 5.2.1 and 5.2.2 describe the prediction and correction steps, respectively, and Section 5.2.3 presents the technique for initializing the particle filter. The following notation will be employed in the chapter: lower case letters represent concepts such as, states, states' values and sensor measurements. Subscripted letters represent indices of synchronized

data packages. Superscripted letters represent the index of a particle. For conciseness, the estimates obtained after the prediction and correction steps will, respectively, be called predicted and corrected values. A bar in the top of a letter represents the predicted value for that concept. For instance, $\bar{s}_t^i$ represents the predicted state value of the i-th particle given the t-th synchronized data package.

5.2.1. Prediction

In the prediction step, data from the odometer are used for updating the particles' values. Let $s_{t-1}^i = (v_{t-1}^i, b_{t-1}^i, \varphi_{t-1}^i, p_{t-1}^i)$ be the corrected state value of the i-th particle given the data package (t - 1) and the offline map, and $\bar{s}_t^i$ be the predicted state value for the same particle given the t-th data package. Then, $\bar{s}_t^i$ is given by:

$$\bar{s}_t^i = \begin{bmatrix} \bar{v}_t^i \\ \bar{b}_t^i \\ \bar{\varphi}_t^i \\ \bar{p}_t^i \end{bmatrix} = \begin{bmatrix} v_t \\ b_{t-1}^i \\ \varphi_t + \bar{b}_t^i \\ p_{t-1}^i \end{bmatrix} + \begin{bmatrix} \epsilon_v \\ \epsilon_b \\ \epsilon_\varphi \\ M_t^i + \epsilon_p \end{bmatrix} \tag{13}$$

where v_t and φ_t are the linear velocity and the steering wheel angle from the t-th data package; $\epsilon_v \sim \mathcal{N}(0, \sigma_v + \sigma_v^\varphi), \epsilon_\varphi \sim \mathcal{N}(0, \sigma_\varphi + \sigma_\varphi^v), \epsilon_b \sim \mathcal{N}(0, \sigma_b)$ represent the errors in v_t, φ_t, and b_{t-1}^i and they are sampled from normal distributions with zero mean and predefined standard deviations. The values σ_v and σ_v^φ are, respectively, the standard deviation of v_t and the squared root of the covariance of φ_t in relation to v_t; σ_φ and σ_φ^v are, respectively, the standard deviation of φ_t and the squared root of the covariance between v_t and φ_t; σ_b is the standard deviation of the bias in the steering wheel angle. M_t^i is an estimate of the car's movement since the last localization step which is estimated using the motion model (Section 3.3), $\bar{v}_t^i$ $\bar{\varphi}_t^i$, and the amount of time since the last localization step. The value $\epsilon_p \sim \mathcal{N}(0, \Sigma_p)$ is a random noise sampled from a normal distribution with zero mean, and covariance matrix Σ_p.

The term ϵ_p is added to account for physical phenomena not considered in the motion model (e.g., drift) and imprecisions in the map representation (e.g., due to discretization of the world in cells). The previous update to the bias of the steering

wheel angle is only applied when $\bar{v}_t^i$ is higher than a threshold. In this is not the case, the bias is maintained it is. The value $\bar{b}_t^i$ is clamped to prevent it from growing to unrealistic values.

The update rule for the components other than the pose can be intuitively understood as follows. It was empirically observed that the bias in the steering wheel angle is systematic, but changes slowly. Because of that, we modelled it as a latent variable that evolves according to a random walk with Normal “steps”. The velocity and steering wheel angle measurements are also subject to unobservable normal error. To try and correct these values, besides adding the bias to the steering wheel angle, the measurements are also corrupted by noise. The corrupted measurements are then used as input for the motion model for estimating the car movement. After the correction step and assuming that the filter did not diverged, the most likely particles will be the ones in which the estimated movement is close to the real movement of the car. In these particles, the predicted bias, linear velocity and steering wheel angle are likely to be close to the truth since they are used for estimating the car movement.

5.2.2. Correction

The correction step is itself subdivided into likelihood computation, and resampling. Likelihood computation consists of evaluating the probability of the particles being the true state of the car. Resampling, on the other hand, is the process of sampling a new set of particles considering the particles’ likelihoods.

Data from the most recent point cloud, and the offline map are used for computing the likelihoods. An instantaneous grid map is built using the point cloud. the likelihoods of the particles are computed assuming independence between the cells in the instantaneous map. With this assumption, the particles’ likelihoods are computed as the product of the likelihoods of each cell in the instantaneous map given the offline map and the particle value. Only the observed cells of the instantaneous map are considered for localization.

Rays that hit moving obstacles or structures that were not present during mapping can lead to inconsistencies in the particles' likelihoods. Due to these noisy measurements (outliers), incorrect particles may receive high weights, while low weights may be assigned to correct particles. To try and minimize this issue, an outlier rejection approach is employed. If the likelihood of a sensor measurement is lower than a threshold for most of the particles, then the measurement is discarded and not considered for computing the particles' likelihoods.

The instantaneous map is built once assuming that the car is in the centre of the map and then, for each particle, the positions of the cells in the offline map are computed using the particle pose. In urban scenarios, the number of cells in the instantaneous map is significantly smaller than the number of points in the point cloud. This is because laser rays that hit the same obstacles are mostly projected into the same cells. Hence, in these scenarios, generating the instantaneous maps and then projecting its cells into the offline maps is more computationally efficient than projecting all points from the original point cloud (remember that the projection is performed for each particle).

Let $z_1, \dots, z_k$ be k sensors measurements, i.e., the set of observed cells in an instantaneous grid map, m be the offline map, and w_i and s_i be, respectively, the weight and state of the *i-th* particle. The weight w_i is defined to be the likelihood of $z_1, \dots, z_k$ given s_i and m. Assuming that $z_1, \dots, z_k$ are conditionally independent given s_i and m, w_i is given by:

$$w_i = \prod_{j=1}^{k} p\left(z_j \middle| s^i, m\right) \tag{14}$$

Since $p\left(z_j \middle| s^i, m\right) \leq 1$ for all i and j, and because the floating-point representation in computers is limited, in most cases the particle weights are truncated to zero. To account for this issue, the log function is used to turn the product in equation (14) into a sum of log likelihoods:

$$l_i = \sum_{j=1}^{k} \log p(z_j | s^i, m) \tag{15}$$

Let $l_{min} = min_j\, l_j$ be the smallest log likelihood among all particles. The value l_{min} is subtracted from the l_i terms to make them positive. Then, the particles' weights are obtained by normalizing the resulting values:

$$w_i = \frac{l_i - l_{min}}{\sum_{j=1}^{n} l_j - l_{min}} \tag{16}$$

In order to compute $\log p(z_j|s^i, m)$, first the particle state is used to calculate which cell of m contains the measurement z_j. Let c_j^i be the value of this cell. Then, $\log p(z_j|s^i, m)$ is obtained by comparing the contents z_j and c_j^i. Depending on the type of map used by the system, different measures are used for comparison.

In semantic grid maps and occupancy grid maps, the $\log p(z_j|s^i, m)$ is obtained by evaluating the probability of z_j's class in c_j^i and, then, applying the log function. For reflectivity grid maps, the Mahalanobis distance is employed:

$$\log p(z_j|s^i, m) = \left(z_j - c_j^i\right)^T {\Sigma_z}^{-1} \left(z_j - c_j^i\right) \tag{17}$$

where Σ_z is the covariance matrix for the measurement z_j.In reflectivity grid maps, the cells contain a single component, and the covariance matrix reduces to the variance of z_j. The Mahalanobis distance is obtained by computing the log probability of a Normal distribution and discarding terms that are constant for all particles

Different from the previous methods, for colour grid maps, the particles weights are given by the entropy correlation coefficient (ECC) [85] between the cells of the instantaneous map and their respective cells in the offline map:

$$ECC(z_{1:k}, c_{1:k}) = 2 - \frac{2\, H(z_{1:k}, c_{1:k})}{H(z_{1:k}) + H(c_{1:k})} \tag{18}$$

where the $H(X)$ represents the marginal entropy of the random variable X:

$$H(X) = -\sum_x p(x) \log p(x) \tag{19}$$

and $H(X, Y)$ represents the joint entropy between the random variables X and Y:

$$H(X,Y) = -\sum_{x}\sum_{y} p(x,y)\log p(x,y) \quad (20)$$

The ECC is scaled to (0, 1), such that 0 indicates full independence and 1 represents complete dependence between the input values. The ECC is quite similar to the normalized mutual information metric and both of them were vastly employed for registration of images from different sources (e.g., medical images from different sensors) [86, 87, 88, 89]. The ECC is chosen instead of the Mahalanobis distance to compare colour data to try and minimize the negative effects of different illumination conditions.

After computing the likelihoods of the particles, the resample step takes place. It consists of sampling a new set of particles from the previous ones considering their likelihoods. We used the low variance resampling method presented by Thrun, Burgard, and Fox [20] to do so.

5.2.3. Initialization

When starting operation, the self-driving car can be in any place of the environment. Estimating the vehicle pose without some kind of prior knowledge requires solving the challenging global localization problem. Although the global localization problem is addressed in the literature [90], as far as we know the current solutions are not accurate enough for practical use in self-driving cars.

To avoid the global localization problem, we assume that a guess for the initial pose of the car is given by GPS, or set manually. After initializing the particle filter, global information is no longer necessary. The particles' poses are sampled from a Gaussian distribution with the mean being the initial guess, and the covariance representing the error of this guess (e.g., the error of the GPS measurements as informed by the manufacturer). The remaining components of state, i.e., the velocity, the steering wheel angle, and the bias in the steering wheel angle, are sampled from Gaussian distributions with zero mean, and predefined variances. The sampled particles are assumed to be equally likely. During the first ten localization steps (including prediction and correction), Gaussian noise is added to the particles' values

using the same covariances used for initialization. By adding noise, we increase the chance of producing a particle that represents the true state of the vehicle.

An elitism strategy is employed in the particle filter. One of the particles is always chosen to be equals to the mean of the particles. This elite particle is initialized as follows. The pose is set to be exactly equal to the initial guess (without adding noise to it), the bias of the steering wheel angle is set to zero, and the velocity and steering wheel angle components are set to be equals to the values provided by the odometer. This elitism scheme is an attempt of maintaining a particle with reasonable value even when the particles spread.

If a particle with high likelihood is present in the set, most of the other particles will be discarded during resample. However, if most of the particles have similar low probabilities, they are likely to be maintained. Since the next prediction steps will introduce noise, the particles will spread. Therefore, the confidence of the particle filter in the state estimate can be evaluated online by analysing the particles likelihoods and how spread they are.

5.3. Building Grid-Maps

The technique for building grid maps receives as input a log and it outputs reflectivity, occupancy, colour, and semantic grid maps. The log is created by driving the self-driving car through an environment and storing the sensors' data. After that, the sensors' data are pre-processed as described in Chapter 4. Finally, the states of the self-driving car along the log are estimated and they are used for projecting the sensors' data into the grid maps. As mentioned in Section 1.3, the technique we propose extends and improves our previous work [8]. The GraphSLAM algorithm [23, 24, 8, 25] is used for estimating the states of the self-driving car. In GraphSLAM, the variables to be estimated (in our case the set of vehicle states) are represented as nodes, and sensors measurements and their covariances are represented by edges.

Our technique comprises a sequence of steps. First, the GraphSLAM algorithm is used to fuse data from GPS and odometry producing an initial guess for the states' values. The result of this step is referred as the fused odometry path. We search for

loop closure regions using the fused odometry path. If such regions are found, the relative poses of the car in the visits to the regions are estimated. The data from the first visit is used for creating grid maps of the region. In the next visits, the path of the vehicle is estimated using the localization algorithm presented in Section 5.2. These path estimates are used for adding loop closure constraints in the graph. The relative poses along with data from GPS, and odometry are used in a second step of optimization using the GraphSLAM algorithm. A formal description of the method used to estimate the vehicle poses is presented next. For conciseness, the notation $x_{a:b}$ is used to refer to the set $\{x_a, \dots, x_b\}, a \leq b$.

