Evualating SLAM in an urban dynamic environment

Author:
S.A. van Schouwenburg

Supervisors:
J.F.P. Kooij
T.M. Hehn

Technische Universiteit Delft

TU Delft

TU Delft

Evaluating SLAM in an urban dynamic environment

by

S.A. van Schouwenburg

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday August 26, 2019 at 14:30 AM.

Student number:	4149394	
Project duration:	Sept 25, 2018 – Aug 26, 2019	
Thesis committee:	Dr. J. F. P. Kooij,	TU Delft, supervisor
	T. M. Hehn, MSc.	TU Delft, daily supervisor
	Prof. dr. D.M. Gavrilla,	TU Delft
	Dr. C. H. Corbato,	TU Delft

This thesis is confidential and cannot be made public until August 26, 2019.

TU Delft

Abstract

Simultaneous Localization And Mapping (SLAM) algorithms provide accurate localization for autonomous vehicles and provide essential information for the path planning module. However, SLAM algorithms assume a static environment in order to estimate a location. This assumption influences the pose estimation in dynamic urban environments. The impact of this assumption on day-to-day scenarios of an intelligent vehicle is unknown. A deeper understanding on the effect of dynamic scenarios in an urban environment could lead to simple and robust solutions for SLAM algorithms in intelligent vehicles. The objective of this research is to develop a methodology that isolates the effect of an urban dynamic environment on the performance of a SLAM algorithm. This requires constant environment conditions including constant weather conditions, lighting conditions and identical trajectories over time. The methodology is tested with a stereo feature based V-SLAM algorithm called ORB SLAM [19], which illustrates the in-depth analysis that is possible with this experiment. The main research question is: *How does a dynamic urban environment influence the pose estimation accuracy of stereo ORB SLAM?*

Two specific dynamic scenarios are designed to represent a dynamic urban environment: driving behind another vehicle and vehicles approaching on the other side of the road. On these scenarios, an in-depth analysis of ORB SLAM is performed to observe how the algorithm's design influences the robustness to a dynamic environment. Functions within the algorithm are bypassed to analyze the effect on the performance. Specifically, the place recognition function and map point filtering function are bypassed. The analysis proofs which functions assist in the overall robustness to a dynamic environment. Moreover, an analysis is performed of the algorithm in localization mode to research the effect of utilizing maps that were created under different conditions. The knowledge gained from the full analysis can be utilized to improve other V-SLAM algorithms.

The experiment is performed in CARLA [6], an open source simulator. CARLA provides an elaborate sensor suite which support multiple camera setups and LIDAR sensors. Furthermore, the simulator provides free maps which represent realistic urban environments and allows for easy and accurate access to the ground truth position. A setup is designed with the simulator that allows complete isolation of the effect of a dynamic environment. The setup allows full control of lighting conditions, weather conditions and allows identical trajectories over time in different dynamic scenarios. Each scenario is simulated over several different trajectories in which the camera images are converted to rosbags. Each variation of the ORB SLAM algorithm is tested on the produced rosbags. The resulting pose estimations in dynamic conditions are compared to the pose estimations made during static conditions to analyze the effect of dynamic scenarios on the performance of the algorithm.

The method successfully isolated the effect of a dynamic environment on the performance of stereo ORB SLAM. It allows for a detailed analysis which aids in finding the source of performance differences. In general, stereo ORB SLAM displays robust behavior to a dynamic environment. The experiment shows that the algorithm is sensitive to false relocalization when the stereo camera setup is driving 10 meters behind another vehicle for a long period of time. During these conditions, ORB SLAM cannot provide accurate pose estimations even when the place recognition module is deactivated. Furthermore, the map point filtering does increase the robustness in certain dynamic scenarios. Finally, the data suggests that utilizing maps created in different conditions does influence the pose estimation in localization mode. However, more data is needed to confirm these results.

The methodology has proven its value for in depth analysis of robustness to an urban dynamic environment for a SLAM algorithm. This experiment is not limited to ORB SLAM but could be utilized for other monocular and stereo V-SLAM methods, as well as LIDAR based methods. New solutions can be developed to increase robustness to a dynamic environment and tested on the same rosbags. This methodology could be an important tool for the development SLAM algorithms for intelligent vehicles.

Acknowledgements

My time as a student has been an eight year journey in which I had so much fun and learned so much. This thesis marks the end of my time as a student. But before I thank the people that have contributed in my thesis, I want to thank my mom and dad that have fully supported my throughout this journey. I started a bachelor Aerospace Engineering even-though I was never talented in math or physics. I was allowed to join DUT Racing even-though I only had 40 ECTS in my first year. I could even pause my studies for 1 year to become a full time operations manager for Formula Student. They have always fully supported me in every choice I made and I am so grateful for that. Thank you mom and dad! I would never be where I am right now without your support.

