FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Methodology to Evaluate the Performance of LiDAR-based SLAM Algorithms

Gonçalo Fernandes Pereira

Mestrado em Engenharia Informática e Computação

Supervisor: Miguel Fernando Paiva Velhote Correia

Second Supervisor: José Alberto Peixoto Machado da Silva

July 31, 2022

This work is supported by European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project nº 047264; Funding Reference: POCI-01-0247-FEDER-047264].

A Methodology to Evaluate the Performance of LiDAR-based SLAM Algorithms

Gonçalo Fernandes Pereira

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. Luis F. Teixeira
External Examiner: Prof. António Silva
Supervisor: Prof. Miguel Velhote Correia
Supervisor: Prof. José Machado da Silva

July 31, 2022

Resumo

Os carros autónomos destinam-se a operar num ambiente público não estruturado do mundo real. A abertura da complexidade do próprio sistema e a situação regulatória representam um desafio considerável para a indústria automóvel.

Acidentes recentes envolvendo veículos a operar em modos autónomos resultaram de falhas na camada de perceção. Muitas pesquisas ainda são necessárias para prevenir esse tipo de acidente por meio do entendimento e caracterização (e treinamento) do comportamento dos algoritmos de perceção em inúmeras situações de condução. É essencial ter um módulo de validação que forneça dados relevantes para treinar algoritmos de Deep Learning, simular diferentes situações de direção e avaliar o desempenho das funções de perceção para ter algoritmos de perceção robustos e precisos. O uso de key performance indicators relevantes, precisos e significativos é fundamental para selecionar as melhores arquiteturas/topologias de rede neural e modelos para uma tarefa de perceção.

Validar a segurança da condução autónoma é um tema complexo e de relevância prática. Como novas abordagens são necessárias, a prova estatística de segurança baseada em testes de campo não é escalável. Uma estratégia que pode ser útil na área de perceção é utilizar algoritmos de Simultaneous Localization and Mapping (SLAM) para gerar ground truths para uma das características da perceção, a deteção de objetos estáticos. Esses algoritmos realizam o mapeamento e a localização de recursos relevantes numa cena específica de estrada e o seu objetivo é construir uma representação globalmente consistente do ambiente.

Esta dissertação tem como objetivo estabelecer um método para avaliar quantitativamente o desempenho de um algoritmo SLAM e, com o auxílio desta ferramenta, verificar o quão confiável é o algoritmo para que possa ser utilizado como ground truth na camada de perceção, na extração de objetos estáticos, avaliando a trajetória estimada e o mapa gerado pelo algoritmo SLAM.

Palavras-Chave: Condução Autónoma, SLAM, Validação, KPI, Ground Truth, Segurança

Abstract

Autonomous driving vehicles are intended to operate in an unstructured, public real-world environment. The openness of the system's complexity and the regulatory situation poses a considerable challenge to the automotive industry.

Recent accidents involving vehicles operating in autonomous modes resulted from the perception layer failures. Much research is still needed to prevent this type of accident through understanding and characterizing (and training) the behavior of Perception algorithms under numerous driving situations. It is essential to have a validation module that provides relevant data to train deep learning algorithms, simulate different driving situations, and evaluate the performance of the perception functions to have robust and accurate Perception algorithms. Using relevant, precise, and meaningful key performance indicators is critical to select the best neural network architectures/topologies and models for a perception task.

Validating the safety of autonomous driving is a complex topic and has practical relevance. Since new approaches are required, statistical proof of security based on field testing does not scale. A strategy that can be useful in the perception area is to use Simultaneous Localization and Mapping algorithms (SLAM) to generate ground truths for one of the features of perception, the detection of static objects. These algorithms perform mapping and localization of relevant features in a specific road scene, and their purpose is to build a globally consistent representation of the environment.

This dissertation aims to establish a method to evaluate the performance of a SLAM algorithm quantitatively and, with the help of this tool, check how reliable the algorithm is so that it can be used as ground truth in the perception layer in the extraction of static objects. This is achieved by evaluating the estimated trajectory and the map generated by the SLAM algorithm.

Keywords: Automated Driving, SLAM, Validation, KPI, Ground Truth, Safety

Acknowledgements

I want to thank my FEUP supervisors, Miguel Fernando Paiva Velhote Correia and José Alberto Peixoto Machado da Silva for all the support, availability, and advice they gave me throughout this dissertation's development. It was a crucial help.

I want to also give a huge thanks to Joana Santos and Diogo Matos, who were tireless in integrating me into the project and the team. Thanks for all the support!

I would also like to thank the ENG63 Bosch team for welcoming me in the best way in the company and giving me all the support and tools to carry out this work. They were essential for this project, and I learned much from the company and every single one.

I want to thank my mother, father, and grandmother. They have always been present, supported, and motivated me on my journey of college and that always had been there for everything!

Finally, I wanted to thank all my friends for giving me strength, believing, and helping me in everything I needed. Without all these people, this project would not have been possible, and I am very grateful to all of them for never giving up on me.

Gonçalo Pereira

"If you recognize that self-driving cars are going to prevent car accidents,
AI will be responsible for reducing one of the leading causes of death in the world."

Mark Zuckerberg

Contents

List of Figures

List of Tables

Abbreviations

SLAM	Simultaneous Localization and Mapping
LiDAR	Light Detection and Ranging
RADAR	Radio Detection and Ranging
ROS	Robot Operating System
AD	Autonomous Driving
KPI	Key Performance Indicator
GPS	Global Positioning System
FOV	Field of View
RMSD	Root Mean Squared Deviation
SVD	Singular Value Decomposition
MSE	Mean Squared Error
MAE	Mean Absolute Error

Chapter 1

Introduction

1.1 Problem and Motivation

LiDAR (Light Detection And Ranging), sometimes called “laser scanning” or “3D scanning”, is an optical remote sensing device and technology that measures properties of reflected light (mainly, time of flight and incidence angle) in order to obtain 3D coordinates, and eventually other information, of the reflecting object. This detection and measurement process allows to create a 3D model of the studied scene using eye-safe laser beams [1]. The set of point reflections captured from an illuminated scene typically comprises point clouds that provide the basis to obtain a detailed and up-to-date map after each scanning.

Given the LiDAR pointcloud, it is possible to detect and recognize different static and moving obstacles in the vehicle’s surrounding space. This operation is performed within the perception data processing module. This task is usually performed considering also data from other imaging devices, such as cameras and RADARs, to ensure higher identification accuracy and better perception [3].

In the past few years, accidents involving vehicles in autonomous driving mode have been observed, resulting from failures in the Perception layer. For example, in an accident involving a vehicle from Tesla (a company known for not using LiDAR sensors), the perception algorithms could not prevent the accident. The RADAR sensor misclassified a stationary object as noise and false positive, even with the support of a thermal sensor [15].

Another example is the accident with an Uber vehicle involving a pedestrian holding a bicycle while crossing the road in an urban setting at night. According to the National Highway Traffic Safety Administration report, the LiDAR and RADAR sensors installed in the vehicle detected the

pedestrian about 6 seconds before impact. They only identified and misclassified it as an "unknown object." Due to reduced visibility at night, the cameras were useless in this situation [14].

More research is needed to avoid these types of accidents and to better understand and characterize the behavior of perception algorithms for multiple driving situations. Therefore, to obtain solid and accurate algorithms, it is essential to have a validation module that provides relevant data for training deep learning algorithms, simulating different driving scenarios, and evaluating the performance of perception functions.

The validation of automated driving is a problem of tremendous complexity and practical relevance. It can involve different strategies, such as test track tests, field tests, simulation, and even domain and system analysis, that contribute to validate the models and establish a continuous improvement of the systems and consecutively the models.

Field testing, for example, can uncover edge case possibilities that would be difficult to detect otherwise. Recorded field test data must be reusable in a closed-loop simulation environment to meet the continuous improvement challenge for testing and generalizing such scenarios in a repeatable manner. Test cases should be created via systematic examination of both the autonomous driving system and the simulation environment and random sampling-based field tests. Improving the realism of the simulation environment is also necessary in order to rely on virtual testing efficiency. Otherwise, crucial instances could be neglected during virtual testing and revealed as surprises during field testing.

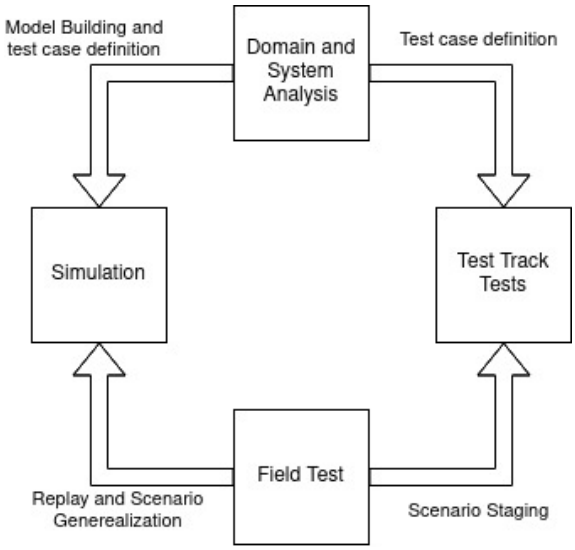

Figure 1.1: Model Validation

As such, Key Performance Indicators (KPI) that are relevant, accurate, and meaningful are crucial to evaluating and selecting the best data processing, notably neural networks, and models for the perception task.