5.3.1. Fused Odometry

Consider a log *L* that, after synchronization, contains n data packages, i.e., *L* = $p_{1:n}$, where $\mathrm{p_i} = \{\mathrm{g_i}, \mathrm{o_i}, \mathrm{i_i}, \mathrm{c_i}, \mathrm{v_i}\}$ is the i-th data package, and $\mathrm{g_i}, \mathrm{o_i}, \mathrm{i_i}, \mathrm{c_i}, \mathrm{v_i}$ correspond to the data from GPS, odometry, IMU, camera, and Velodyne, respectively. Let $x_1, \dots, x_n$ be the car's poses when each of these packages were acquired. In the first step, we compute estimates of the poses, $\hat{\mathrm{x}}_{1:n}$ using data from GPS, and odometer. These estimates are obtained by solving the following optimization problem:

$$\hat{\mathrm{x}}_{1:n} = \mathrm{argmin}_{\,\mathrm{x}_{1:n}}\, \mathrm{F}(\mathrm{x}_{1:n}, \mathrm{g}_{1:n}, \mathrm{o}_{1:n}) \tag{21}$$

where the objective function, *F*, is given by:

$$\mathrm{F} = \mathrm{G}(\mathrm{x}_1, \mathrm{g}_1) + \sum_{\mathrm{t}=2}^{\mathrm{n}} \mathrm{O}(\mathrm{x}_{\mathrm{t}-1}, \mathrm{x}_\mathrm{t}, \mathrm{o}_\mathrm{t}) + \mathrm{G}(\mathrm{x}_\mathrm{t}, \mathrm{g}_\mathrm{t}), \tag{22}$$

where the term $\mathrm{G}(\mathrm{x_t}, \mathrm{g_t})$ penalizes values of $\mathrm{x_t}$ that deviates from $\mathrm{g_t}$, and, so, it enforces global consistency; the term $\mathrm{O}(\mathrm{x_{t-1}}, \mathrm{x_t}, \mathrm{o_t})$ penalizes values of $\mathrm{x_t}$ and $\mathrm{x_{t-1}}$ that are not consistent with the motion model. This term tries and enforces consecutive poses to be locally smooth.

The penalty term $\mathrm{G}(\mathrm{x_t}, \mathrm{g_t})$ is given by the Mahalanobis distance between $\mathrm{x_t}$ and $\mathrm{g_t}$:

$$G(x_t, g_t) = (x_t - g_t)^T R_t^{-1}(x_t - g_t), \tag{23}$$

where R_t is the covariance matrix representing the error in the measurement g_t.

Let $\delta_t(o_t)$ be the transformation matrix obtained by applying the odometry data o_t into the motion model. The penalty term $O(x_{t-1}, x_t, o_t)$ is given by the Mahalanobis distance between x_t and the pose obtained by adding (in the sense of pose composition) δ_t to x_{t-1}:

$$O(x_{t-1}, x_t, o_t) = \left(x_t - x_{t-1} + \delta_t(o_t)\right)^T Q_t^{-1}(x_t - x_{t-1} + \delta_t(o_t)) \tag{24}$$

where Q_t is the covariance matrix associated with the motion estimate δ_t.

The graph representing the optimization problem presented in Equation 21 and Equation 22 is illustrated in Figure 24. Nodes represent the fused odometry poses, while the edges represent the sensor measurements and their covariances (omitted in the figure). The blue edges represent constraints induced by GPS measurements, while red edges represent constraints between poses induced by odometry measurements and the motion model. The illustration employs the graph notation used by the g2o framework [91] which is different from the traditional graphical model representation.

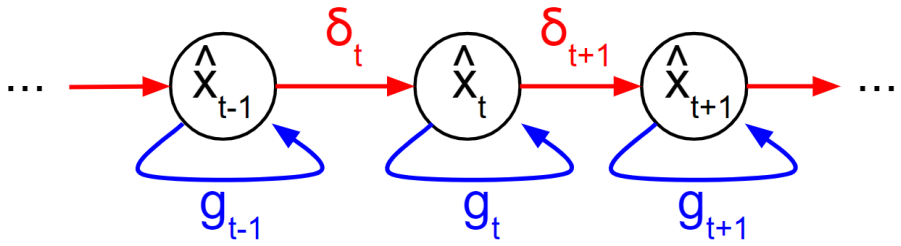


Figure 24. Graph representing the optimization problem described by Equation 18 and Equation 19. Nodes represent the estimated fused odometry poses, while the edges represent sensor measurements and their covariances (omitted in the figure). The blue edges represent GPS data, while the red edges represent the motion constraints induced by the odometry data and the motion model.

5.3.2. Loop Closures Detection and Displacement Estimation

Due to imprecisions in GPS and odometry measurements, the poses' values obtained in the first step of optimization are imperfect. Consider a situation in which the self-driving car moves over the same region more than once. Because of error in the poses, the projections of sensor data from consecutive visits to the same region into the grid map do not necessary match. This inconsistency results in a "ghosting effect" observed as a drift, duplication, or blurring of objects in the map. In the literature, this issue is known as the loop closure problem [23, 24, 25]. Figure 25 presents reflectivity grid maps before and after considering loop closures for estimating the poses. It can be observed that the "ghosting effects" are no longer observable after considering loop closures.

(a) **(b)**

Figure 25: (a) Example of ghosting effect observed when revisiting a region and using fused odometry poses for projecting sensor data into the grid map. (b) The same region after considering loop closures in the optimization process.

To account for the loop closure problem, we run a second step of optimization adding constraints to penalize poses' values whose sensor projections do not match. The process of adding the loop closure constraints into GraphSLAM's graph consists itself of two steps: the detection of loop closures, and the estimation of their displacements. The loop closure detection technique is simple yet effective. Each

pose $\hat{\mathrm{x}}_{\mathrm{t}}$ is classified as being part of a loop closure or not. To do so, we evaluate if there are poses in $\hat{\mathrm{x}}_{1:\mathrm{t}-1}$ (the set of previous poses) such that their Euclidean distance (considering only the x and y coordinates) to $\hat{\mathrm{x}}_{\mathrm{t}}$ is smaller than a threshold τ_{d} and their distance in time (difference of timestamps) to $\hat{\mathrm{x}}_{\mathrm{t}}$ is bigger than a second threshold τ_{t}. If there are poses that satisfy both conditions, $\hat{\mathrm{x}}_{\mathrm{t}}$ is classified as being part of a loop closure. The second condition is important to prevent the classification of consecutive poses as loop closures. If there are not such poses, we assume that the self-driving car is visiting the region for the first time. For every pose classified as belonging to a loop closure, we also store the nearest pose from it in $\hat{\mathrm{x}}_{1:\mathrm{t}-1}$.

The technique for estimating the displacements in loop closures operates as follows. Reflectivity and occupancy grid maps are created the first time the self-driving car visits a region. For every pose that is not a loop closure, the respective data package is used to update the maps. On the other hand, when a loop closure is detected, the maps are no longer updated and, instead, the localization technique is used for estimating the car pose in the map. Define as:

- $\tilde{x}_t$ the pose estimated by the localization technique for some fused odometry pose $\hat{\mathrm{x}}_{\mathrm{t}}$ classified as a loop closure;
- $\hat{\mathrm{x}}_{\mathrm{s}}$ the nearest pose to $\hat{\mathrm{x}}_{\mathrm{t}}$ from $\hat{\mathrm{x}}_{1:\mathrm{t}-1}$;
- $\tilde{x}_t^s$ the value of $\tilde{x}_t$ in $\hat{\mathrm{x}}_{\mathrm{s}}$'s coordinate system. This value represents the relative pose between $\hat{\mathrm{x}}_{\mathrm{s}}$ and $\tilde{x}_t$.

Then, the final estimates for the poses' values, $x_{1:n}^*$, are obtained by solving the following optimization problem:

$$x_{1:n}^* = \operatorname{argmin}_{\mathrm{x}_{1:\mathrm{n}}} \mathrm{J}(\mathrm{x}_{1:\mathrm{n}}, \mathrm{p}_{1:\mathrm{n}}) \tag{25}$$

The objective function, J, is given by:

$$\mathrm{J} = \mathrm{F}(\mathrm{x}_{1:\mathrm{n}}, \mathrm{g}_{1:\mathrm{n}}, \mathrm{o}_{1:\mathrm{n}}) + \sum_{\mathrm{t}=1}^{\mathrm{n}} \mathrm{L}(\mathrm{x}_{\mathrm{t}}, \mathrm{x}_{\mathrm{s}}, \tilde{x}_t^s) \tag{26}$$

where F is the objective function of the first step of optimization and, for each t, the term $L(x_t, x_s, \tilde{x}_t^s)$ is a penalty for values of x_t and x_s whose relative pose deviates from $\tilde{x}_t^s$ or zero if x_t has not been classified as a loop closure. The penalty is similar to the odometry penalties:

$$L(x_t, x_s, \tilde{x}_t^s) = (x_t - x_s + \tilde{x}_t^s)^T S_t^{-1} (x_t - x_s + \tilde{x}_t^s), \quad (27)$$

where S_t is the predefined covariance matrix associated with the estimate computed by the localization technique. The graph representing the optimization problem described by Equation 22 and Equation 23 is illustrated in Figure 26. The overall structure of the graph is similar to the one presented in Figure 24. Green edges represent constraints induced by loop closures.

In preliminary experiments, we also tried and considered semantic and colour grid maps in the process of handling loop closures. However, they were not accurate enough and adding constraints obtained using them into the graph caused divergence. It is worth noting, however, that the proposed method is not limited to occupancy and reflectivity grid maps. Any kind of map can be used for handling loop closures depending only of the capacity of localizing in it with sufficient accuracy.

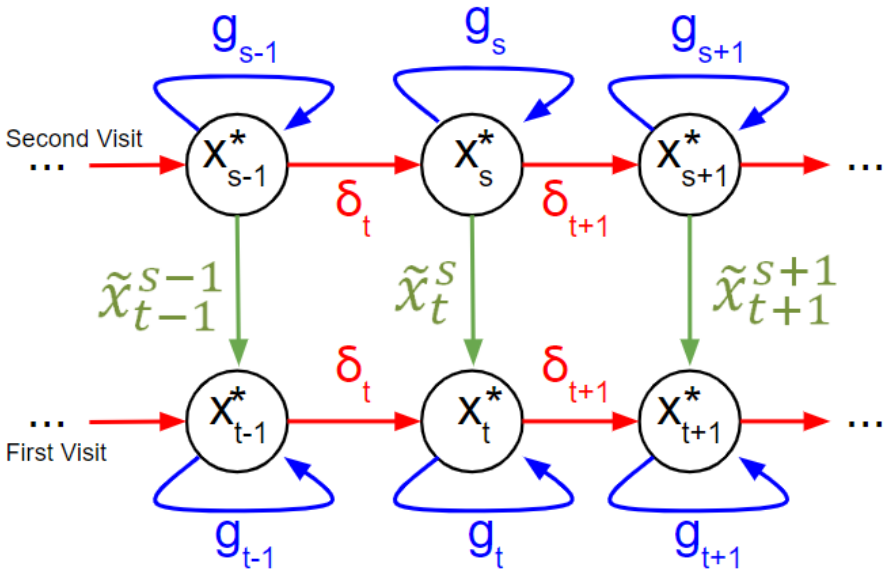


Figure 26. Graph representing the optimization problem described by Equation 22 and Equation 23. Loop closure constraints are represented in green.

The poses' values obtained in both steps of optimization are maximum likelihood estimates. Minimizing the F objective function is equivalent to maximizing the log likelihood of $\mathrm{g}_{1:\mathrm{n}}$, and $\delta_{1:\mathrm{n}}$ given the poses, assuming that these measurements are independent and follow Gaussian distributions. Likewise, minimizing the J objective function is equivalent to maximizing the log likelihood of $\mathrm{g}_{1:\mathrm{n}}$, and $\delta_{1:\mathrm{n}}$, and $\tilde{x}_{1:\mathrm{n}}$ under the same assumptions.

In determined situations such as in slow traffic, busy cross walks, and due to traffic lights, the self-driving car stays in the same place for long periods. In these situations, the poses in which the car is stopped will have more "importance" in the optimization process than the others since several sensors' measurements are captured in that single location. An effect of this fact is that the optimizer may accept to produce imprecise values for the remaining poses for sake of maintaining the poses in which the car is stopped correct. We found that poses estimates are obtained (and so better maps) if we do not consider the poses in which the car speed is smaller than a threshold in the optimization process. Moreover, if the distance between consecutive GPS measurements is bigger than a threshold, we assume that a jump happened and we discard the GPS measurement. In this case, only odometry and loop closure edges are added to the node.

The method for estimating the loop closures displacements is based on the localization technique which is susceptible to momentary failures. These failures can lead to divergence in the pose estimation process. To try and minimize this issue, the poses estimated using the localization technique are compared with the respective fused odometry poses. If the distances between them or the difference between their orientations are higher than predefined thresholds, the loop closure constraint is discarded.

6. Experiments

This chapter describes the experimental methodology for evaluating the proposed techniques and the obtained results. Qualitative analyses of the mapping technique and quantitative analyses of the localization technique are presented. Experiments were performed in several large-scale environments, in diverse conditions of operation. We compared the accuracy achieved by the localization technique when using occupancy grid maps, reflectivity grid maps, semantic grid maps, and colour grid maps.