I would like thank my supervisors, Julian Kooij and Thomas Hehn, for their great support throughout the year. I also would like to thank the other members of the exam committee Prof. Dr. Dariu Gavrilla and Dr. Carlos Corbato for taking the time to assess my thesis.
I want to thank the developers of CARLA for making their research platform available. If you ever want to use CARLA, the platform has an easy to use python API and great support on Github. Also special thanks to the developers of ORB SLAM, Raúl Mur-Artal and Juan D. Tardós, for making their code open source.

I want to give a shout-out to my friends and colleagues that have helped in the final stages. Thanks to Geert Sprong, Dirk van der Valk and Jiaao Dong for proof reading my report. Also thanks to Tom Elands and studio Rendier for providing the cover art.
Finally I want to thank especially Thomas, who took the time to asses my chaotic writing for an entire year. Thanks for pointing out some embarrassing mistakes which were easy to spot, but let it be hindsight. Thank you for finding detailed spelling mistakes as well as commenting on the overall flow of the report. Making the final thesis not only complete, but whole.

S.A. van Schouwenburg
Delft, August 2019

Contents

1 Introduction

For a safe journey in an autonomous vehicle, it is essential that the vehicle can perform accurate localization. The vehicle needs to know its position within its environment, so it is able to plan a safe path to follow. This means the vehicle should have access to an accurate environment representation, as well as an accurate position estimation. An incomplete environment representation could lead to a planned trajectory that crashes into undetected objects. A perfect environment representation but an inaccurate position estimation leads to collision with detected objects, since the vehicle expects to be at a different location. A big challenge for fully autonomous vehicles would be solved when accurate localization can be performed.

Current technology is not able to provide reliable and accurate localization for vehicles. Advanced localization systems use Global Navigation Satellite Systems (GNSS) and Inertial Navigation Systems (INS) that can provide a pose estimation to the vehicle. However, INS can only provide an accurate pose estimation over a short period of time as the technique accumulates error. This technique, called dead reckoning, utilizes Inertial Measurement Units (IMUs) to measure the vehicle's accelerations. The accelerations are integrated over time to provide position estimations. Each IMU measurement contains a small error, which accumulates over time and results in inaccurate localization for large time spans. GNSS use Global Positioning System (GPS) satellites to provide a location estimation for the vehicle, which is not accurate enough to solely use for autonomous vehicles. The technology needs to receive a signal from multiple satellites to provide a location estimate. This signal can easily be blocked by for example buildings, making it an unreliable method for fully automated driving in urban environments. New technology is needed that allows autonomous vehicles to localize itself reliably and accurately.

Simultaneous Localization And Mapping (SLAM) algorithms could provide a reliable position estimation for intelligent vehicles. SLAM algorithms use sensors, like cameras or a LIDAR system, which map the environment and simultaneously provide the vehicle's location within this map. The algorithm is able to recognize and match the sensor data over time. For example, ORB SLAM [19] is able to recognize and match corners from an image under different viewing angles. The matched data provides constraints for which the algorithm can optimize an estimated position [13]. After the position is estimated, the detected sensor data can be plotted on the map which the algorithm maintains. Furthermore, the algorithm is able to recognize previously visited locations and utilize these as additional constraints in the position estimation. This method, called loop closure, corrects for the accumulated error; making it a better alternative than dead reckoning methods. SLAM plays a vital role in pose estimation for intelligent vehicles since its environment representation and localizing abilities are accurate enough to be utilized for automated driving.

The dynamic conditions that an intelligent vehicle operates in could heavily influence the position estimation, since SLAM frameworks assume that the environment is static. How much the dynamic environment influences SLAM is unknown. However, solutions exist to enforce the static environment assumption. Some have developed algorithms that track and filter dynamic objects within the SLAM framework (e.g. [29], [23]) or used masks over a V-SLAM framework to filter out features that describe vehicles [15]. Note that these solutions work under very specific conditions. More than enough research is done to find a solution, but little effort is done to understand the problem. To the best of my knowledge, no research exists that measures the effect of different dynamic situations on the pose estimation of SLAM algorithms. It could be that only specific dynamic scenarios significantly influence the pose estimation. Gaining knowledge in the effect of dynamic scenarios on SLAM would aid in creating simpler and more robust algorithms for SLAM in intelligent

vehicles.