Simultaneous Localization and Mapping (often abbreviated to SLAM) addresses the problem of a robot or an autonomous vehicle navigating an unknown environment [8]. While navigating the environment, the SLAM algorithm helps the robot or vehicle to create a map and, at the same time, tries to localize itself within that very same map. SLAM problems can be helpful in two different situations: to acquire detailed environment models or seek to maintain a real sense of robot or vehicle location. SLAM serves both of these purposes.

1.2 Objectives

This dissertation aims to establish a method to evaluate the performance of the SLAM algorithms quantitatively and, with the help of this tool, check how reliable the algorithm is so that it can be used as ground truth in the perception layer, notably in the extraction of static objects.

1.3 Main contribution

The approach being proposed evaluates the performance of the LiDAR-SLAM algorithms, which will allow the developers to ensure the quality of the ground truth in the perception layer in the extraction of static objects. For that purpose, the algorithm compares the estimated trajectory by the SLAM algorithm with the Odometry data and compares the map generated using the Bosch LiDAR sensor with the map generated by the Velodyne LiDAR sensor (the reference sensor in the market)

Through this methodology, which provides meaningful key performance indicators, the developers can have a much more robust validation module due to more robust ground truths.

1.4 Document Structure

This dissertation is divided in six chapters. The present chapter, chapter 1 provides an introduction, presenting the problem, motivation, objectives, and the main contribution of the work that was carried out. Next, chapter 2 provides some essential concepts to understand the domain in which this work is included, i. e., how a LiDAR works, what is the SLAM process and how it works, and related work that is helpful to understand what has already been done. After that, in chapter 3 is presented the development environment and data structures, with a detailed description of the LiDAR specifications, the ROS (Robot Operating System) bags structure, and the Scenarios used. Then in chapter 4, the methodology for the trajectory and the map comparison is presented in detail, and the results are shown in chapter 5, along with an analysis of the results. Finally, in chapter 6, some final considerations are presented, and the dissertation is concluded.

Chapter 2

State of Art

2.1 The SLAM Algorithm

During its operation, the LiDAR sensor emits pulsed light waves into the surrounding environment, which bounce off surrounding objects and return to the sensor (Figure 2.1). Then the sensor uses the time it took for each pulse to fly to the object and return to the sensor to calculate the distance from the LiDAR to the object.

SLAM is a complex task due to the need for involving two procedures that must be processed simultaneously. That is, the target vehicle needs to localize itself within a map generated concurrently. This complexity derives from the mutual dependency between the posture and map estimations, requiring searching for a solution in a high-dimensional environment.

The SLAM algorithm being evaluated is a particular SLAM method called LiDAR-SLAM; this means that the algorithm's primary input will be the data collected by the LiDAR sensor. The algorithm generates a map of the static environment and an estimated trajectory (in the case there is a displacement of the equipment in relation to the environment) using this data. In order to be used as ground truth, the generated map of the static objects must be validated to have a high degree of reliability.

The existing algorithm [12] is a lifelong SLAM algorithm that analyzes a robot's long-term operation in changing settings, revisiting previously mapped places. However, the initial graph-based SLAM algorithms became incredibly slow due to the graph's constant development and sparsity loss. A graph pruning technique can solve both concerns by carefully deleting vertices and edges while maintaining the information required for optimal SLAM results. An example of the effects of pruning is shown in Figure 2.2.

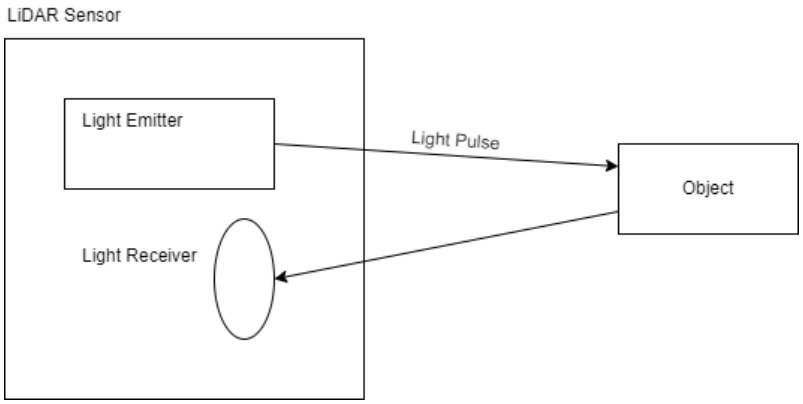

Figure 2.1: LiDAR Architecture

This algorithm was evaluated on two publicly available real-world, long-term datasets compared to the unpruned case and ground truth. They concluded that even on an extended dataset (captured during 25 hours), the approach manages to keep the graph sparse and the speed high while still providing good accuracy (40 times speed up, 6 cm map error compared to the unpruned case).

The purpose of graph pruning is to remove unnecessary vertices and edges with as little loss of accuracy in localization and mapping as possible. The algorithm assures that the graph's size is solely proportional to the size of the environment and that it is unaffected by the amount of time spent there or the length of the robot's journey.

This algorithm returns a ROS bag containing a point cloud with a vast collection of the individual point plotted in a 3D space and the estimated trajectory of the vehicle.

2.2 Related Work

Regarding this work, we will be using and evaluating a particular type of SLAM, the LiDAR-SLAM algorithm [4], which means that the algorithm's primary input is the data collected from the LiDAR sensor. However, there are other types of SLAM algorithms, such as Visual SLAM [2], Acoustic SLAM [5], and Collaborative SLAM [10].

The LiDAR-SLAM provides various advantages compared to other SLAM types, such as more accurate and consistent results, because, due to the short wavelength of the emitted laser emission, it provides higher resolution detection and allows to determine which objects are detected, such as people, walls, or cars, for example.

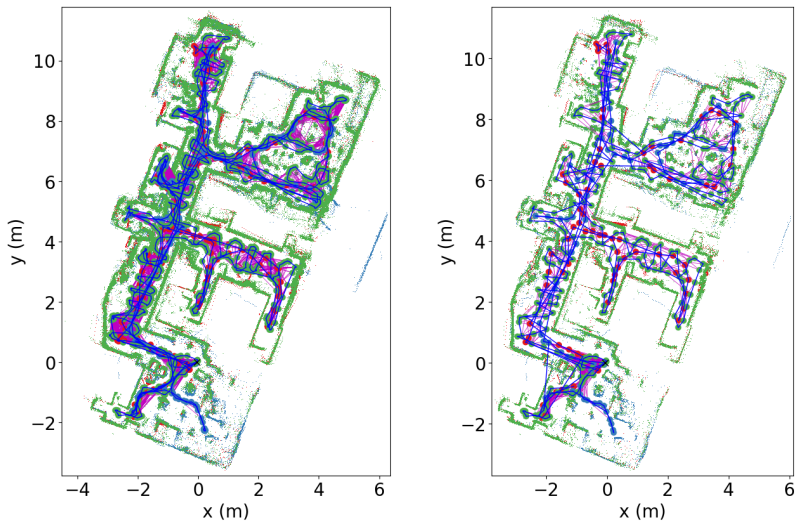

Figure 2.2: SLAM map of an apartment without (left) and with pruning (right) using the proposed approach.

2.2.1 SLAM Performance Evaluation

The SLAM algorithm can be executed in run time or offline. Although, to better evaluate the performance of the SLAM, some existing works capture and record the data in ROS bags to make the comparison as fair as possible and place SLAM algorithms on an equal footing. ROS topics are recorded with an appropriate time sampling in ROS bag files. The advantage of this method is that there are no differences between the data obtained from a recorded bag file and real-time data. In the case of [17], as they use the input to a mobile robot platform, they even record the joystick input used to teleoperate the robot to later replicate the same input in the SLAM algorithm while offline.

Because the study above intended to compare the performance between different SLAM algorithms, the best approach was to save the data to run the SLAM algorithms offline. Once we are only evaluating the performance of only one existing SLAM algorithm, our methodology can be applied either in run time or offline, with no negative impact on the results or conclusions.

Regarding metrics, [13] offers a metric that only considers relative geometric relationships between positions along a robot's path. The metric considers the deformation energy needed to transfer the estimate into the ground truth. Even if perfect ground truth data is not accessible, their method enables comparisons. However, this measure does not involve a global reference frame and relies on close relationships between poses, and it requires manual work by humans familiar with the environment's topology.

When choosing ground truths, [13] uses two approaches: GPS (to bound the global uncertainty of a vehicle moving outdoors) and aerial images (extracted from popular web tools like Google-Earth or Microsoft Live-Earth). On smaller scales,[11] used a Microsoft Kinect RGB-D to obtain

accurate and synchronized ground truth poses.

When comparing the trajectory estimated by the SLAM algorithm with the ground truth, some alignment may be needed in some cases. For example, in [18], they only faced translation alignment because the type of SLAM they are using is the vSLAM, and "it is usually sufficient to calculate the trajectory alignment transformation using only the translational components of the estimation and the ground truth". They adapted Umeyama's method [16] to align the trajectory estimated with the ground truth to calculate the transformation for visual-inertial systems. In [13] they did not face any alignment problems because their method does not involve any global reference frame.

Regarding pointcloud comparison, the most common and adopted method was comparing the map of the static objects sorted out by the SLAM algorithm with absolute ground truth generated by a FARO laser tracker or a MultiStation, for example. They both constitute a reference in the market with exact measurements.

In terms of metrics, [17] used the average distance to the nearest neighbor as a metric (ADNN). It concluded that it was a good metric to evaluate the performance of the algorithms in the perception of static objects. In [6] they address 2D SLAM, so they represent the map by an occupancy grid. They implemented three metrics: the proportion of occupied and free cells, the number of corners on a map, and the number of enclosed areas.