6.1. Implementation

The mapping and localization techniques were implemented as independent modules of the carmen_lcad framework[2], a branch of the Carnegie Mellon Navigation framework [92] that is maintained and continually extended by the Laboratório de Computação de Alto Desempenho (LCAD) from the Universidade Federal do Espírito Santo (UFES). The IARA self-driving car described in Chapter 3 was used for collecting the datasets.

The source code was developed in C/C++ to try and maximize the computational efficiency of the techniques. The g2o framework [91] was employed for implementing the GraphSLAM algorithm described in Section 5.3. This framework provides high level tools for modeling probabilistic methods that can be represented as graphs. It also provides efficient optimization methods for inference.

[2] https://github.com/LCAD-UFES/carmen_lcad

6.2. Datasets

Experiments are performed in several environments in diverse and real conditions of operation. A dataset is created for each environment. In this context, a dataset comprises a set of logs in which one is used for mapping, and the remaining ones for testing. The environments and the logs that compose their associated datasets are presented below. The mapping logs, their dates, and characteristics are listed in Table 2. Likewise, the test logs, their dates, and characteristics are listed in Table 3. The mapping logs were recorded when the traffic of vehicles and pedestrians was low (e.g. holidays and Sundays).

Logs are named using a tag representing the environment and a suffix that represents their main features. Logs with suffix “M” are used for mapping. Logs with suffix “D” have a high number (from tens to hundreds) of distractors (e.g. vehicles, pedestrians, and environmental changes). Logs with suffix “N” were recorded at night (after 7 p.m.). Logs with suffix “T” are test logs with conditions similar to those present in the logs used for mapping. The datasets created and their associated logs are the following:

Table 2. Logs used for mapping and their characteristics.

Dataset	Date	Characteristics
UFES_M	2018/09/07	Recorded in a holiday.
PLOT_M	+2018/12/08	Recorded in a Sunday with an empty parking lot. Strong GPS noise (probably due to trees).
URBAN_M	2018/12/08	Recorded in a holiday. Small number of dynamic objects in the streets. Presence of a small tunnel without GPS signal.
AIRPRT_M	2017/07/26	Recorded at sunrise. Featureless surroundings. Presence of lane marks.
AV_M	2018/11/16	Several dynamic objects.

Table 3. Logs used for testing the localization system and their characteristics.

Dataset	Date	Characteristics
UFES_D	2018/12/06	Several parked cars. Several dynamic objects (cars and pedestrians). Illumination different from the mapping log. Smooth curves.
UFES_N	2018/11/30	Small number of parked cars and dynamic objects. Recorded at night. Smooth curves. Different initial position.
PLOT_D	2018/12/07	Several parked cars. Several dynamic objects (cars and pedestrians). Challenging U-turns. Illumination different from the mapping log.
PLOT_N	2019/07/03	Small number of parked cars and dynamic objects. Recorded at night.
URBAN_D	2018/12/07	Several parked cars. Several dynamic object (hush hour). Challenging curves. Transitions between diverse scenarios Illumination different from the mapping log. Presence of a small tunnel.
URBAN_N	2019/07/03	Significant number of dynamic objects. Transitions between diverse scenarios Challenging curves. Recorded at night. Presence of a small tunnel.
AIRPRT_T	2017/07/26	Recorded at sunrise. Different initial position. Featureless surroundings. Presence of lane marks.
AV_T	2018/11/16	Several dynamic objects. Different lane than the mapping log.

UFES. Dataset created in the ring road of the Universidade Federal do Espírito Santo (UFES). The dataset consists of three logs, one for mapping (UFES_M) and two for testing (UFES_D and UFES_N). The path performed by the vehicle in the environment can be visualized in Figure 27. Figure 28 illustrates the difference between the images captured by the camera in the each of the logs along with their semantic segmentations. Note the higher number of parked cars in the test logs and the presence of light blobs and motion blur in the log recorded at night.

PARKING LOT. Dataset recorded in a parking lot at UFES. The parking lot is not part of the ring road. As before, the dataset consists of three logs, one for mapping (PLOT_M) and two for testing (PLOT_D and PLOT_N). The mapping log was created when the parking lot was empty. Both of the test logs were recorded in moments in which tens of cars were parked. This dataset allows evaluating the robustness of the localization technique in face of tight U-turns and of several objects that were not present during mapping. The path of the vehicle in the mapping log is illustrated in Figure 29.

Figure 27. Satellite view of the environment in the UFES dataset with the path of the self-driving car highlighted in blue.

URBAN. Dataset created in the neighbourhood of Jardim da Penha, in the City of Vitória, Brazil. Similarly to the previous two logs, the dataset is composed of a mapping log (URBAN_M), a daytime test log (URBAN_D) and a night time test log (URBAN_N). The environment contains diverse urban scenarios such as tree-lined streets, tunnels, suburbs, loops around plazas, and an avenue with intense traffic. The path of the vehicle can be visualized in Figure 32.

AVENUE. Dataset created in part of the Dante Michelini Avenue, one of the main avenues of the City of Vitória, ES, Brazil. This environment was chosen to evaluate the localization system in traffic scenarios. Two logs were created, one for mapping (AV_M), and one for testing the localization technique (AV_T). The test log was recorded in a different lane than the one used for mapping. The avenue is a long corridor-like environment. The path of the vehicle can be visualized in Figure 30. Unfortunately, we did not managed to produce a log of this environment during nighttime.

AIRPORT. Dataset created in the runway of an airport located at Aeroclube[3] do Espírito Santo. Because the access to the airport is restricted, the mapping and test logs (AIRPRT_M and AIRPRT_T, respectively) were created in the same day at sunrise (about 5 a.m.), one after another. This dataset allows evaluating the capacity of the localization technique of operating in adverse conditions of illumination (not as harsh as the logs recorded at night though) and in flat terrains without apparent features in the surroundings. The path of the vehicle can be visualized in Figure 31.

[3] https://www.aeroclube-es.com.br/

Figure 28. Variation in the illumination conditions and presence of distractors in the UFES dataset. The three images correspond to the same place. The top image is part of the UFES_M log, and middle image is part of the UFES_D log, and the bottom image is part of the UFES_N log. Note the higher number of parked cars in the test logs and the presence of light blobs and motion blur in the log recorded at night.

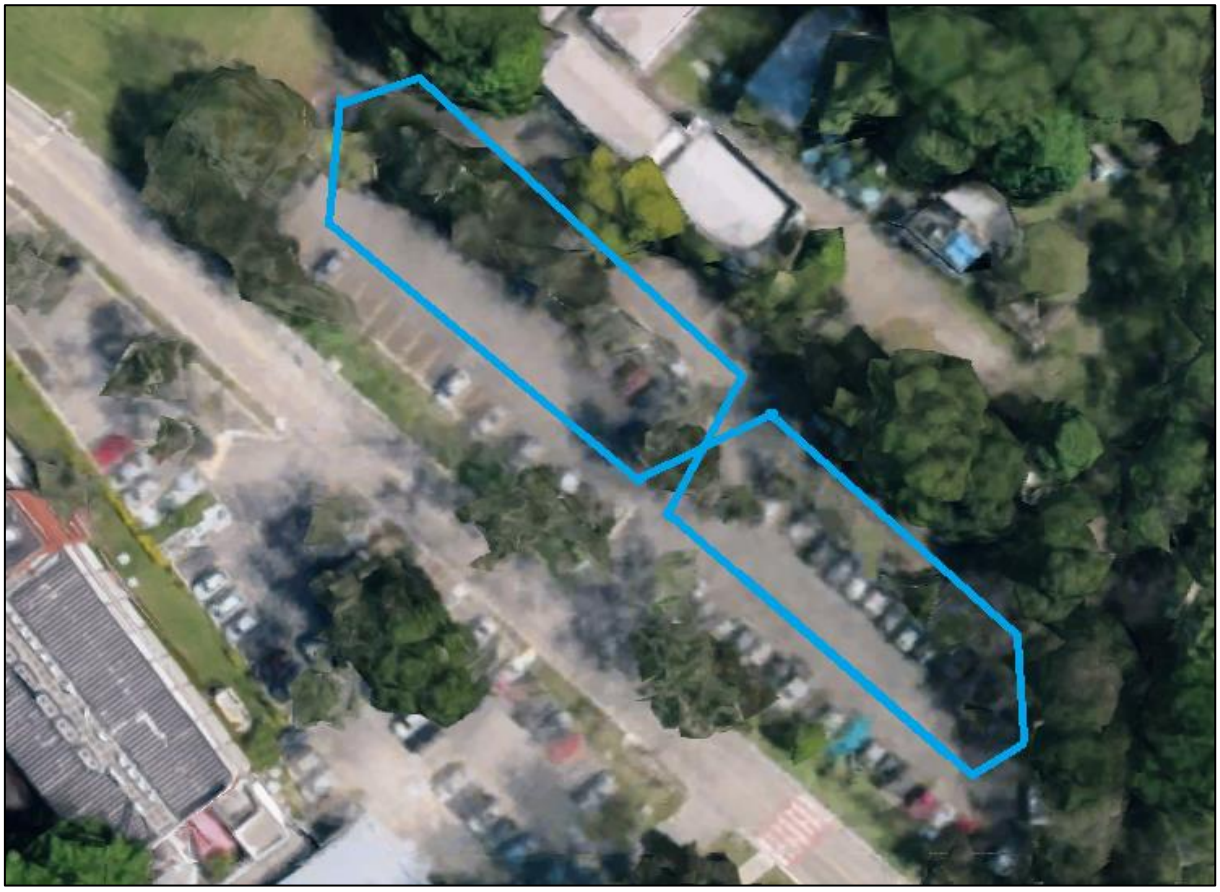


Figure 29. Satellite view of the environment in the PARKING LOT dataset with the path of the self-driving car highlighted in blue.

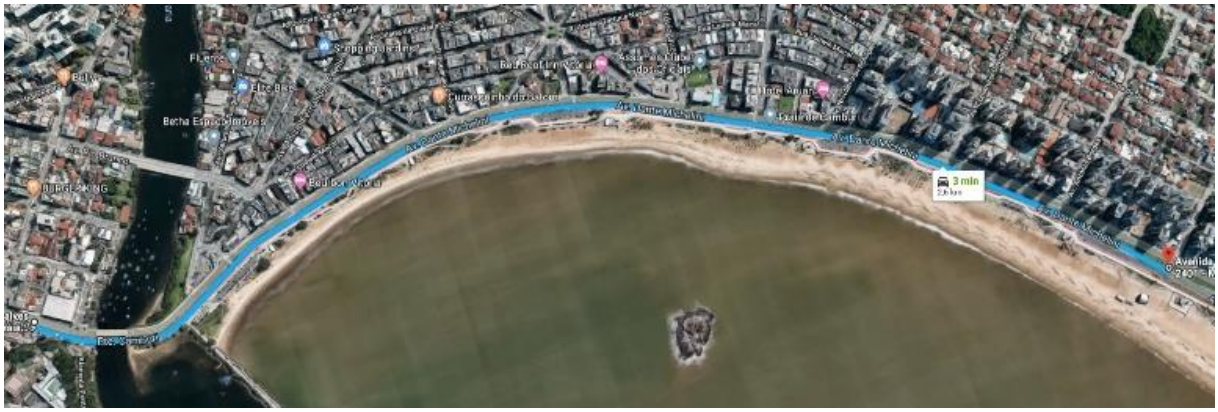


Figure 30. Satellite view of the environment in the AVENUE dataset with the path of the self-driving car highlighted in blue.

Figure 31. Satellite view of the environment in the AIPORT dataset with the path of the self-driving car in the log used for mapping highlighted in blue.

Figure 32. Satellite view of the environment in the URBAN dataset with the path of the self-driving car in the log used for mapping highlighted in blue.

6.3. Experiments

The experiments performed to evaluate the proposed techniques are the following:

Map Building. This experiment evaluates the capacity of the mapping technique of producing grid maps with consistent loop closures. For each environment, the mapping technique is employed for building occupancy, reflectivity, colour, and semantic grid maps. A qualitative analysis of the maps is presented.

Localization Accuracy at Daytime. This experiment evaluates the accuracy of the localization technique using different types of grid maps during daytime. The experiment aims at evaluating the sensitivity of the method to different conditions of illumination (e.g., shadows and different intensities of light) and its robustness in face of objects that were not present during mapping. We do not compare our method with previous methods from the literature. Performing a fair comparison of localization techniques is challenging because the localization accuracy depends of several aspects such as the sensor set used by the vehicle, the environment, and the conditions (weather, illumination, etc.) in which experiments are performed.

Localization Accuracy at Night and Shadows. This experiment evaluates the accuracy of the localization technique at night and in conditions of illumination that are significantly different from the ones present in the mapping logs.