The objective of this research is to develop a methodology that isolates the effect of a dynamic environment on the performance of a SLAM algorithm. This research will focus on the performance of stereo ORB SLAM [19] in a dynamic environment to illustrate the depth of the analysis possible with this methodology. ORB SLAM is an open source, state of the art and feature based V-SLAM method. It is a versatile algorithm that handles monocular, RGB-D and stereo camera setups. ORB SLAM is chosen since the complete code is available for public use. Although many papers have been published that have better performing SLAM algorithms [12], none of the state of the art algorithms are open source. Furthermore, ORB SLAM utilizes some unique filter methods to establish reliable keypoints which are then used to optimize a pose estimation [20]. This filter mechanism could make the algorithm more robust in certain dynamic scenarios. If this is the case, the solution could be implemented in other SLAM algorithms to ensure more robustness to a moving environment. Finally, the algorithm provides a localization mode which re-uses the map that was created, and localizes the pose real time without altering the map. This feature allows to research how the pose estimation is influenced when the map is made in different conditions. This is essential research for mapping companies which could provide detailed maps that will be utilized by intelligent vehicles for localization.

The challenge of this research is to design an experiment that directly quantifies the effect of a dynamic environment to the pose estimation of a SLAM algorithm. It is challenging since many conditions heavily influence the results of SLAM. For example, the vehicle should drive exactly the same trajectory in dynamic and static conditions to isolate the effect of a dynamic environment. If there are deviations in the trajectory, it would be difficult to trace the cause of performance differences. Is the difference in pose estimation caused by the trajectory deviation or caused by a moving environment? Furthermore, the experiments should be performed at the same time of day and under the same weather conditions, since V-SLAM algorithms are influenced by shadows and clouds [15]. Isolating the effect of a dynamic environment is therefore nearly impossible to perform in real life, however a simulator provides a solution. Doing the experiment in a simulation allows to drive exactly the same trajectory under different dynamic conditions and enables full control of the environment. The simulation is performed in CARLA [6], an open source simulation platform designed to assess autonomous driving algorithms. CARLA offers an elaborate sensor suite, allowing multiple camera set-ups, depth images and LIDAR placement. The multiple camera setup feature allows ORB SLAM to be evaluated on a stereo camera setup. CARLA is an unique simulator that is able to create an environment that isolates the effect of dynamic scenarios on the performance of stereo ORB SLAM.

This research aims to gain knowledge on the effect of a dynamic environment on a SLAM algorithm, by analyzing the performance of ORB SLAM in different dynamic scenarios. A deeper understanding of the problem could result into more robust and simpler solutions for SLAM in an automotive environment. This report provides an unique experiment setup which isolates the effect of moving vehicles on the performance of stereo ORB SLAM by performing it in a simulator. Furthermore, the effect of the filtering mechanism within ORB SLAM is measured to see whether this method adds robustness to certain dynamic scenarios. Finally, the effect of localization mode on the pose estimation is researched by utilizing maps that that are made in different conditions. The gained knowledge aids the development of SLAM algorithms in the automotive industry and increase the localization accuracy of autonomous vehicles.

1.1. Research questions

There needs to be a better understanding on the effect of a dynamic environment on the pose estimation of SLAM to develop more robust algorithms for intelligent vehicles. SLAM algorithms assume that the environment is static, which is not true for an autonomous vehicle. The central question is how big is the influence of this assumption? Is any movement in the environment problematic, or are only certain situations a problem? A better understanding on the effect of the static environment assumptions aids the development of SLAM algorithms for automotive applications. This research will develop a methodology that isolates the effect of a dynamic environment on stereo ORB SLAM [19]. Stereo vision is the likely sensor setup for V-SLAM algorithms in the automotive industry, since the setup allows for the most accurate depth estimations. This research will focus on dynamic scenarios in an urban environment. The main research question is:

How does a dynamic urban environment influence the pose estimation accuracy of stereo ORB SLAM?

Hypothesis: It is expected that the pose estimation will become less accurate as the amount of dynamic features in the image increases. It is likely that the amount of dynamic features increases when more pixels are

moving in the image. This makes the static environment assumption less valid. Hence, the pose estimation would become less accurate.

The dynamic urban environment defined in the main research question shall be divided into a set of dynamic scenarios. These scenarios will focus on interactions with other vehicles, as oppose to pedestrians. ORB SLAM utilizes corner detection for keypoint selection. It is more likely that ORB SLAM selects vehicles as keypoints than pedestrians, since vehicles have a larger surface area and have more corner features. This research will focus on two scenarios: driving behind another vehicle and driving while other vehicles are driving on the opposing lane. The sub research questions are:

1 *How does driving behind another vehicle influence the pose estimation accuracy of stereo ORB SLAM?*

2 *How do vehicles driving on the opposite lane influence the pose estimation accuracy of stereo ORB SLAM?*

Hypothesis: It is expected that, performing SLAM while driving behind another vehicle, is the most likely scenario where the algorithm will have the worst pose estimation performance. A large percentage of the image could be blocked by the vehicle in front. This would make it more difficult for the SLAM algorithm to find static key points on which to optimize a position. Vehicles driving in opposite direction are expected to be less influential. The total time span that a vehicle is in the camera image is lower and therefore the influence is expected to be lower. Still, it is expected that the overall performance is negatively influenced.