Chapter 3

Development Environment and Data Structures

3.1 Development Environment

The development environment was Ubuntu, version 20.04, a Linux distribution, and the ROS version used was Noetic. Therefore, the code developed within the scope of this thesis was also in C++, since it is the language already implemented in the existing code and it has much faster execution time than for example Python.

On each test campaign, the car collects data, like the Odometry data from the iMAR sensor and GPS, Camera Images, and pointclouds from the Bosch LiDAR and the Velodyne VLS-128. After collecting data on a test campaign, the data is split into smaller ROS bags due to the size and to facilitate the processing of individual chunks.

Scenarios with specific characteristics were selected in this work to have a richer set of results for analysis that allow drawing conclusions and verifying how different factors influence the outcomes between different scenarios, such as velocity, distance drove, or time analyzed by the SLAM algorithm.

3.2 LiDAR Sensors

The Rosbags used as input have data from both these sensors because, within this work's scope, one of the methodologies implemented to compare the performance of SLAM required data from

each of those sensors. As we will see, each sensor's specifications will influence the processing. The Velodyne sensor is the reference sensor in the market, so it is used in the analysis of the SLAM relative to the Bosch LiDAR sensor.

The following table shows the characteristics of both LiDAR sensors used to capture the data from around the car, which will be afterwards used as input to run the SLAM algorithm.

	VLS-128	Bosch B2
Measurement Range	up to 300 m	up to 250 m
Range accuracy	+/- 3 cm	+/- 5 cm
Vertical FOV	40 degrees	24 degrees
Horizontal FOV	360 degrees	120 degrees
Angular Resolution	0.1 degrees	0.05 degrees

Table 3.1: LiDAR Sensors Specifications

3.3 ROS Bags structure

3.3.1 Input ROS Bag

This subsection describes the structure of the bags used as input to the SLAM algorithm. Before running the SLAM algorithm, the bag is processed, applying a transformation matrix to all the messages in the Velodyne topic, so that the pointclouds obtained by both sensors are aligned.

Since the Velodyne and the Bosch sensor have different specifications, the pointcloud topics have been adjusted to have common characteristics. The Horizontal FOV (Field of View) has been reduced to 120 degrees, the Vertical FOV reduced to 24 degrees, and the range adjusted to between 3 meters and 240 meters. The various topics contain the following messages of:

/camera/image type sensor_msgs/Image, and an uncompressed image obtained by the camera installed in the car.

/imar/full_odom type lidar_msgs/Odometry, a custom type created to hold information about the position, velocity, linear acceleration and GPS position estimation.

/imar/odom type nav_msgs/Odometry, and it includes an estimation of a position and velocity in free space.

/rblidar/pointcloud type sensor_msgs/PointCloud2, including a collection of points broadcasted from the Bosch Sensor that form the pointcloud.

/velodyne/pointcloud type sensor_msgs/PointCloud2, including a collection of points broadcasted from the Velodyne Sensor that form the pointcloud.

3.3.2 Output Trajectory ROS Bag

The structure of the bags outputted by the SLAM contains the estimated trajectory output by the slam and the data extracted from the odometry. This is the information that is going to be analyzed in order to evaluate the performance of the SLAM algorithm in the generation of estimated trajectories.

/trajectory/base/pose This topic contains messages of type geometry_msgs/Pose, including the position and orientation in relation to the start position of the bag analyzed by the SLAM algorithm.

/odometry/base/pose This topic contains messages of type geometry_msgs/Pose, including the position and orientation in relation to the start position of the test campaign.

3.4 Scenarios Description

In this chapter, the test scenarios will be presented, used to test the different factors that could influence the evaluation result. Each Scenario is labeled with a letter to facilitate data processing.

The Table 3.2 represents the distance traveled and average velocity for each scenario. The distance traveled was calculated using the sequence of positions extracted from the odometry. It results from the sum of all the differences between each pair of consecutive positions. The velocity was calculated by dividing the distance traveled by the scenario's duration. It also shows if the Scenario has dynamic objects and what is the type of each scenario.

	Distance Travelled (m)	Average Velocity (m/s)	Dynamic Objects	Type of Scenario
Scenario A	48.2	4.8	no	Urban
Scenario B	51.8	4.7	yes	Urban
Scenario C	58.4	5.8	yes	Urban
Scenario D	103.4	10.3	no	Urban
Scenario E	260.5	8.7	yes	Highway
Scenario F	261.0	21.8	yes	Highway

Table 3.2: Scenario Overview

It is important to note that each scenario was chosen intentionally and with specific characteristics to allow conclusions by comparing different scenarios, trying to keep some features constant. For example, to evaluate the possible impact of dynamic objects on the performance of the SLAM algorithm (such as imprecision in the scan matching process), we use Scenario A and Scenario B. These scenarios have about the same distance traveled and average speed, while Scenario B has dynamic objects and Scenario A does not.

Another factor to be evaluated is the average speed in each scenario. To make this comparison, for example, Scenario E and Scenario F are used because they have about the same distance covered, the same type of scenario, and both have dynamic objects. But Scenario E has more than twice the average speed of Scenario F, with Scenario E having a duration of about thirty seconds and Scenario F about ten seconds.

This data will be helpful further ahead to analyze and make conclusions on the results.

Scenario A (Figure 3.1) Short section without any dynamic object, at reduced speed, with a first left turn followed by another to the right.

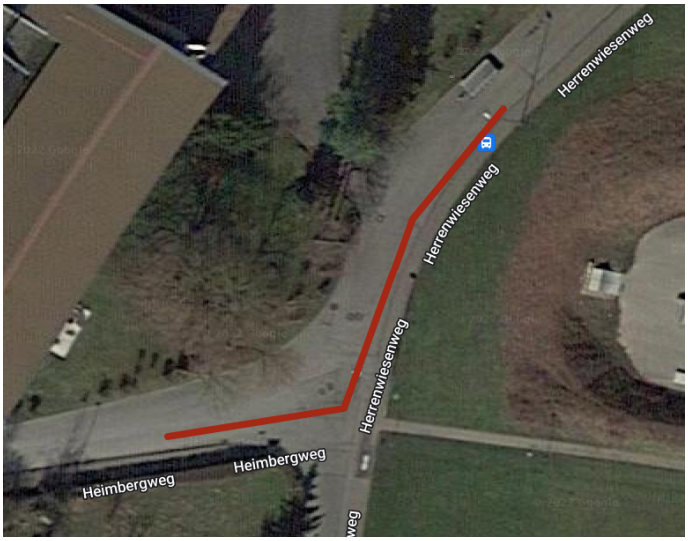

Figure 3.1: Scenario A Route

Scenario B (Figure 3.2) Short section with cars passing in the opposite direction, with a right 90 degrees turn at reduced speed.

Scenario C (Figure 3.3) Short and slow section, more or less straight, in which the car passes cyclists (who are traveling in the same direction) and in which cars simultaneously pass in the opposite direction.

Scenario D (Figure 3.4) Section with double the distance and velocity of Scenario C. It's also a more or less straight section, with no dynamic objects in the scene.

Scenario E (Figure 3.5) Long section, and it consists of an access ramp to a motorway composed of a curve to the right followed by a turn and counter-curve.

Scenario F (Figure 3.6) Long section with the same distance traveled in Scenario E but more than double the average velocity. It consists of a section in the highway the car entered in the previous scenario and contains multiple cars and trucks passing by in the opposite direction.