This previous pair of experiments evaluates the hypotheses that: (i) localization with occupancy and reflectivity grid maps are nearly invariant to different conditions of illumination and perform as well at night as in daylight; (ii) given a robust method for extracting semantic information from sensor data, semantic grid maps can be successfully used for localization in daylight and at night; and (iii) using the entropy correlation coefficient for computing the particle weights, colour grid maps can be used for localization in different conditions of illumination.

Localization Accuracy in Flat Terrains with Featureless Surroundings. This experiment evaluates the capacity of the localization technique of operating in the challenging environment of an airport runway. Some highways are similar to an

airport runway in the sense that they consist of a flat road with eventual lane marks and with a surrounding area that has not perceptible features for localization.

6.4. Metrics

We use the same criteria used in previous works [23, 24, 25] for evaluating the localization technique. In particular, the root mean squared error (RMSE) is employed for measuring the accuracy of the technique. Assuming that $x_{1:t}$ are the localization estimates associated with t data packages, and $g_{1:t}$ are the ground-truth poses, the RMSE is given by:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{t}(x_i^x - g_i^x)^2 + \left(x_i^y - g_i^y\right)^2}{t}} \tag{28}$$

where the superscripted x and y represent the respective coordinates of the poses.

Bresson et al. argue that for operating safely, the error in the localization estimates should never be bigger than 20cm [17]. Inspired by this observation, we also present for each test log the percentage of data packages in which the localization error is smaller than a given threshold. The chosen threshold values are 0.5m, 1.0m, and 2.0m.

6.5. Parameters

The parameters used in the mapping technique were empirically adjusted to maximize the quality of the maps. The resolution of the maps is set to 20cm, and the tile size is set to 70m. The parameters of the localization technique are presented in Table 4. Two sets of parameters are evaluated, one favours stability in the estimates produced by the particle filter (STABLE), and the second that favours recoverability / diversity in the particles values (DIVERSE). The different behaviours of the particle filter are chosen by using different amounts of noise to update the particles values during the prediction step. The amount of noise used in the filter is controlled by the standard deviations (std) of the Normal distributions used to sample noise. By adding

more noise into the particles values, the diversity of the particles is increased which can increase the capacity of the filter of recovering from incorrect estimates. On the other hand, high levels of noise can cause large variations in the pose estimates. These variations can cause oscillations and abrupt movements during autonomous operation as a result of the decision making system trying to adjust the vehicle trajectory given the new pose estimate.

6.6. Ground truth Generation

Producing a ground truth for evaluating the localization technique is quite challenging. The comparison of localization estimates with GPS, for example, leads to a poor evaluation. Although globally consistent, GPS data can be subject to significant amounts of noise, and the GPS measurements are not necessarily consistent with the map. Maps are built using poses that result from an optimization algorithm which may drift locally from the GPS path. These local drifts can make the maps inconsistent with the real world. Since the localization is computed in relation to the map, the localization estimate may be inconsistent with the GPS, while being consistent with the map.

The approach used in our work for producing the ground truth is similar to the ones employed in [23] [24][25]. First, the mapping technique presented in Section 5.3 is employed for optimizing the poses of the mapping log and building the maps. From this point onwards, the poses of the mapping log are fixed and do not change.

In order to compute the true poses of the test log, we pretend that the test log is a loop closure of the mapping log, and the tools used for handling loop closures are used to optimize the poses of the test log. The localization method presented in Section 5.2 is used for estimating the localization of the vehicle in the map (built using the mapping log). As before, reflectivity and occupancy grid maps are used for localization. The poses estimates obtained using these maps along with data from GPS, odometry, and loop closures constraints are used as input for the GraphSLAM algorithm and the resultant poses are assumed as the ground truth. By using the sensors' data together with the localization estimates, we can compensate local

errors in the localization. The localization estimates are introduced in the graph as global edges similarly to GPS measurements.

Table 4. Parameters used for evaluating the localization technique.

Parameter	STABLE	DIVERSE
Std of the position in the initial guess of the localization	2.5m	2.5 m
Std of the orientation in the initial guess of the localization	20 deg	20 deg
Std of reflectivity measurements	3	3
Std of colour measurements	3	3
Number of particles	200	200
Std in velocity measurements	0.2 m/s	0.2 m/s
Std in steering wheel angle measurements	0.5 deg	0.5 deg
Std in XY direction in ϵ_p	0.01 m	0.1 m
Std in the orientation in ϵ_p	0.1 deg	0.5 deg
Outlier rejection rate	0.7	0.7

Using the same notation used in Chapter 5.3, the ground truth poses can be obtained by solving the following optimization problem:

$$x^*_{1:n} = \operatorname{argmin}_{\mathrm{x_{1:n}}} \mathrm{Q}(\mathrm{x_{1:n}}, \mathrm{p_{1:n}}, \mathrm{l_{1:n}}) \tag{29}$$

where $x^*_{1:n}$ are the ground truth poses, $\mathrm{p_{1:n}}$ are the data packages from the test log, and $\mathrm{l_{1:n}}$ are the localization estimates in relation to the map. The objective function Q is given by:

$$\mathrm{Q} = \mathrm{J}(\mathrm{x_{1:n}}, \mathrm{p_{1:n}}) + \sum_{\mathrm{t=1}}^{\mathrm{n}} \mathrm{G}(\mathrm{x_t}, l_\mathrm{t}) \tag{30}$$

where J is the objective function of the second step of optimization from Chapter 5.3 and G is the function for global comparison (the same used for GPS measurements), but using the covariance matrix for the localization estimates instead of the covariance matrix associated with GPS measurements. Since the ground-truth results from an optimization process, it is not perfect. However, a qualitative analysis shows that it is significantly more accurate than the GPS measurements.

6.7. Results

The grid maps created using the logs UFES_M, PLOT_M, URBAN_M, AV_M, and AIRPRT_M are presented, respectively, in Figure 33, Figure 34, Figure 35, Figure 36, and Figure 37. The proposed mapping technique was successfully employed for creating grid maps of all logs. As the reader can observe, the loop closures are consistent and ghosting effects are not observable.

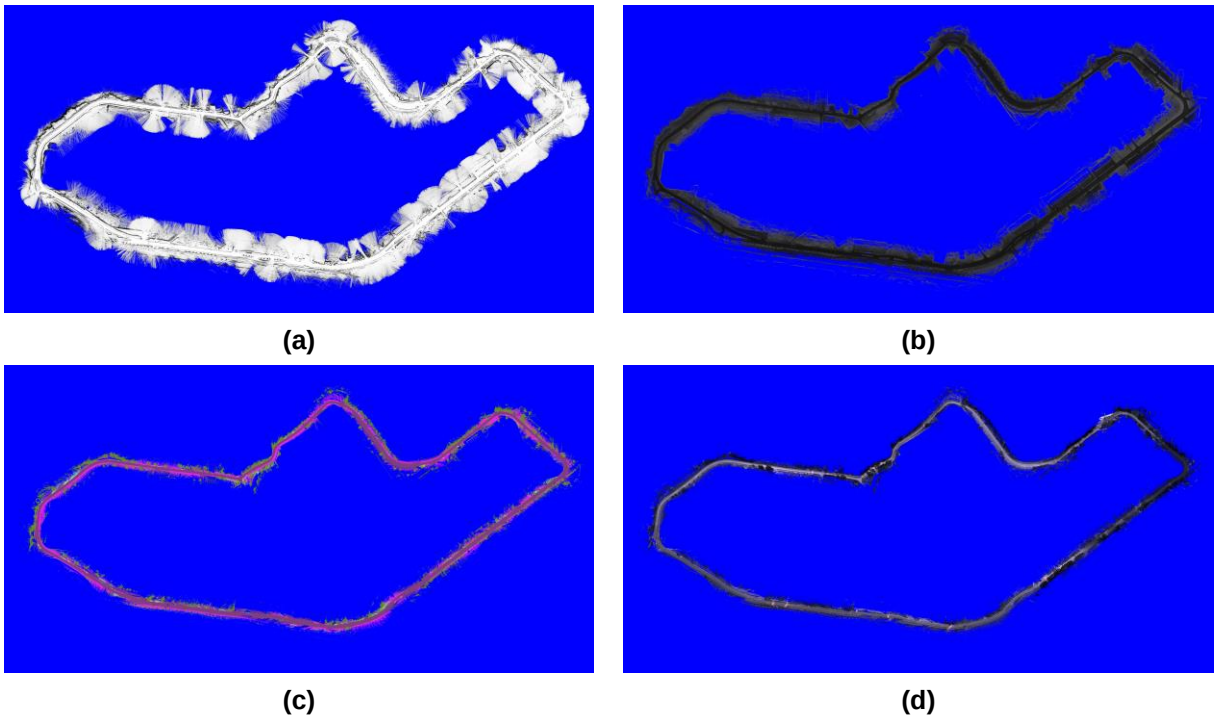


Figure 33. Grid maps created using the UFES_M log. (a) Occupancy grid map. (b) Reflectivity grid map. (c) Semantic grid map. (d) Visual grid map.

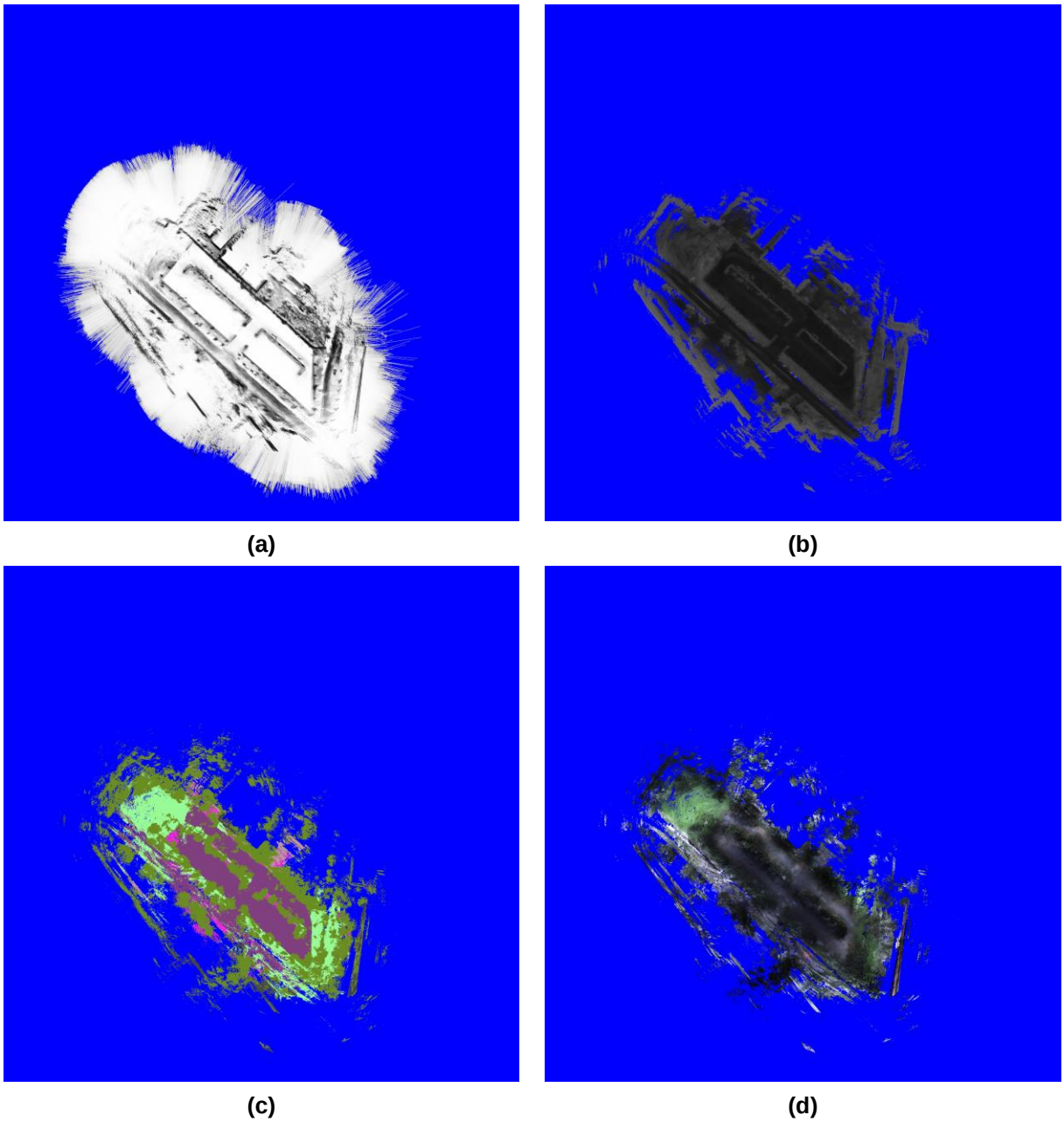


Figure 34. Grid maps created using the PLOT_M log. (a) Occupancy grid map. (b) Reflectivity grid map. (c) Semantic grid map. (d) Visual grid map.

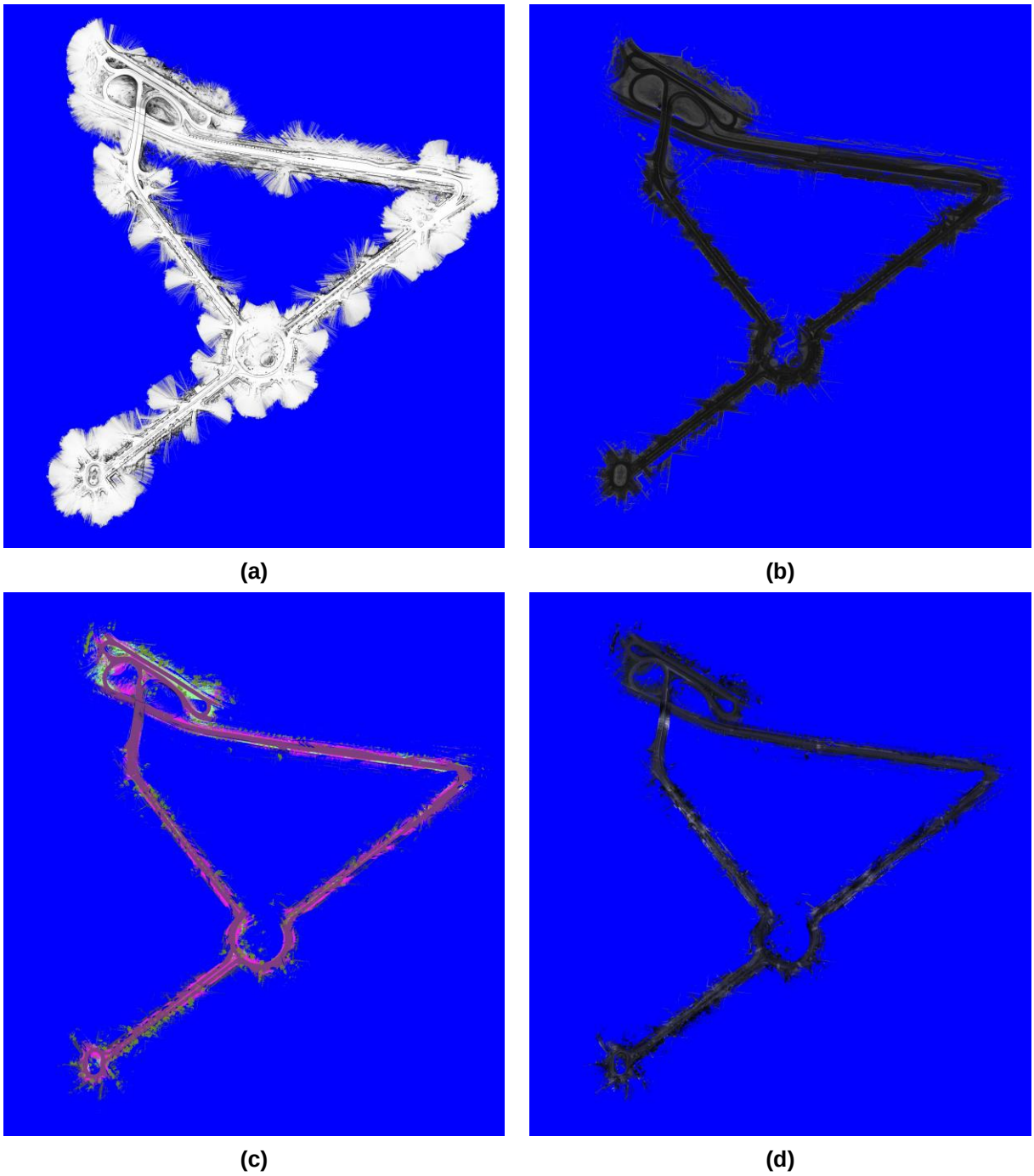


Figure 35. Grid maps created using the URBAN_M log. (a) Occupancy grid map. (b) Reflectivity grid map. (c) Semantic grid map. (d) Visual grid map.

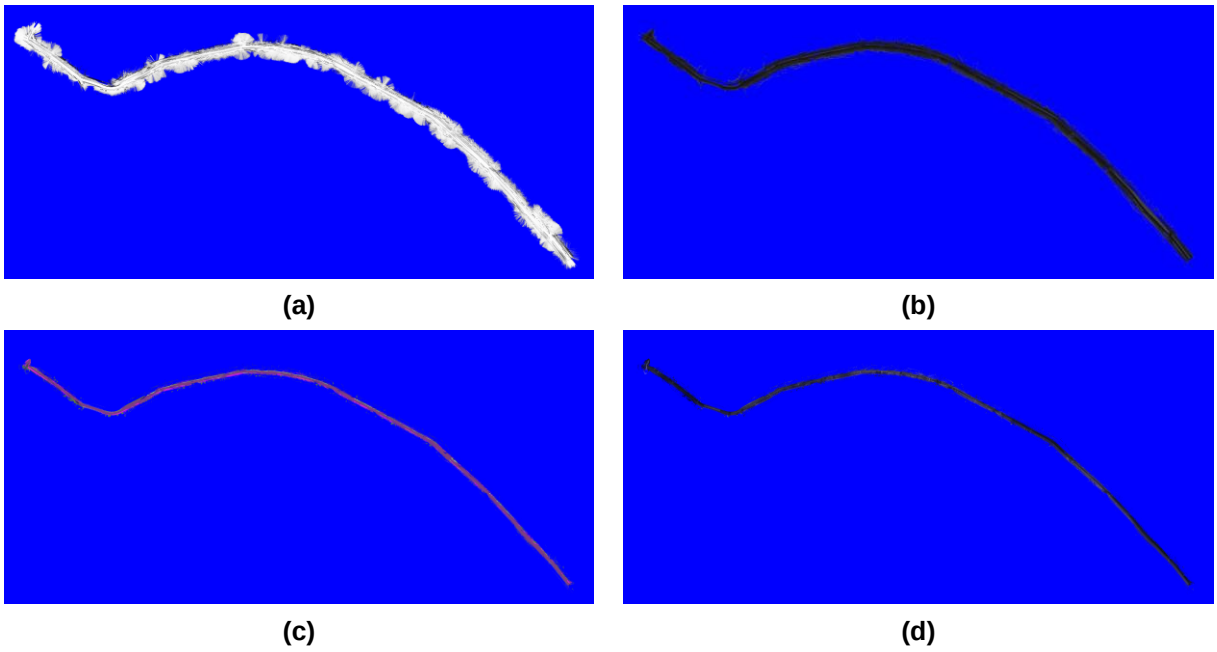


Figure 36. Grid maps created using the AV_M log. (a) Occupancy grid map. (b) Reflectivity grid map. (c) Semantic grid map. (d) Visual grid map.

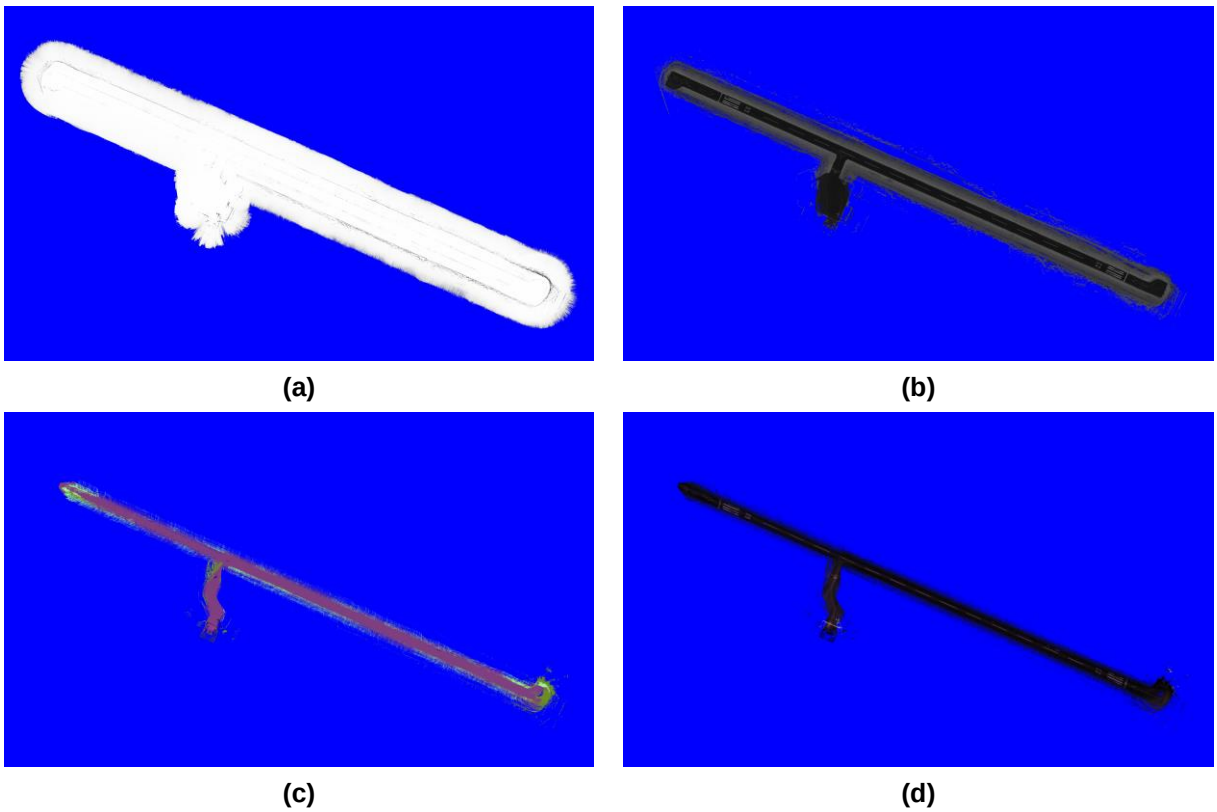


Figure 37. Grid maps created using the AIRPRT_M log. (a) Occupancy grid map. (b) Reflectivity grid map. (c) Semantic grid map. (d) Visual grid map.

Table 5 presents the metrics achieved by the localization technique in the logs UFES_D, PLOT_D, URBAN_D, and AV_T with the DIVERSE configuration. These logs were recorded during daylight. The methods based solely on LiDARs achieved the highest performance in all tests. For two out of four logs, the localization with occupancy grid maps achieved the smaller error. In the other two, the most accurate results were achieved using the reflectivity grid maps. The localization with semantic grid maps resulted in an RMSE that is bigger than the other two, but it managed to maintain the position tracking in all logs without divergence. In the UFES_D and AV_T logs, the localization with colour grid maps did not diverge as well, but it maintained a nearly constant offset from the correct pose. In the other logs it diverged from the correct path.

The results observed in the logs UFES_D and PLOT_D show that the localization with occupancy, reflectivity, and semantic grid maps is robust to distractors (e.g., parked cars and pedestrians), to changes in the environment due to time passing (note the difference of months between the mapping and test logs), to different conditions of illumination, and to the complex U-turns present in the PLOT_D log.

In the URBAN_D logs, the localization with occupancy grid maps diverged from the true path. By analysing the method, we observed that the divergence happened in a situation of intense traffic in which the self-driving car was surrounded by other vehicles. Examples of such situations are presented in Figure 38.

Even using a sophisticated measure for computing the particles weights when using colour grid maps, the method did not managed to achieve a performance as good as with the other maps, particularly in face of shadows that are not uniformly distributed over the environment (see top image in Figure 28).

Table 5. Results of the localization technique for the DIVERSE configuration.

UFES_D					
Mode	**RMSE (m)**	**STD (m)**	**% < 2m**	**% < 1m**	**% < 0.5m**
Occupancy	**0.18**	0.02	100.00	100.00	94.58
Reflectivity	0.19	0.02	100.00	100.00	93.83
Semantic	0.30	0.02	100.00	99.49	92.95
Colour	1.85	2.64	89.03	83.39	68.27
PLOT_D					
Occupancy	0.19	0.02	100.00	100.00	95.93
Reflectivity	**0.17**	0.01	100.00	100.00	98.99
Semantic	0.28	0.03	100.00	100.00	88.12
Colour	20.94	216.08	22.62	18.54	12.60
URBAN_D					
Occupancy	83.49	5849.75	68.31	67.84	61.07
Reflectivity	**0.52**	0.18	98.34	93.72	82.32
Semantic	0.60	0.19	97.61	93.28	78.00
Colour	17.46	246.27	60.80	59.71	55.08
AV_T					
Occupancy	**0.21**	0.03	100.00	99.11	96.75
Reflectivity	0.23	0.03	99.98	99.06	95.66
Semantic	1.92	1.22	70.51	43.83	10.49
Colour	3.32	6.01	61.54	51.08	14.51

Figure 38. Examples of intense traffic situations in the URBAN dataset.