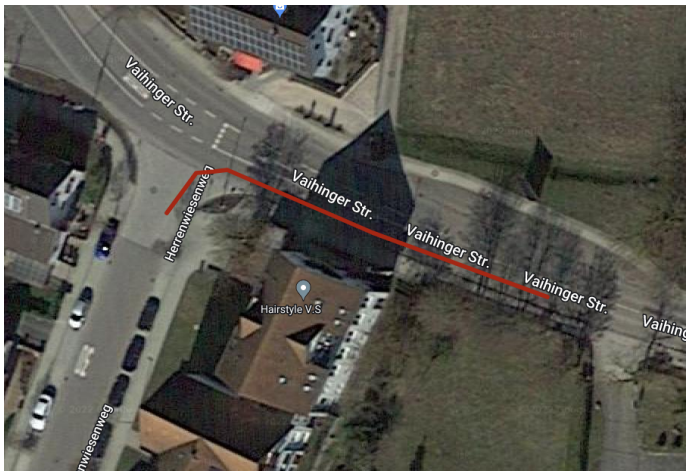

Figure 3.2: Scenario B Route

Figure 3.3: Scenario C Route

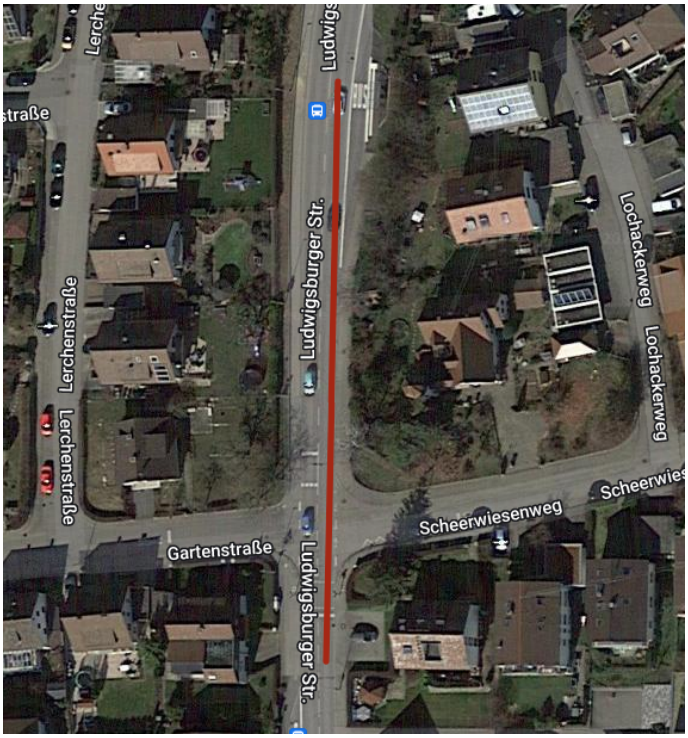

Figure 3.4: Scenario D Route

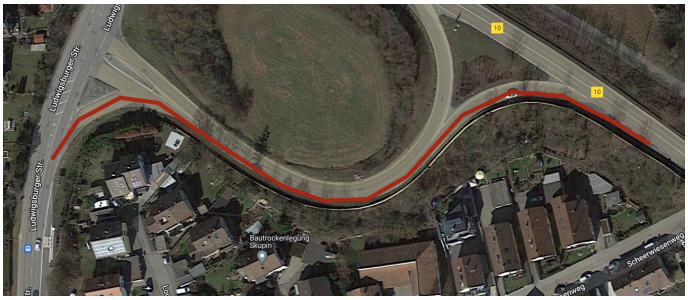

Figure 3.5: Scenario E Route

Figure 3.6: Scenario F Route

Chapter 4

Methodology

4.1 Trajectory Comparison

Trajectory comparison can be an essential tasks for evaluating the SLAM algorithm because the comparison of estimated trajectory and odometry data can represent how well the SLAM can represent the static objects. Once we compare the executions of the SLAM algorithm using each LiDAR sensor, we can raise conclusions about the algorithm's performance with sensors with different specifications. This approach aims to compare the trajectory obtained by the iMAR sensor (considered a ground truth) with the trajectory output by the SLAM algorithm.

The trajectory is output in the rostopics trajectory/base/pose and odometry/base/pose as a ROS message of type geometry_msgs/Pose. In addition to containing the position of each point of the trajectory, these messages also contain the orientation. Figure 4.1 represents a flow diagram of the general operation of the trajectory comparison.

The program starts by opening the bag in reading mode only and queries the bag for the desired topics. After that, we align both trajectories because they are in different frames of reference. The position of the odometry topic messages is relative to the initial pose of the campaign. In contrast, the position of the estimated trajectory messages is relative to the point where the SLAM algorithm started running.

We store the data in custom structures to ease the organization of the code responsible for the alignment and order them by their timestamp to guarantee time consistency. The calculus of the transformation matrix that aligns the trajectory begins by using the Kabsch algorithm [9], a method for calculating the optimal matrix between two paired sets of points that minimizes the Root Mean Squared Deviation (RMSD).

The steps of this algorithm are:

1. Computation of the covariance matrix.

2. Calculate the singular value decomposition (SVD) of the covariance matrix.

3. Check if we need to correct the rotation matrix to guarantee a right-handed coordinate system.

4. Calculate the optimal rotation matrix.

After obtaining the rotation matrix, we apply the transformation to one side of each pair of points, and then, with each pair of points, we calculate the distance between them and store the values to apply the metrics.

To make this comparison, the metrics chosen to evaluate the performance of the SLAM are the Mean Squared Error (MSE) — the average squared difference between the Velodyne sensor and the Bosch sensor —, and the Mean Absolute Error (MAE) — the average absolute error expressed in meters between the values obtained by the SLAM with the Velodyne sensor and the Bosch sensor. To further analyze the results in more specific cases, the MAE was separated in each axis (MAEx, MAEy, MAEz)

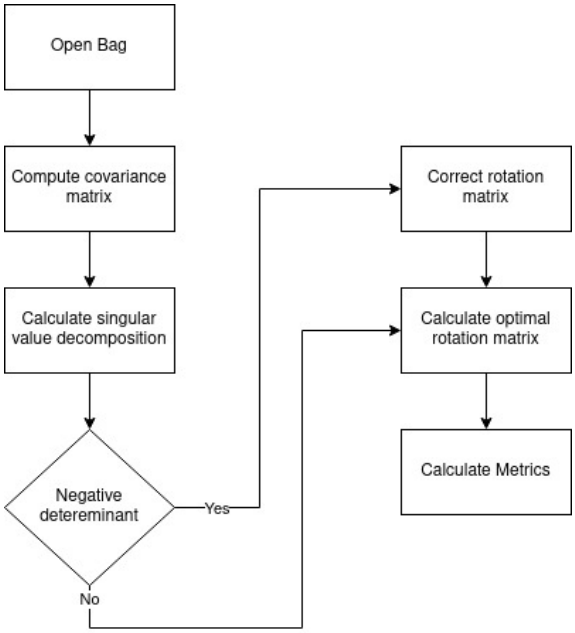

Figure 4.1: Trajectory comparison flowchart

4.2 Map Comparison

Map comparison is another approach to evaluate the performance of the SLAM algorithm. The objective is to compare the results of the SLAM algorithm using the Velodyne LiDAR sensor as a reference with the map generated by the SLAM using the Bosch LiDAR sensor.

This approach encompasses some challenges because of the different specifications of the sensors used. If we take a look at Table 3.1 it can be seen that the sensors have different Measurement Range, Range Accuracy, Vertical FOV, Horizontal FOV, and angular resolution. To better compare the maps that result from the SLAM algorithm using each sensor, we restricted the pointcloud topics of the input bags and adjusted them to have common characteristics.

The Horizontal FOV has been reduced to 120 degrees, the Vertical FOV reduced to 24 degrees, and the range adjusted to between 3 meters and 240 meters. Although, there are factors we cannot control, for example, the map density (the map extracted from the Bosch LiDAR sensor is denser than the map extracted from the Velodyne Sensor).

In this comparison, we used two different metrics: the Hausdorff distance and the Mean Euclidean Distance.

The Hausdorff distance measures how far two subsets of a metric space are from each other. In other words, it is the greatest of all distances from a point in one set to the closest point in the other set. The Mean Euclidean Distance is the mean of the distance of the closest point from one map to another.

The program starts by opening both bags of the SLAM output containing the map generated by the SLAM. After that, for each map generated by the SLAM algorithm, a K-D tree will be created, which provides a fast nearest neighbor search. This was chosen because it optimizes the algorithm in searching for the nearest points of the other map.

The next step is, for each map point (obtained by the SLAM algorithm using each of the sensors), to find the nearest neighbor of the other map, searching in the K-D tree generated.

For each point, to obtain the Hausdorff distance, it will check if this is the maximum distance already recorded from each pair of points. If it is the maximum, it stores the values and moves on to the next point.

To obtain the Mean Euclidean Distance, it simply sums all the distances to the nearest neighbor and divides by the number of points at the end of iterating on all points of the map.

Finally, the program outputs the metrics, and it is essential to know that the metrics of this method are directional (point A in map X being the nearest from point B in map Y does not imply that the point B in map Y is the nearest from point A in map X). Because of this, the Hausdorff distance definition will be the biggest of both metrics.

Chapter 5

Results

5.1 Trajectory Comparison

To test our methodology, the comparison of trajectories is applied to our previously described scenarios in Table 3.2. The result of this analysis is presented below.

	Velodyne		Bosch	
	MAE [m]	Max. MAE [m]	MAE [m]	Max. MAE [m]
Scenario A	0.25	0.56	0.2	0.66
Scenario B	0.61	0.98	1.25	2.68
Scenario C	0.10	0.31	0.10	0.33
Scenario D	0.09	0.21	0.11	0.22
Scenario E	1.22	3.04	2.07	5.70
Scenario F	1.20	4.66	0.88	4.35

Table 5.1: Results of the comparison of the trajectories (MAE)

The table Table 5.1 represents the results for the metrics implemented for the trajectory comparison. Both metrics are expressed in *meters* and they are used to measure the errors between the estimated trajectory by the SLAM algorithm using each LiDAR sensor with the Odometry data.

The Table 5.2 represents the result of the metric of the MSE. It is expressed in $meters^2$ and represents how close the points are to the ground truth data (Odometry data). The smaller the MSE is, the closest is the estimated trajectory to the Odometry data. MSE is a good indicator in case there are isolated points that have a large discrepancy in relation to the odometry, such that they constitute values that are too far apart. This metric gives more value to points that are further away due to the squared calculation, so the MSE gets bigger.

	MSE [m]	
	Velodyne	Bosch
Scenario A	0.08	0.05
Scenario B	0.45	1.91
Scenario C	0.01	0.01
Scenario D	0.01	0.01
Scenario E	1.75	5.26
Scenario F	1.74	0.99

Table 5.2: Results of the comparison of trajectories (MSE)

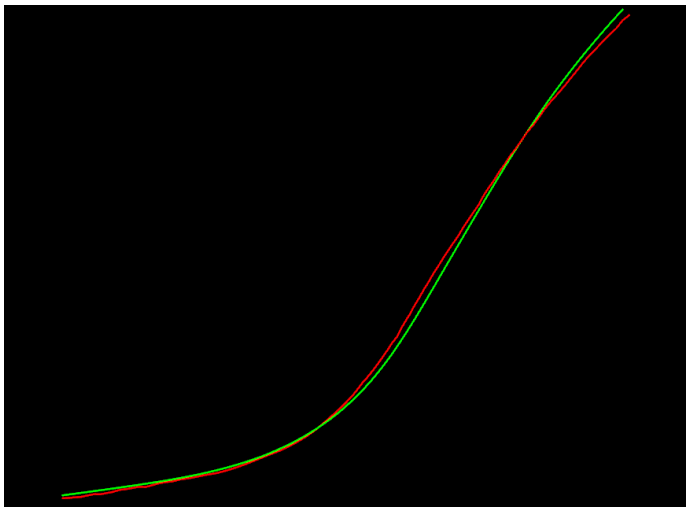

Figure 5.1: Scenario A: Trajectory comparison between the Odometry data and the estimated trajectory of the SLAM using the Velodyne sensor

If we look at Scenario A results, they are similar for each sensor, if we compare the MAE in the Table 5.1 of the SLAM results using each sensor. And if we look at Figure 5.1 and Figure 5.2 we can also see that the discrepancy between each estimated trajectory and the Odometry data is not that significant. Analyzing the results given in Table 5.2 we can observe that the MSE for Scenario A is very low, which means that the points from the estimated trajectory do not diverge significantly.

In the results of Scenario B we have some higher discrepancies. The MAE in Table 5.1 is significantly bigger than in the Scenario A. If we compare the MAE in Scenario B we can see that the performance of the SLAM algorithm using the Bosch LiDAR is worst than using the Velodyne sensor. The error using the Bosch LiDAR is about double that of using the Velodyne sensor.

If we take a look at Figure 5.3 and Figure 5.4, we can confirm that with the Bosch LiDAR, the SLAM generates a worst estimation of the trajectory, with larger deviation concerning the odometry data. As we can see in the Table 3.2, both Scenario A and Scenario B have about the same distance travelled, about the same average velocity and they are of the same type of scenario

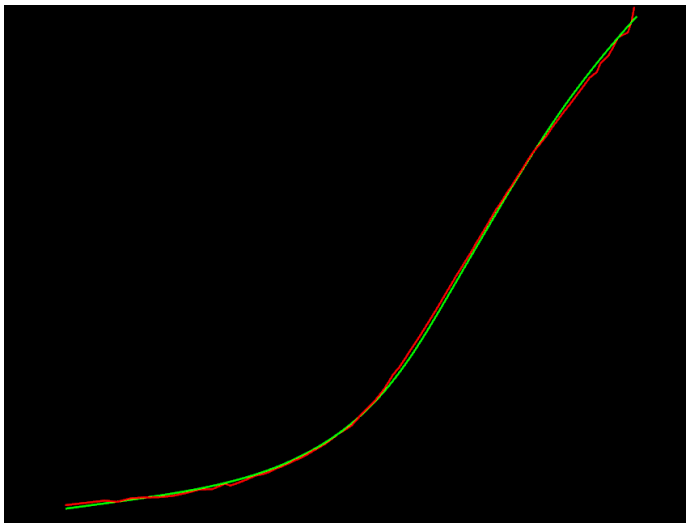

Figure 5.2: Scenario A: Trajectory comparison between the Odometry data and the estimated trajectory of the SLAM using the Bosch sensor

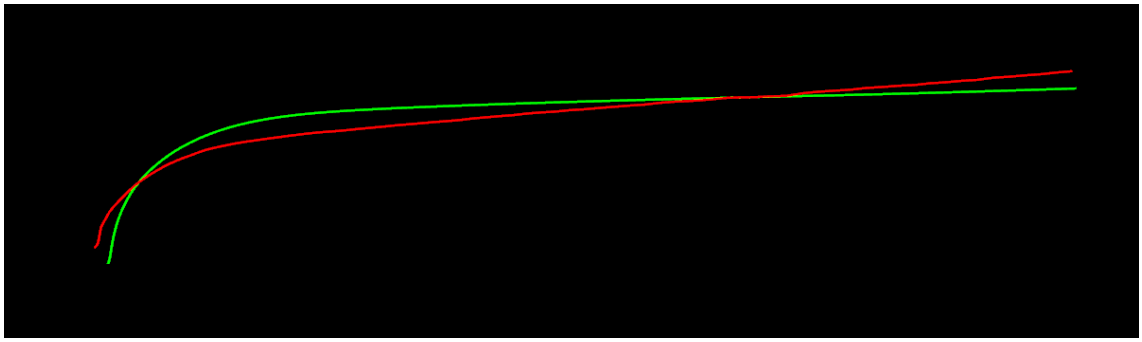

Figure 5.3: Scenario B: Trajectory comparison between the Odometry data and the estimated trajectory of the SLAM using the Velodyne sensor

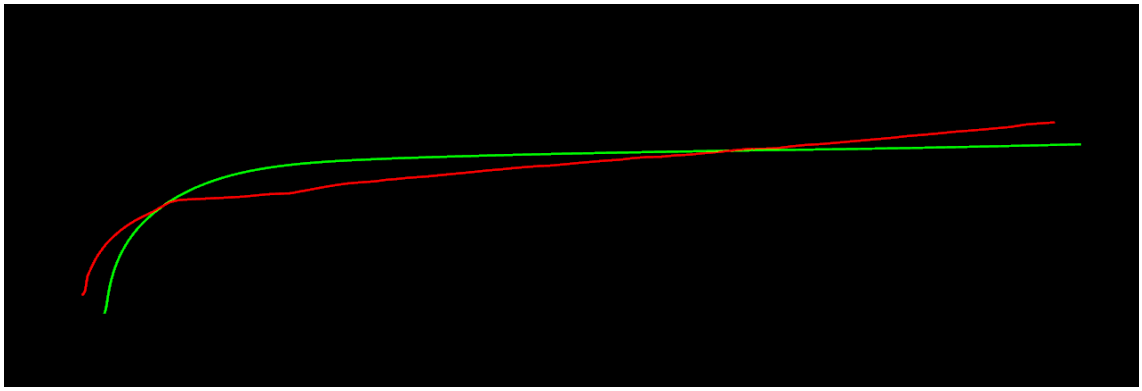

Figure 5.4: Scenario B: Trajectory comparison between the Odometry data and the estimated trajectory of the SLAM using the Bosch sensor

(urban).

A characteristic that could potentially influence the results of Scenario B is that in this scenario, multiple cars pass by in the opposite direction. Another factor influencing these results could be the rapid turn to the car's right.

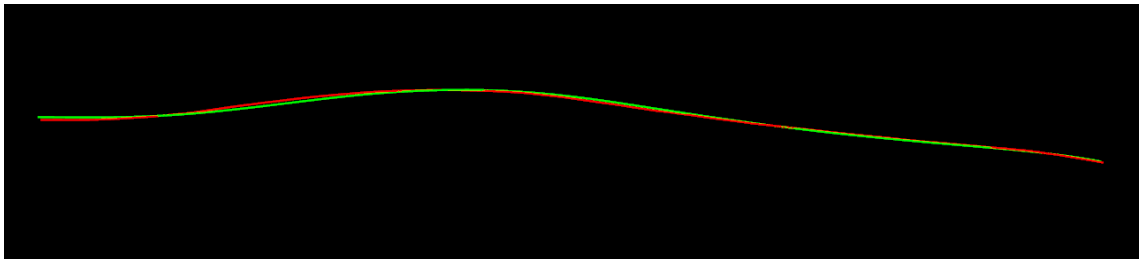

Figure 5.5: Scenario C: Trajectory comparison between the Odometry data and the estimated trajectory of the SLAM using the Velodyne sensor

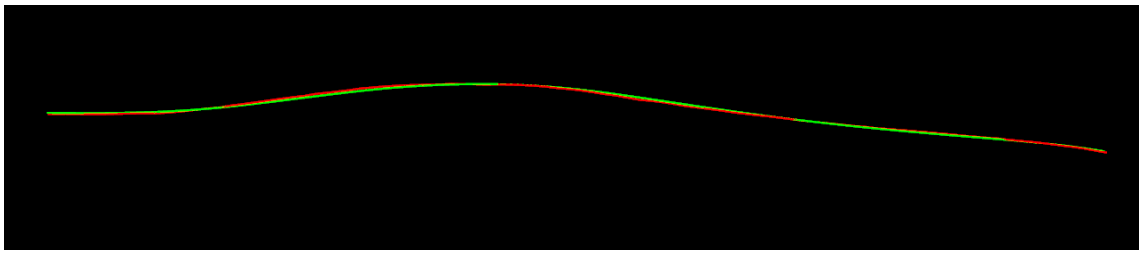

Figure 5.6: Scenario C: Trajectory comparison between the Odometry data and the estimated trajectory of the SLAM using the Bosch sensor

In the results of Scenario C, the MAE values in Table 5.1 are significantly low for the SLAM algorithm with each sensor. The distance travelled and the average velocity of this scenario is similar to the Scenario B, having both scenarios dynamic objects and being both of the same type (urban).

Observing the data from Table 5.2, the MSE is very small, which means that the discrepancy from the ground truth is very small, almost nonexistent. This could be considered a good result of the SLAM algorithm, and if we take a look at Figure 5.5 and Figure 5.6, both estimated trajectories are almost coincident, which confirms the results of the MAE and MSE metrics.

In Scenario D, the results are similar to the results in Scenario C. The MAE values in Table 5.1 are about the same in both scenarios and the MSE in Table 5.2 are equal. Despite having about double the distance travelled and average velocity of Scenario C and not having dynamic objects, the performance of the SLAM algorithm in the trajectory estimation is about the same in both scenarios. We can also observe that in Figure 5.7 and Figure 5.8 are almost coincident, which exactly demonstrates the results of the metrics.