Table 6 presents the localization results for the same logs, but using the STABLE configuration. The errors were on average bigger than when using the DIVERSE configuration. This result supports the hypothesis that adding more noise during the prediction step increases the capacity of the method of correcting momentary failures. It is worth noting, however, that the localization with occupancy grid maps did not diverged in the URBAN_D logs with this configuration. Recall that the divergence happened when the self-driving car was surrounded by other cars. By adding less noise into the particles, they spread more slowly which allowed the surrounding cars to move before the localization drifted to an unrecoverable pose.

Table 7 and Table 8 present the metrics achieved by the localization technique in the logs UFES_N, PLOT_N, and URBAN_N with the DIVERSE and STABLE configurations, respectively. These logs were recorded at night. The results are consistent with the ones observed in the daylight experiments. The localization using occupancy and reflectivity grid maps resulted in the smaller error, with the method based on occupancy grid maps being the more precise in five out of six logs.

Table 6. Results of the localization technique for the STABLE configuration.

UFES_D					
Mode	**RMSE (m)**	**STD (m)**	**% < 2m**	**% < 1m**	**% < 0.5m**
Occupancy	0.59	0.23	97.16	92.35	80.10
Reflectivity	**0.55**	0.19	97.85	93.05	86.18
Semantic	0.61	0.18	98.86	89.10	71.67
Colour	11.28	44.17	18.77	16.14	14.17
PLOT_D					
Occupancy	0.22	0.02	100.00	100.00	93.81
Reflectivity	**0.17**	0.01	100.00	100.00	99.33
Semantic	0.35	0.05	100.00	98.50	84.52
Colour	9.61	9.11	0.54	0.30	0.24
URBAN_D					
Occupancy	**0.50**	0.18	97.86	95.03	88.04
Reflectivity	0.83	0.52	95.55	90.77	81.96
Semantic	0.55	0.17	97.78	94.76	81.89
Colour	0.60	0.20	97.88	92.16	78.48
AV_T					
Occupancy	0.64	0.30	94.56	91.56	83.83
Reflectivity	**0.61**	0.25	96.05	92.24	81.69
Semantic	2.14	1.89	66.42	41.70	24.33
Colour	26.76	112.47	1.57	0.03	0.02

Even the localization with sematic grid maps relying on images captured by a camera, the method also kept the position tracking. This capacity is mostly to the high-quality semantic segmentations produced by the Deeplabv3+ deep neural network even in challenging conditions of illumination (see Figure 39 and Figure 40). This result also demonstrates the potential of using semantic information for localization. Given a robust method for extracting semantic information from sensors data, the semantic information is invariant to different illumination and weather conditions.

Table 7. Results of the localization technique in night conditions for the DIVERSE configuration.

UFES_N					
Mode	**RMSE (m)**	**STD (m)**	**% < 2m**	**% < 1m**	**% < 0.5m**
Occupancy	**0.19**	0.02	100.00	99.98	96.17
Reflectivity	0.20	0.02	99.98	99.93	96.85
Semantic	0.51	0.10	99.44	95.20	76.73
Colour	6.05	16.75	40.30	23.27	11.40
PLOT_N					
Occupancy	0.22	0.02	100.00	100.00	95.68
Reflectivity	**0.20**	0.02	100.00	100.00	95.15
Semantic	0.63	0.17	98.32	93.34	63.48
Colour	22.85	221.43	11.48	3.96	1.02
URBAN_N					
Occupancy	**0.19**	0.02	100.00	99.82	96.59
Reflectivity	0.22	0.03	100.00	98.12	96.51
Semantic	0.60	0.14	98.54	93.00	69.22
Colour	11.79	57.33	14.14	8.46	4.61

Table 9 and Table 10 present the metrics achieved by the localization technique in the logs AIRPRT_T with the DIVERSE and STABLE configurations, respectively. These logs were recorded in the airport runway which consists of a flat area with few features. Similarly to the previous experiments, the localization based on occupancy and reflectivity grid maps achieved the higher performance, with the top performance being achieved with the STABLE configuration.

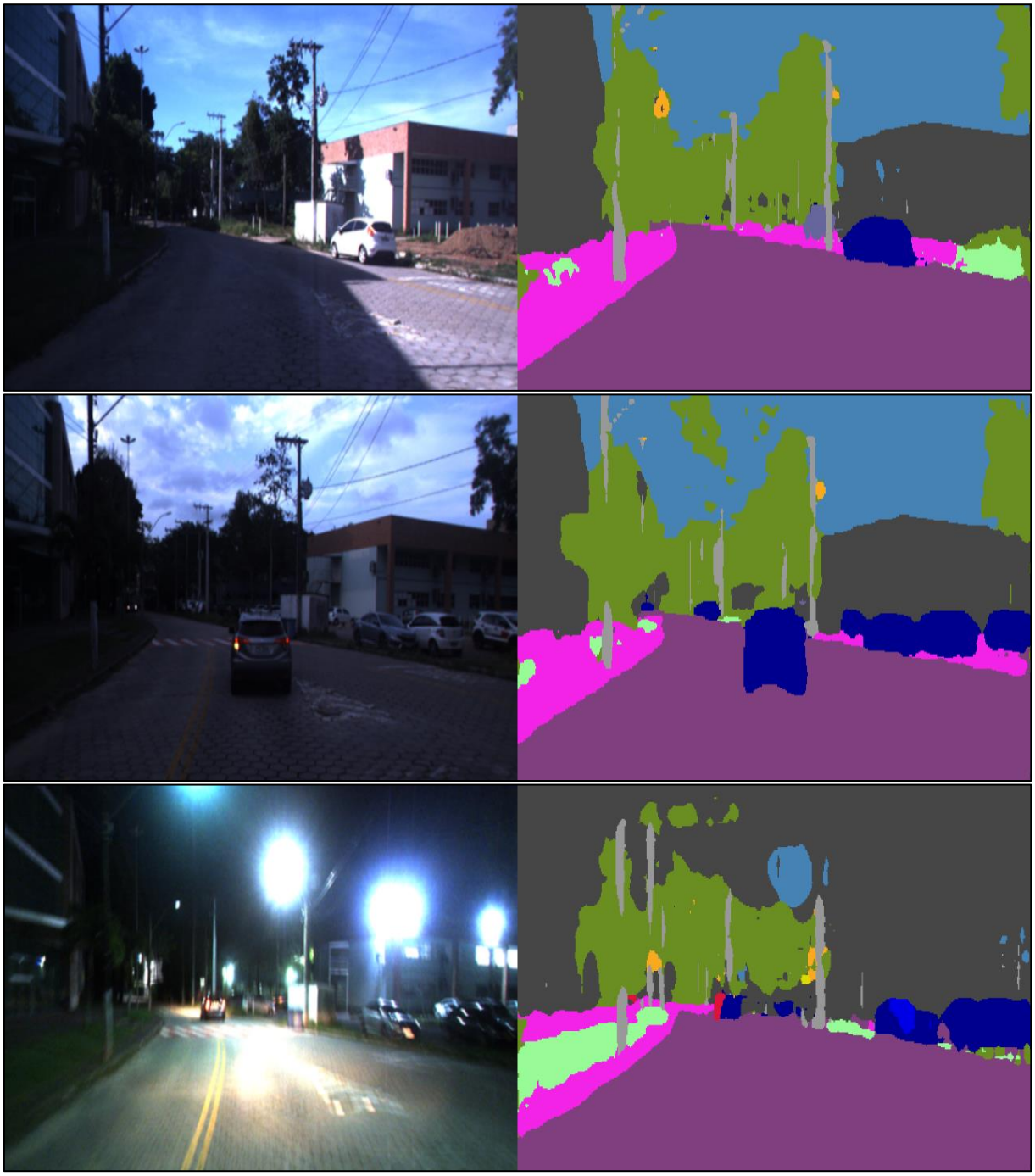


Figure 39. Good semantic segmentations produced by the deep neural network Deeplabv3+ under challenging conditions of illumination. The three images correspond to the same place, with the top image being from the UFES_M log, the second one being from the UFES_D log, and the last one being from the UFES_N log.

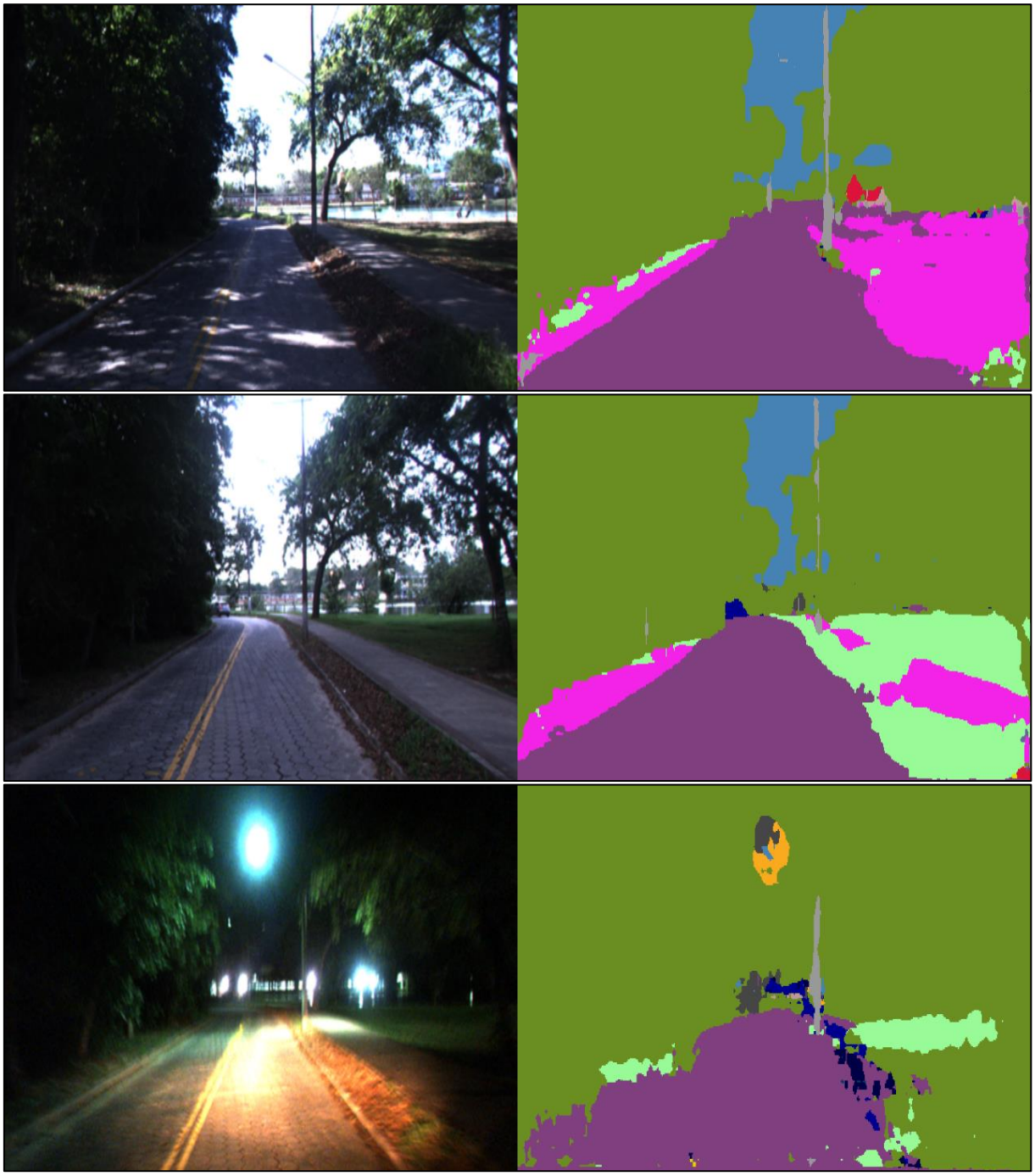


Figure 40. Another example of good semantic segmentations produced by the deep neural network Deeplabv3+ under challenging conditions of illumination. The three images correspond to the same place, with the top image being from the UFES_M log, the second one being from the UFES_D log, and the last one being from the UFES_N log.

Table 8. Results of the localization technique in night conditions for the STABLE configuration.