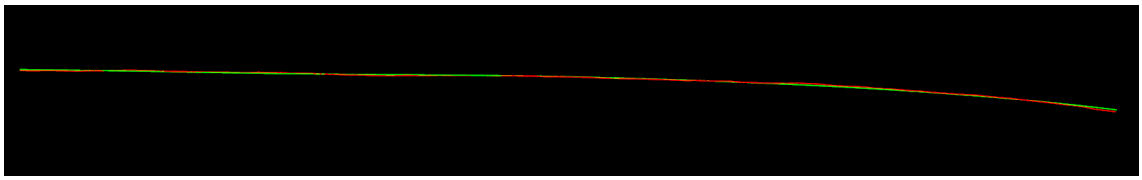

Figure 5.7: Scenario D: Trajectory comparison between the Odometry data and the estimated trajectory of the SLAM using the Velodyne sensor

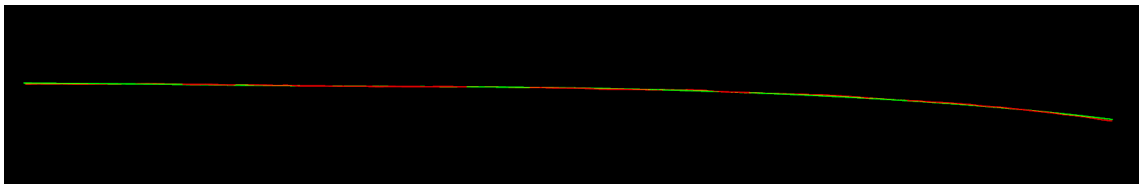

Figure 5.8: Scenario D: Trajectory comparison between the Odometry data and the estimated trajectory of the SLAM using the Bosch sensor

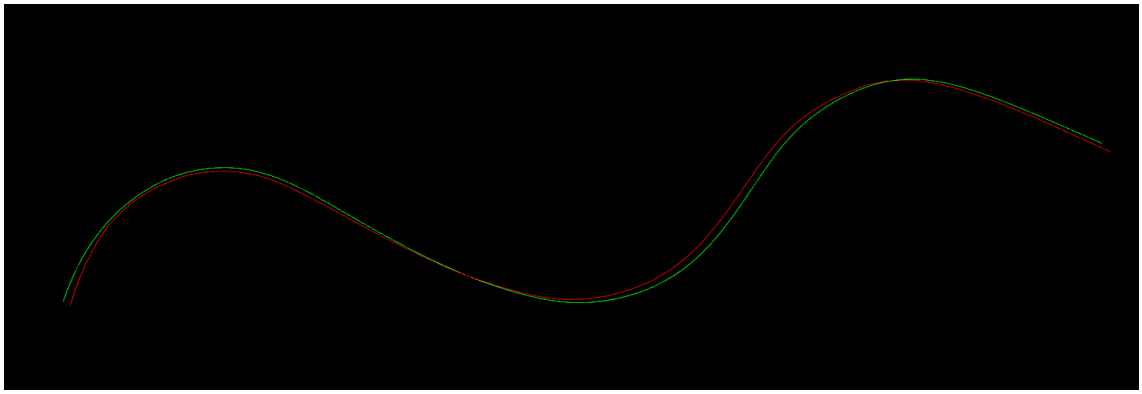

Figure 5.9: Scenario E: Trajectory comparison between the Odometry data and the estimated trajectory of the SLAM using the Velodyne sensor

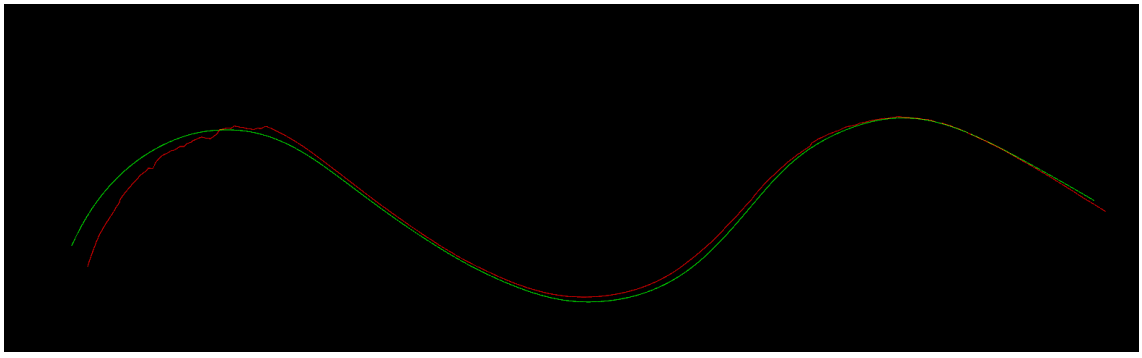

Figure 5.10: Scenario E: Trajectory comparison between the Odometry data and the estimated trajectory of the SLAM using the Bosch sensor

We have some bigger errors in the results of Scenario E. Comparing the results of the SLAM algorithm using the Velodyne sensor with those obtained with the Bosch sensor data, this last one has a worst performance, with the MAE being almost double that of the Velodyne sensor. If we analyze Figure 5.9 and Figure 5.10 we can clearly see the discrepancies between the Odometry data and estimated trajectory of the SLAM algorithm using each LiDAR sensor.

This scenario is the longest of our sample (both in time and distance), with three direction changes. This could be the cause for the MAE being so high, but other factors could be the reason for these errors. In relation to Scenario D for example, Scenario E has dynamic objects and more than double of the distance travelled. These factors can be one of the reasons for this performance being worst.

In this scenario, one thing that was noted was the presence of crosstalk [7], a phenomenon in which the LiDAR sensor accepts a pulse or an echo that is not from its own emitter, an incorrect range measurement and an incorrect 3D point are produced. This could also be a factor that influences the results and the performance of the SLAM algorithm, since in each campaign, both LiDAR sensors are simultaneously collecting data. However, crosstalk was only detected in the Bosch sensor.

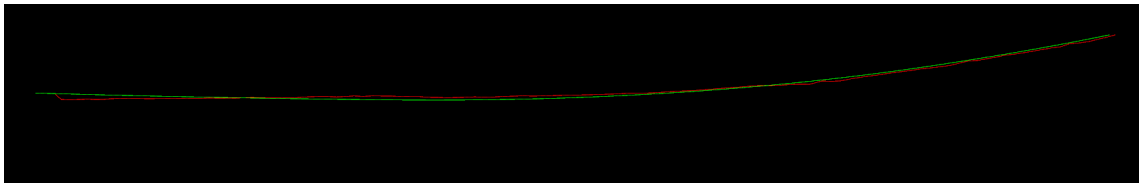

Figure 5.11: Scenario F: Trajectory comparison between the Odometry data and the estimated trajectory of the SLAM using the Velodyne sensor

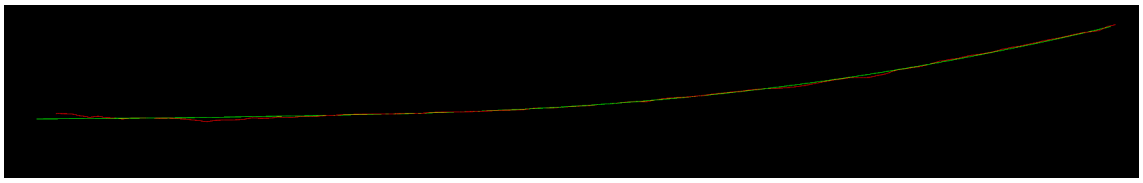

Figure 5.12: Scenario F: Trajectory comparison between the Odometry data and the estimated trajectory of the SLAM using the Bosch sensor

In Scenario F we can see that the MAE in Table 5.1 is about the same as the Scenario E for the Velodyne sensor and significantly smaller with the Bosch sensor. If we take a look at Figure 5.9 and Figure 5.10 we can see that we have discrepancies in both images, with the errors in Figure 5.9 being more significant. This is something that verifies our metrics used in the trajectory comparison.

In relation to Scenario E, they both have the same distance travelled, the same scenario type

(highway) and both have dynamic objects. However, Scenario F has more than double of average velocity. This does not seem to be a factor that influences the results.

If we take a look at Figure 3.5 and Figure 3.6, we can see that the route is more complex in Scenario E, with three curves. In this scenario, the SLAM algorithm seems to have about the same performance as in Scenario E using the Velodyne LiDAR sensor, but a better performance when using the Bosch LiDAR sensor

As we can see, the metrics presented in Table 5.1 and Table 5.2 represent very well the performance of the SLAM algorithm when we compare it with the images representing the trajectory comparison between the Odometry data and the estimated trajectory of the SLAM algorithm using each sensor.

5.2 Map Comparison

In this section are shown the results of comparing the map generated by the SLAM algorithm. As in the previous section, this comparison was applied to the scenarios described in Table 3.2.

	Hausdorff Distance [m]	Mean Euclidean Distance [m]
Scenario A	7.75	0.04
Scenario B	8.35	0.48
Scenario C	47.34	0.21
Scenario D	7.02	0.55
Scenario E	23.58	9.39
Scenario F	7.91	4.61

Table 5.3: Results of the comparison of the maps Bosch→Velodyne

All the values expressed in Table 5.3 are in *meters*. The notation Bosch→Velodyne means that the metric was applied such that in each iteration, for each point of the map generated by the SLAM algorithm using the Bosch LiDAR sensor, we find the nearest neighbor in the map generated by the SLAM algorithm using the Velodyne sensor. As mentioned before, these are directional metrics, such that point A in map X being the nearest from point B in map Y does not imply that point B in map Y is the nearest from point A in map X. Therefore, the values for each Hausdorff distance in each direction are different.

In this analysis, we did not consider the Hausdorff distance because it does not represent the results of the performance of the SLAM algorithm in the generation of the maps effectively. This is because the sensors have different specifications and different densities of points. However, we mainly used the mean Euclidean distance as a metric that resulted in an excellent metric to analyze the performance of the SLAM in the generation of the maps.

The figures bellow represent the output maps of the SLAM algorithm for each Scenario, using both LiDAR sensors. The map generated using the Velodyne LiDAR sensor is represented by the green color and the map generated using the Bosch sensor is represented by the red color.

If we take a look at Figure 5.13 we can observe that the generated map by the SLAM using the Bosch LiDAR sensor coincides with the generated map by the SLAM using the Velodyne sensor for Scenario A. This corroborates the result of the mean euclidean distance, presented in Table 5.3, where it only has a distance of 0.04 meters. This is a very insignificant value, and we can conclude that this value represents what we see when comparing visually the maps generated by the SLAM algorithm. As seen in the previous section in trajectory comparison, the SLAM presented good results when estimating a trajectory for Scenario A.

Observing Figure 5.14, the generated map by the SLAM algorithm using the Bosch LiDAR sensor pretty much corresponds to the generated map by the SLAM using the Velodyne sensor for Scenario B. However, we can see some small imprecision in the lower part of the figure, but

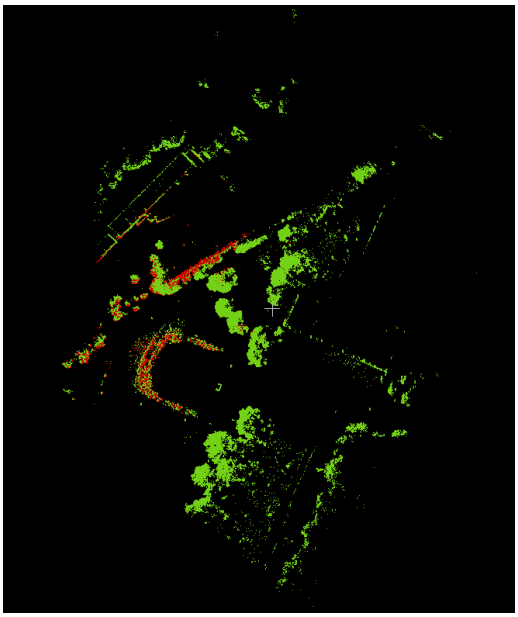

Figure 5.13: Scenario A: Output maps from the SLAM algorithm using the Velodyne sensor (green) and the Bosch sensor (red)

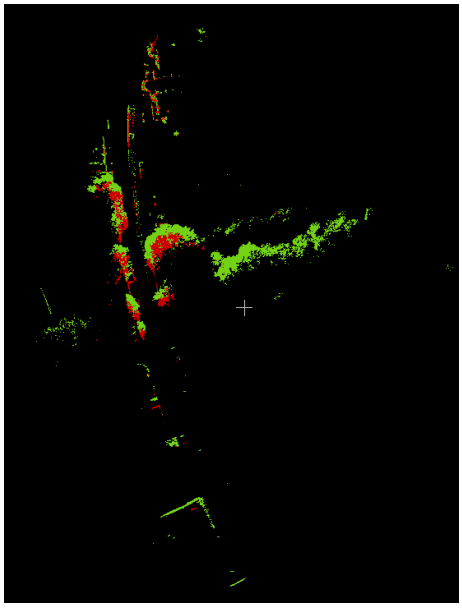

Figure 5.14: Scenario B: Output maps from the SLAM algorithm using the Velodyne sensor (green) and the Bosch sensor (red)

nothing significant. This is reflected in Table 5.3 in the mean euclidean distance. It is slighly higher in Scenario B than in Scenario A. But this is not a compromising deviation, since it only corresponds to a mean euclidean distance to the map generated with the Velodyne of 0.48 meters. Once again, this metric resulted in a good representation of the results, that could be corroboreed with the Figure 5.14

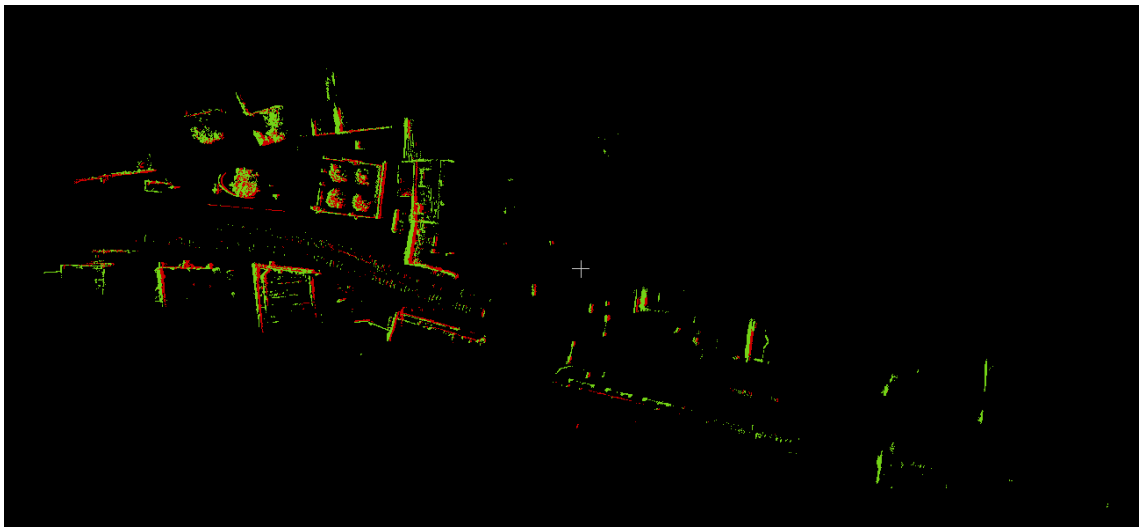

Figure 5.15: Scenario C: Output maps from the SLAM algorithm using the Velodyne sensor (green) and the Bosch sensor (red)

In Figure 5.15, we have about the same result as seen in Figure 5.14. Some imprecision can be observed, less than in Figure 5.14, but nothing compromising, with a mean euclidean distance of only 0.21 meters. This value is easily confirmed by looking at Figure 5.15

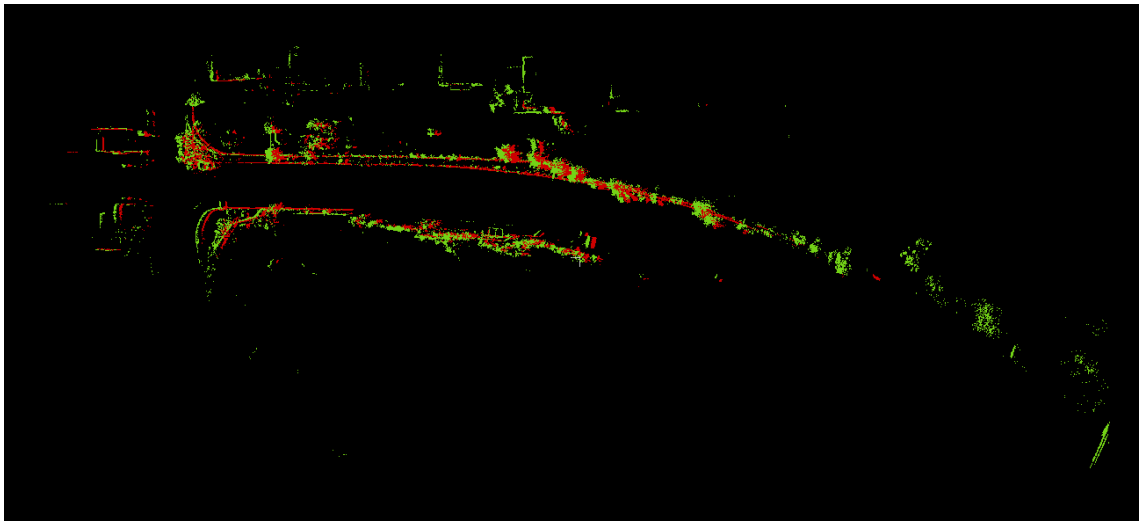

Figure 5.16: Scenario D: Output maps from the SLAM algorithm using the Velodyne sensor (green) and the Bosch sensor (red)

In Figure 5.16, the generated map by the SLAM algorithm using the Bosch LiDAR sensor coincides with the generated map by the SLAM algorithm using the Velodyne sensor for Scenario D. Nonetheless, we can see some deviation from both maps, but nothing compromising. The SLAM algorithm keeps having a good performance and the mean euclidean distance still provides a good feedback of how consistent both maps are. In this scenario, the mean euclidean distance

has a value of only 0.55 meters.