UFES_N					
Mode	**RMSE (m)**	**STD (m)**	**% < 2m**	**% < 1m**	**% < 0.5m**
Occupancy	**0.28**	0.05	100.00	97.74	93.90
Reflectivity	0.34	0.08	99.96	95.99	90.48
Semantic	1.86	2.54	85.13	80.48	70.87
Colour	6.71	19.69	28.24	17.10	8.45
PLOT_N					
Occupancy	**0.20**	0.02	100.00	100.00	96.15
Reflectivity	0.22	0.02	100.00	100.00	95.99
Semantic	0.34	0.03	100.00	100.00	88.44
Colour	10.85	21.84	2.94	0.67	0.04
URBAN_N					
Occupancy	**1.18**	1.16	93.21	89.77	83.51
Reflectivity	2.10	2.75	69.66	63.87	59.99
Semantic	2.28	3.92	84.68	82.66	68.07
Colour	29.44	297.68	0.78	0.08	0.05

With all the grid maps, the localization technique managed to maintain the position tracking without divergence. However, both, the localization based on visual and semantic grid maps drifted from the correct poses in the beginning of the test log, and the offset was maintained during the rest of the log.

The high accuracy of the localization in this dataset is surprising given the apparent lack of features in the environment. Observing the maps in Figure 37, however, the reader can find features that are identified in a superficial analysis. The limits of the runway and the surrounding vegetation are detected as obstacles which support the localization with occupancy grid mas. The lane marks on the road are also visible in the reflectivity grid maps which can aid the localization with them.

Table 9. Localization accuracy in the AIRPORT dataset with the DIVERSE configuration.

AIRPRT_T					
Mode	**RMSE (m)**	**STD (m)**	**% < 2m**	**% < 1m**	**% < 0.5m**
Occupancy	**0.10**	0.01	100.00	99.95	99.65
Reflectivity	0.25	0.05	100.00	97.89	97.19
Semantic	3.14	4.49	56.78	38.44	20.09
Colour	7.37	3.99	0.82	0.35	0.12

Table 10. Localization accuracy in the AIRPORT dataset with the STABLE configuration.

AIRPRT_T					
Mode	**RMSE (m)**	**STD (m)**	**% < 2m**	**% < 1m**	**% < 0.5m**
Occupancy	0.10	0.00	100.00	100.00	99.65
Reflectivity	**0.10**	0.00	100.00	99.85	99.75
Semantic	4.66	3.95	20.44	14.84	10.40
Colour	7.16	4.00	0.94	0.23	0.12

In all cases, the localization with colour grid maps led to accuracies smaller than the ones achieved with the other types of maps. Even with a robust metric being used for computing the particles weights, the non-uniformity of the illumination and the color variation in face of it may have caused a negative impact in the method. The deep neural network Deeplabv3+, on the other hand, was less sensitive to variations in the illumination. It managed to produce semantic segmentations that were sufficient for localizing in semantic grid maps without divergence. The semantic data demonstrated to be a good intermediary representation of the features of the environment. Semantic information is invariant to illumination and weather conditions, and also invariant to the sensor modality (in principle, it can be extracted from any sensors). The method for extracting semantic information from sensors can be sensitive to these factors, however. Given a robust method for extracting semantic information, this work demonstrates that localization with semantic grid maps is possible.

The hypothesis that it is possible to use the proposed techniques to build grid maps of large-scale environments was validated in all experiments. On the other hand, the hypothesis that it is possible to estimate the car localization with an average accuracy of 0.2m was invalidated. It is worth noting, however, that this accuracy was achieved in some of the tested datasets using occupancy and reflectivity grid maps. We conjecture that similar results would be achieved using semantic grid maps if LiDAR point clouds were directly segmented instead of fusing data from LiDAR and camera for obtaining semantic information.

Table 11 summarizes the experimental results as a qualitative analysis of the localization accuracy for different types of grid maps and conditions of operation. Reflectivity grid maps achieved good localization accuracy in all conditions. Similar results were achieved by occupancy grid maps, except in intense traffic conditions. In this scenario, a divergence from the true path was observed. Since the localization based on colour and semantic grid maps rely on camera data, they are sensitive to illumination conditions and presence of shadows. This effect is even more perceptible when using colour data. The localization based on colour grid maps achieved a poor performance in all conditions. We hypothesize that the main cause of the low accuracy is the variation in the illumination.

Table 11. Qualitative analysis of the localization accuracy for different grid maps and conditions.

Conditions \ Type of maps	**Occupancy**	**Reflectivity**	**Semantic**	**Colour**
Varying illumination	Good	Good	Medium	Poor
Night	Good	Good	Medium	Poor
Outliers (movable and dynamic vehicles and pedestrians)	Good	Good	Good	Poor
Tree-lined streets	Good	Good	Good	Poor
Tunnels	Good	Good	Good	Poor
Time-passing	Good	Good	Good	Poor
Airport conditions (flat terrain, lane marks)	Good	Good	Medium	Poor
Intense traffic	Poor	Good	Good	Poor
Challenging U-turns	Good	Good	Good	Poor
Different types of pavements	Good	Good	Good	Poor

7. Conclusion

This work presented novel techniques for creating grid maps of complex large-scale environments, and for estimating the localization of self-driving cars in relation to them. The proposed mapping technique allowed the creation of occupancy, reflectivity, colour, and sematic grid maps. For creating semantic grid maps, we fused data from the Velodyne with visual semantic segmentations computed by a deep neural network, the DeepLabv3+, pre-trained in the Cityscapes dataset.

The localization technique was based on a particle filter. We introduced novel methods for computing the particles' likelihoods using semantic and colour data. A new version of the GraphSLAM algorithm was developed and successfully used for estimating the states of the self-driving in a set of logs. In the method, the data from GPS and odometry are fused to produce an initial guess for the values of the states. Then, loop closure regions are detected using these values. In the first visit to the loop closure regions, reflectivity grid maps are created. In the next visits, the localization of the vehicle in relation to the map of the first visit is computed. The final estimates of the vehicle states are obtained by using data from GPS, odometry, and the restrictions due to loop closures as input for the GraphSLAM algorithm.

The proposed techniques were evaluated in different environments. The method used for handling loop closures during the mapping was successfully used for producing the ground truth for evaluating the localization technique. The experimental results showed that the localization with occupancy grid maps led to the most accurate results, followed by reflectivity grid maps, and semantic grid maps. Even using the ECC measure for computing the particles weights, the localization with colour data was significantly less precise than with the other types of maps.

Errors in the process of fusing data from Velodyne and camera can explain the inferior performance achieved with semantic and colour grid maps. Among others, the lack of synchrony between the sensors, latency in the camera, and incorrect calibration of the sensors’ relative poses are potential causes of such errors.

In future works, we intend to evaluate the localization accuracy with different types of grid maps present in the literature (e.g., height grid maps) and in different environments (e.g., highways and hills) and conditions of operation (e.g., variable weather conditions). The direct semantic segmentation of point clouds instead of the fusion with segmented images is also a promising direction of study. If errors in the sensor fusion process caused the imprecisions observed in this work, the segmentation of the point clouds can boost the localization accuracy when using semantic grid maps. The study of different measures for comparing colour data can similarly increase the localization accuracy when using colour grid maps. The fact that humans can drive relying in vision, and the richness of colour data motivate this direction of research. Finally, the accuracies achieved when using occupancy grid maps and reflectivity grid maps suggest that even higher localization accuracies can be achieved by using both maps together. This hypothesis will be evaluated in future works. The ground-truth used for evaluating the localization accuracy is imperfect. The error in the ground-truth can impact the localization evaluation. These factors, the ground-truth accuracy and its impacts in the localization evaluation will be measured in subsequent works. Moreover, techniques for assessing the localization accuracy considering the uncertainty in the ground-truth will be studied.

REFERENCES

[1] National Highway Traffic Safety Administration, U.S. Department of Transportation, “Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey”. 2015. URL: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115. Access in 17/05/2019.

[2] Securing America’s Future Energy, “America’s Workforce and the Self-Driving Future: Realizing Productivity Gains and Spurring Economic Growth”. 2018. URL: https://avworkforce.secureenergy.org/wp-content/uploads/2018/06/SAFE_AV_Policy_Brief.pdf. Access in 17/05/2019.

[3] Bureau of Labor Statistics, “Occupational Outlook Handbook”. United States Department of Labor. 2016. URL: https://www.bls.gov/ooh/. Access in: 17/05/2019.

[4] M. Buehler, K. Iagnemma, and S. Singh, “The 2005 DARPA grand challenge: the great robot race”. Springer-Verlag Berlin Heidelberg. 2007.

[5] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, “Stanley: the robot that won the darpa grand challenge”, in Journal of Field Robotics, vol. 23, no. June, 2007, pp. 1–43.

[6] M. Buehler, K. Iagnemma, and S. Singh, “The DARPA urban challenge: autonomous vehicles in city traffic”. SpringerVerlag Berlin Heidelberg. 2009.

[7] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.- W. Seo, S. Singh, J. Snider, A. Stentz, W. “Red” Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, “Autonomous driving in urban environments: Boss and the urban challenge”, Journal of Field Robotics, vol. 25, no. 8, pp. 425–466, 2008.

[8] F. Mutz, L. P.Veronese, T. Oliveira-Santos, E. de Aguiar, F. A. A. Cheein,, and A. F. De Souza. “Large-scale mapping in complex field scenarios using an autonomous car”. Expert Systems with Applications, 46, 439-462. 2016.

[9] V. Cardoso, J. Oliveira, T. Teixeira, C. Badue, F. Mutz, T. Oliveira-Santos, L. Veronese, and A. F. De Souza, “A model-predictive motion planner for the iara autonomous car”, in 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 225–230.

[10] R. Guidolini, C. Badue, M. Berger and A. F. De Souza, “A Simple Yet Effective Obstacle Avoider for the IARA Autonomous Car”, 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC 2016), Rio de Janeiro,

Brazil, 2016.

[11] R. Guidolini, A. F. De Souza, F. Mutz, and C. Badue, “Neural-based model predictive control for tackling steering delays of autonomous cars”, International Joint Conference on Neural Networks (IJCNN), 2017, pp. 4324–4331.

[12] “Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles”, SAE International, 2016.

[13] R. P. D. Vivacqua, M. Bertozzi, P. Cerri, F. N. Martins, and R. F. Vassallo. “Self-localization based on visual lane marking maps: An accurate low-cost approach for autonomous driving”. IEEE Transactions on Intelligent Transportation Systems, 19(2), 582-597. 2018.

[14] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, “Deepdriving: Learning affordance for direct perception in autonomous driving”. In Proceedings of the IEEE International Conference on Computer Vision. 2015. pp. 2722-2730.

[15] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, and X. Zhang, “End to end learning for self-driving cars”. arXiv preprint arXiv:1604.07316. 2016.

[16] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. F. R. Jesus, R. F. Berriel, T. M. Paixão, F. Mutz, T. Oliveira-Santos, A. F. De Souza, “Self-Driving Cars: A Survey” arXiv: 1901.04407, 2019.

[17] G. Bresson, Z. Alsayed, L. Yu, S. Glaser, “Simultaneous Localization And Mapping: A Survey of Current Trends in Autonomous Driving”. IEEE Transactions on Intelligent Transportation Systems. Vol. 2, nº 3, pp. 194-220. 2017.

[18] J. J. Leonard and H. F. Durrant-Whyte. “Simultaneous map building and localization for an autonomous mobile robot”. In Proceedings IROS'91: IEEE/RSJ International Workshop on Intelligent Robots and Systems' 91. pp. 1442-1447. 1991.

[19] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. “Monte carlo localization for mobile robots.” In proceedings of IEEE International Conference of Robotics and Automation (ICRA). Vol. 2. pp. 1322-1328. 1999.

[20] S. Thrun, W. Burgard, and D. Fox, “Probabilistic robotics”. MIT press. 2005.

[21] H. Durrant-Whyte, and T. Bailey., “Simultaneous localization and mapping: part I”. IEEE robotics & automation magazine, 13(2), 99-110. 2006.

[22] T. Bailey, and H. Durrant-Whyte, “Simultaneous localization and mapping: part II”. IEEE Robotics & Automation Magazine, 13(3), 108-117. 2006.

[23] J. Levinson, M. Montemerlo, and S. Thrun, “Map-based precision vehicle localization in urban environments”. In Robotics: Science and Systems Vol. 4, pp. 1. 2007.

[24] J. Levinson, and S. Thrun, “Robust vehicle localization in urban environments using probabilistic maps”. In 2010 IEEE International Conference on Robotics and Automation (ICRA). 2010. pp. 4372-4378.