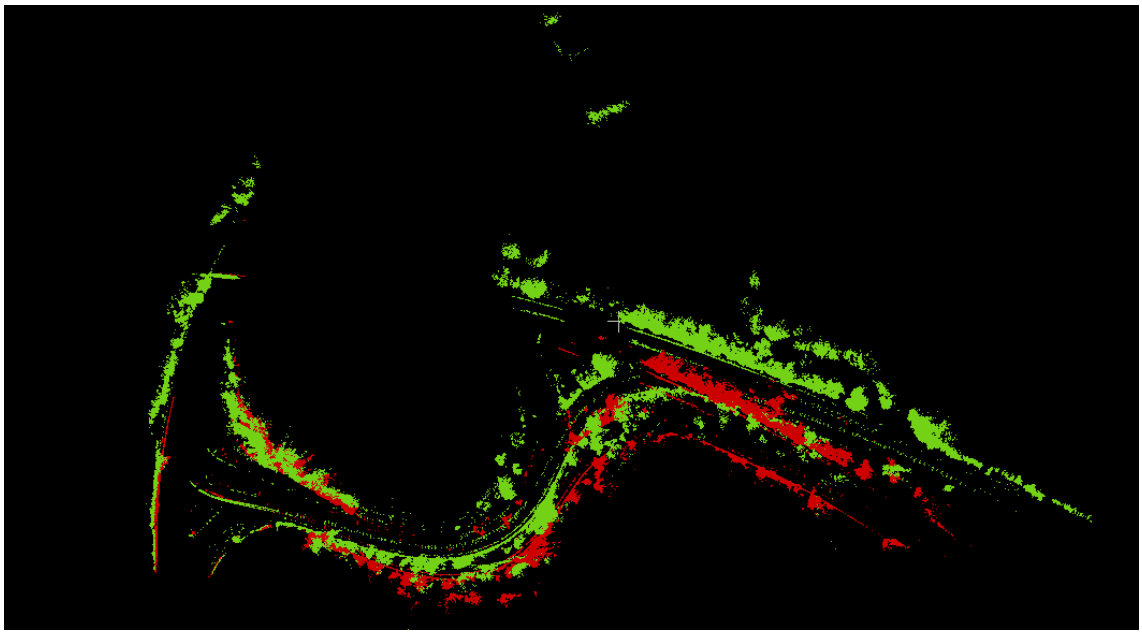

Figure 5.17: Scenario E: Output maps from the SLAM algorithm using the Velodyne sensor (green) and the Bosch sensor (red)

In Figure 5.17, the generated map by the SLAM algorithm using the Bosch LiDAR sensor does not match the generated map by the SLAM algorithm using the Velodyne sensor for Scenario E. As we can observe, the performance of the SLAM algorithm was significantly lower, with a mean euclidean distance of 9.39 meters. This is a significant value, because a imprecision of such magnitude is really bad performance. For terms of comparison, the width of a carriageway is about 3.5 meters, and the mean euclidean distance in this scenario is of about two and a half times bigger than a carriageway. As we already saw in the trajectory comparison, the SLAM algorithm in this scenario did not have a good performance either. The cause to this problem could be the accumulation of errors or the fact that this scenario is the longest of our sample (both in time and distance), with three direction changes. In relation to Scenario D for example, Scenario E has dynamic objects and more than double of the distance travelled. These factors can be one of the reasons for this performance being worst.

As mentioned before in the trajectory comparison, in this scenario, one thing that was noted was the presence of crosstalk [7], that could also be a factor that influences the results and the performance of the SLAM algorithm, since in each campaign, both LiDAR sensors are simultaneously collecting data.

In Figure 5.18, the generated map by the SLAM algorithm using the Bosch LiDAR sensor does not entirely match the generated map by the SLAM algorithm using the Velodyne sensor for Scenario F. In this scenario we have a mean euclidean distance of 4.61 meters, a high and

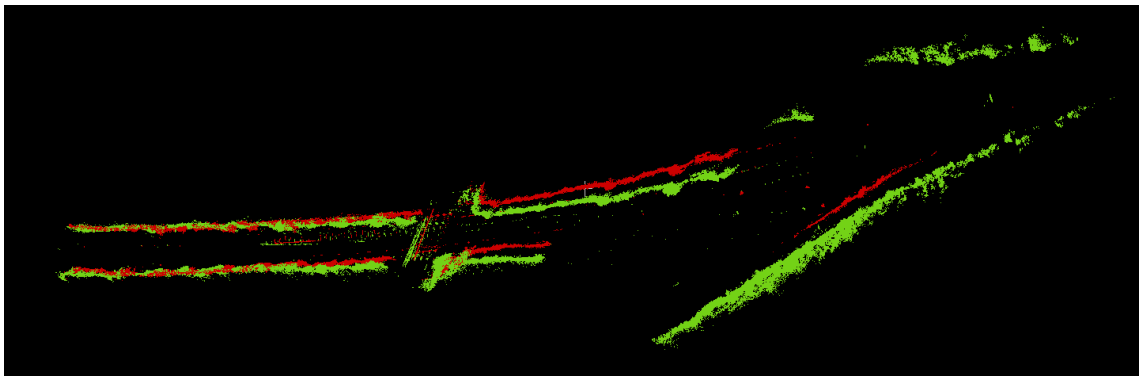

Figure 5.18: Scenario F: Output maps from the SLAM algorithm using the Velodyne sensor (green) and the Bosch sensor (red)

significant value. If we look at Figure 5.18 we can see the discrepancy between the generated maps. This could have to do with the same causes presented in the previous Scenario E. It could be a accumulation of errors, the fact that has the longest distance travelled of 261 meters, the highest average velocity of about 21.8 meters/seconds (78.48 kilometers/hour).

Chapter 6

Conclusion

In this work, we addressed the problem of the validation of automated driving, and this work contributed by creating a methodology to evaluate one of the ground truths in the perception layer, the SLAM algorithm. The metrics proposed in this work represent the SLAM algorithm's performance very well.

In the trajectory comparison, the Mean Absolute Error (MAE) and the Mean Squared Error represented with outstanding accuracy the scenarios in which the SLAM algorithm had an excellent performance in estimating the trajectory. This could be verified by analyzing the images containing both the Odometry data and the estimated trajectory by the SLAM algorithm using each LiDAR sensor. The MAE represents very well how the average of the points coincides with the Odometry data. At the same time, the MSE proved to be a good indicator in case isolated points have a large discrepancy to the Odometry.

In the map comparison, the Mean Euclidean Distance turned out to be an excellent Key Performance Indicator (KPI). It could indicate the scenarios in which the SLAM algorithm had the worst and better performance generating the map using each sensor using KPIs. It could detect which scenarios had errors in the maps, which could be verified if we observe the correspondent figures of each scenario, containing both maps generated by the SLAM using the Velodyne LiDAR sensor and the Bosch LiDAR sensor.

Through this methodology, which provides meaningful key performance indicators, the developers can have a much more robust validation module due to more robust ground truths and ensure the quality of the ground truth in the perception layer in the extraction of static objects.

References

[1] What is Lidar? Learn How Lidar Works.

[2] Hriday Bavle, Paloma De La Puente, Jonathan P. How, and Pascual Campoy. VPS-SLAM: Visual Planar Semantic SLAM for Aerial Robotic Systems. *IEEE Access*, 8:60704–60718, 2020. Conference Name: IEEE Access.

[3] César Debeunne and Damien Vivet. A Review of Visual-LiDAR Fusion based Simultaneous Localization and Mapping. *Sensors*, 20(7):2068, April 2020.

[4] David Droeschel and Sven Behnke. Efficient Continuous-Time SLAM for 3D Lidar-Based Online Mapping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5000–5007, May 2018. ISSN: 2577-087X.

[5] Christine Evers, Alastair H. Moore, and Patrick A. Naylor. Acoustic simultaneous localization and mapping (A-SLAM) of a moving microphone array and its surrounding speakers. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6–10, March 2016. ISSN: 2379-190X.

[6] Anton Filatov, Artyom Filatov, Kirill Krinkin, Baian Chen, and Diana Molodan. 2D SLAM quality evaluation methods. In *2017 21st Conference of Open Innovations Association (FRUCT)*, pages 120–126, Helsinki, November 2017. IEEE.

[7] Marcus Hebel, Marcus Hammer, Michael Arens, and Axel L. Diehm. Mitigation of crosstalk effects in multi-LiDAR configurations. In Gary Kamerman and Ove Steinvall, editors, *Electro-Optical Remote Sensing XII*, page 3, Berlin, Germany, October 2018. SPIE.

[8] Margaret E. Jefferies and Wai K. Yeap. *Robotics and cognitive approaches to spatial mapping*. Number v. 38 in Springer tracts in advanced robotics. Springer, Berlin ; New York, 2008. OCLC: ocn173721135.

[9] W. Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 34(5):827–828, September 1978. Number: 5 Publisher: International Union of Crystallography.

[10] Marco Karrer, Patrik Schmuck, and Margarita Chli. CVI-SLAM—Collaborative Visual-Inertial SLAM. *IEEE Robotics and Automation Letters*, 3(4):2762–2769, October 2018. Conference Name: IEEE Robotics and Automation Letters.

[11] Amey Kasar. Benchmarking and Comparing Popular Visual SLAM Algorithms, December 2018. arXiv:1811.09895 [cs].

[12] Gerhard Kurz, Matthias Holoch, and Peter Biber. Geometry-based Graph Pruning for Lifelong SLAM. In *arXiv:2110.01286 [cs]*, October 2021. arXiv: 2110.01286.

[13] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27(4):387–407, November 2009.

[14] Aarian Marshall. The Uber Crash Won't Be the Last Shocking Self-Driving Death. *Wired*. Section: tags.

[15] Jack Stewart. Why Tesla's Autopilot Can't See a Stopped Firetruck. *Wired*. Section: tags.

[16] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, April 1991.

[17] Rauf Yagfarov, Mikhail Ivanou, and Ilya Afanasyev. Map Comparison of Lidar-based 2D SLAM Algorithms Using Precise Ground Truth. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1979–1983, November 2018.

[18] Zichao Zhang and Davide Scaramuzza. A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251, Madrid, October 2018. IEEE.