[25] R. W. Wolcott, and R. M. Eustice, “Robust LIDAR localization using multiresolution Gaussian mixture maps for autonomous driving”. The International Journal of

Robotics Research, Vol. 36, Num. 3, pp. 292-319. 2017.

[26] M. Dymczyk, M. Fehr, T. Schneider, and R. Siegwart. “Long-term Large-scale Mapping and Localization Using maplab”. arXiv preprint arXiv:1805.10994. 2018.

[27] R. Spangenberg, D. Goehring, and R. Rojas, “Pole-based localization for autonomous vehicles in urban scenarios”. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2161-2166. IEEE. 2016.

[28] M. Sefati, M. Daum, B. Sondermann, K. D. Kreisköther, and A. Kampker. “Improving vehicle localization using semantic and pole-like landmarks”. In proceedings IEEE Intelligent Vehicles Symposium (IV). pp. 13-19. IEEE. 2017.

[29] L. P. Veronese, J. Guivant, F. A. A. Cheein, T. Oliveira-Santos, F. Mutz, E. de Aguiar, and A. F. De Souza. “A light-weight yet accurate localization system for autonomous cars in large-scale and complex environments”. In IEEE 19th International Conference on Intelligent Transportation Systems (ITSC). pp. 520-525. IEEE. 2016.

[30] L. Li, M. Yang, C. Wang, and B. Wang. “Road DNA based localization for autonomous vehicles”. In 2016 IEEE Intelligent Vehicles Symposium (IV). pp. 883-888. IEEE. 2016.

[31] L. P. Veronese, E. de Aguiar, R. C. Nascimento, J. Guivant, F. A. A. Cheein, A. F. De Souza, and T. Oliveira-Santos. “Re-emission and satellite aerial maps applied to vehicle localization on urban environments”. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 4285-4290. IEEE. 2015.

[32] W. Maddern, G. Pascoe, and P. Newman, “Leveraging experience for large-scale LIDAR localisation in changing cities”. In IEEE International Conference on Robotics and Automation (ICRA). pp. 1684-1691. IEEE. 2015.

[33] J. Guivant, and R. Katz, “Global urban localization based on road maps”. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1079-1084. IEEE. 2007.

[34] L. P. Veronese, J. E. Guivant, A. F. De Souza. “Improved Global Urban Localization Based on Road Maps and 3D Detection of Road Intersections”. In proceedings of the Australasian Conference on Robotics and Automation. 2015.

[35] S. Bauer, Y. Alkhorshid, and G. Wanielik, “Using high-definition maps for precise urban vehicle localization”. In IEEE 19th International Conference on Intelligent Transportation Systems (ITSC). pp. 492-497. IEEE. 2016.

[36] I. Kostavelis and A. Gasteratos, “Semantic mapping for mobile robotics tasks: A survey,” Robotics and Autonomous Systems, vol. 66, pp. 86–103, 2015.

[37] Z. Zhao and X. Chen, “Building 3d semantic maps for mobile robots using rgb-d camera,” Intelligent Service Robotics, vol. 9, no. 4, pp. 297–309, 2016.

[38] V. Vineet, O. Miksik, M. Lidegaard, M. Nießner, S. Golodetz, V. A. Prisacariu, O. Kahler, D. W. Murray, S. Izadi, P. P ¨erez, ´ et al., “Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction,” in Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 75–82, IEEE, 2015.

[39] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, “Semanticfusion: Dense 3d semantic mapping with convolutional neural networks,” in 2017 IEEE International Conference on Robotics and automation (ICRA), pp. 4628–4635, IEEE, 2017.

[40] S. Yang, Y. Huang, and S. Scherer, “Semantic 3d occupancy mapping through efficient high order crfs,” arXiv preprint arXiv:1707.07388, 2017.

[41] S. Sengupta, P. Sturgess, P. H. Torr, et al., “Automatic dense visual semantic mapping from street-level imagery,” in Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, pp. 857–862, IEEE, 2012.

[42] S. Sengupta, and P. Sturgess. “Semantic octree: Unifying recognition, reconstruction and representation via an octree constrained higher order MRF”. In IEEE International Conference on Robotics and Automation (ICRA). pp. 1874-1879. IEEE. 2015.

[43] D. Lang, S. Friedmann, and D. Paulus. “Semantic 3D Octree Maps based on Conditional Random Fields”. MVA. Vol. 13, pp. 185-188. 2013.

[44] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation”. In Proceedings of the European Conference on Computer Vision (ECCV). pp. 801-818. 2018.

[45] A. Segal, D. Haehnel, and S. Thrun. “Generalized-icp”. In Robotics: science and systems. Vol. 2, No. 4, pp. 435. 2009.

[46] J. Weingarten and R. Siegwart. “EKF-based 3D SLAM for structured environment reconstruction”. In 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 3834-3839. IEEE. 2005.

[47] E. A. Wan, and R. Van Der Merwe. “The unscented Kalman filter for nonlinear estimation”. In Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium. Cat. No. 00EX373. pp. 153-158). IEEE. 2000.

[48] G. Sibley, G. S. Sukhatme, and L. H. Matthies, “The Iterated Sigma Point Kalman Filter with Applications to Long Range Stereo”. In Robotics: Science and Systems Vol. 8, No. 1, pp. 235-244. 2006.

[49] S. Huang, and G. Dissanayake, “Convergence and consistency analysis for extended Kalman filter based SLAM”. IEEE Transactions on robotics, 23(5), 1036-1049. 2007.

[50] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. “Fastslam: A factored solution to the simultaneous localization and mapping problem”. In Proceedings of the AAAI/IAAI. pp. 593–598. 2002.

[51] M. Montemerlo, S. Thrun. “Fastslam 2.0. FastSLAM: A scalable method for thesimultan eous localization and mapping problem in robotics, pp. 63–90. 2007.

[52] S. Thrun, and, M. Montemerlo, “The graph SLAM algorithm with applications to large-scale mapping of urban structures”. The International Journal of Robotics Research, Vol. 25, pp. 403-429. 2006.

[53] F. Lu, and E. Milios, “Globally consistent range scan alignment for environment

mapping". Autonomous robots, Vol. 4, nº 4, 333-349. 1997.

[54] T. Duckett, S. Marsland, and J. Shapiro, "Learning globally consistent maps by relaxation". In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065). Vol. 4, pp. 3841-3846. IEEE. 2000.

[55] U. Frese, and G. Hirzinger, "Simultaneous localization and mapping-a discussion". In Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics (pp. 17-26). 2001.

[56] K. Konolige, "Large-scale map-making". In AAAI pp. 457-463. 2004.

[57] M. Golfarelli, D. Maio, and S. Rizzi "Elastic correction of dead-reckoning errors in map building". In Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190). Vol. 2, pp. 905-911. IEEE. 1998.

[58] L. D. L. Perera, W. S. Wijesoma, S. Challa, and M. D. Adams. "Sensor bias correction in simultaneous localization and mapping". In 6th Int. Conf. on Information Fusion. pp. 151-158. 2003.

[59] R. Kümmerle, "State estimation and optimization for mobile robot navigation". (Doctoral dissertation, Verlag nicht ermittelbar). 2013.

[60] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. "OctoMap: An efficient probabilistic 3D mapping framework based on octrees". Autonomous robots, Vol. 34, nº 3, 189-206. 2013.

[61] J. Chen and S. Shen, "Improving octree-based occupancy maps using environment sparsity with application to aerial robot navigation", 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 3656–3663.

[62] D. Droeschel, M. Schwarz, and S. Behnke, "Continuous mapping and localization for autonomous navigation in rough terrain using a 3d laser scanner", Robotics and Autonomous Systems, vol. 88, pp. 104–115, 2017.

[63] S. Sengupta, P. Sturgess, and P. H. Torr. "Automatic dense visual semantic mapping from street-level imagery". In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 857-862. IEEE. 2012.

[64] N. Atanasov, M. Zhu, K. Daniilidis, and G. J. Pappas. "Semantic Localization Via the Matrix Permanent". In Robotics: Science and Systems, Vol. 2. 2014.

[65] N. Atanasov, M. Zhu, K. Daniilidis, and G. J. Pappas. "Localization from semantic observations via the matrix permanent". The International Journal of Robotics Research, Vol. 35, pp. 73-99. 2016.

[67] N. Otsu, "A threshold selection method from gray-level histograms". IEEE transactions on systems, man, and cybernetics, Vol. 9, nº 1, pp. 62-66. 1979.

[68] L. C. Fernandes, J. R. Souza, G. Pessin, P. Y. Shinzato, D. Sales, C. Mendes, et al. "CaRINA intelligent robotic car: architectural design and applications". Journal of Systems Architecture, Vol. 60, nº 4, pp. 372-392. 2014.

[70] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. “Introduction to autonomous mobile robots”. MIT press. 2011.

[71] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. “Principles of robot motion: theory, algorithms, and implementation”. MIT press. 2005.

[72] P. Corke. “Robotics, vision and control: fundamental algorithms In MATLAB® second, completely revised”. Vol. 118. Springer. 2017.

[73] R. Eberhart, and J. Kennedy. “A new optimizer using particle swarm theory”. In MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. pp. 39-43. IEEE. 1995.

[74] M. Clerc, and J. Kennedy “The particle swarm-explosion, stability, and convergence in a multidimensional complex space”. IEEE transactions on Evolutionary Computation, Vol. 6, nº 1, pp. 58-73. 2002.

[75] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, et al. “The cityscapes dataset for semantic urban scene understanding”. In Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3213-3223. 2016.

[76] R. Hartley, and A. Zisserman. “Multiple view geometry in computer vision”. Cambridge university press. 2003.

[77] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. “Vision meets robotics: The KITTI dataset”. The International Journal of Robotics Research, Vol. 32, nº 11, pp. 1231-1237. 2013.

[78] S. Thrun. “Robotic mapping: A survey”. Exploring artificial intelligence in the new millennium. 2002.

[79] I. A. Bârsan, P. Liu, M. Pollefeys, and A. Geiger. “Robust dense mapping for large-scale dynamic environments”. In IEEE International Conference on Robotics and Automation (ICRA). pp. 7510-7517. IEEE. 2018.

[80] A. Geiger, M. Roser, and R. Urtasun. “Efficient large-scale stereo matching”. In Asian conference on computer vision. pp. 25-38. Springer, Berlin, Heidelberg. 2010.

[81] J. R. Chang, and Y. S. Chen. “Pyramid stereo matching network”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5410-5418. 2018.

[82] Y. Chen, C. Schmid, and C. Sminchisescu. “Self-supervised Learning with Geometric Constraints in Monocular Video: Connecting Flow, Depth, and Camera”. arXiv preprint arXiv:1907.05820. 2019.

[83] Y. Mou, M. Gong, H. Fu, K. Batmanghelich, K. Zhang, and D. Tao. “Learning Depth from Monocular Videos Using Synthetic Data: A Temporally-Consistent Domain Adaptation Approach”. arXiv preprint arXiv:1907.06882. 2019.

[84] C. H. Yeh, Y. P. Huang, and M. J. Chen. “Scale Invariant Multi-view Depth Estimation Network with cGAN Refinement”. In International Computer Symposium. pp. 681-687. Springer, Singapore. 2018.

[85] J. Astola, and I. Virtanen. “A measure of overall statistical dependence based on the entropy concept.”. University of Vaasa. 1983.

[86] C. Studholme, D. L. Hill, and D. J. Hawkes. “An overlap invariant entropy measure of 3D medical image alignment”. Pattern recognition, Vol. 32, nº 1, pp. 71-86. 1999.

[87] H. T. Amaral-Silva,L. O. Murta-Jr, P. M. de Azevedo-Marques, L. V. B. Wichert-Ana, and C. Studholme, “Validation of Tsallis Entropy”. In Inter-Modality Neuroimage Registration. arXiv preprint arXiv:1611.01730. 2016.

[88] A. Bardera, M. Feixas, I. Boada, and M. Sbert. “High-dimensional normalized mutual information for image registration using random lines”. In International Workshop on Biomedical Image Registration. pp. 264-271. Springer, Berlin, Heidelberg. 2006.

[89] N. D. Cahill. “Normalized measures of mutual information with general definitions of entropy for multimodal image registration”. In International Workshop on Biomedical Image Registration. pp. 258-268. Springer, Berlin, Heidelberg. 2010.

[90] M. A. Brubaker, A. Geiger, and R. Urtasun. “Lost! leveraging the crowd for probabilistic visual self-localization”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3057-3064. 2013.

[91] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. “g2o: A general framework for graph optimization”. In 2011 IEEE International Conference on Robotics and Automation. pp. 3607-3613. IEEE. 2011.

[92] M. Montemerlo, N. Roy, and S. Thrun. “Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit”. In Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Cat. No. 03CH37453). Vol. 3, pp. 2436-2441. IEEE. 2003.