The developed experiment will not only illustrate the effect of a dynamic environment on the performance of a SLAM algorithm but can also be utilized to analyze its strength and weaknesses. The methodology allows an in-depth analysis to pinpoint which functions contribute to the robustness. Several functions of ORB SLAM are bypassed to illustrate the depth of the analysis possible within this experiment. The next set of sub-research questions define which functions are bypassed. The knowledge gained with this experiment can be utilized to improve the robustness of other V-SLAM algorithms.

Failure in ORB SLAM can be divided into two categories: tracking failure and false relocalization. Tracking failure happens when the algorithm is not able to match ORB features in the image to previously mapped ORB features. The algorithm stops the pose estimation until it is able to relocalize to a previous mapped point. Relocalization is caused by the loop closure module of ORB SLAM. It will relocate its estimated position when this module recognizes an environment which was driven before. It does this by storing the ORB features of each image and tries to match these with the current image. This function is to account for the accumulated error during a SLAM session. Note that the relocalization is purely visual based in this setup. The relocalization module will be bypassed to test the algorithm's performance without relocalization failure. A SLAM algorithm without place recognition is essentially a Visual Odometry (VO) algorithm. Therefore ORB SLAM without the place recognition module will be referred to as ORB VO. The sub-question will be:

3 *How do the dynamic scenarios influence the pose estimation accuracy of stereo ORB VO?*

Hypothesis: It is expected that ORB VO shows a performance decrease when the static environment assumption is significantly violated. Thus, when a large amount of ORB features describe a moving vehicle.

ORB SLAM inherently has a filter mechanisms that removes unreliable keypoints from its map. This map culling method could positively influence the performance of ORB SLAM in dynamic scenarios. If this is the case, other SLAM researchers could use this knowledge to improve their algorithm on robustness to dynamic environments. The sub research question is:

4 *How does the map culling method influence ORB VO's pose estimation in the dynamic scenarios?*

Hypothesis: It is expected that the method does increase the robustness to a dynamic environment. Moving features are less likely to match over three consecutive keyframes and will therefore be filtered out. This method could be already an effective dynamic object filter and additional filtering might not be needed.

A realistic scenario for localization in autonomous vehicles could be that mapping companies provide accurate maps which the vehicles utilize to localize itself in. ORB SLAM provides a localization mode that allows the user to localize themselves real-time in a previously created map. It is important to know if the conditions during the mapping process play a significant role in the pose estimations during localization mode. For example, suppose that during the mapping process, the mapping vehicle was stuck behind another vehicle. The map could contain many map points that describe the vehicle in front. When another vehicle tries to localize

itself in this map in localization mode, can it still accurately find its position even-though the conditions are different? The final sub-research question is:

5 *How do the conditions in which a map is created influence the pose estimation of ORB SLAM in localization mode?*

Hypothesis: If the algorithm is able to provide an accurate pose estimation during the mapping process, then the pose estimation in localization mode will also provide an accurate estimation. When the pose estimation during the mapping process is accurate, it means that sufficient static features are matched and mapped. These static features should match when the vehicle utilizes localization mode and add enough constraints to provide an accurate pose estimation.

2

Related work

SLAM algorithms for intelligent vehicles is a fast developing research topic with significant unexplored research areas. An aspect that needs major development is functioning SLAM in dynamic conditions, since the framework assumes a static environment. However, a methodology that isolates performance difference of SLAM algorithms caused by a moving environment does not exist. This chapter contains the related work that illustrates the difficulty of this experiment. Furthermore, it explains the design choices that were made through the literature. First the importance of this research is highlighted by discussing the lack of automotive oriented benchmarks. Second, the difficulty of this experiment is explained, the benefits of utilizing simulators is discussed and concludes with the reasoning behind the chosen simulator. Third the possible metrics that are used for SLAM oriented experiments are discussed. Finally the variety of SLAM algorithms that can be utilized for intelligent vehicles and the solutions that are researched for SLAM in dynamic environments are summarized.

2.1. SLAM benchmarks

Benchmarks are vital for the development of SLAM algorithms and provides a platform for researchers to test and compare their SLAM algorithms. In the last decade, several benchmarks have been published which have aided the development of SLAM algorithms. However, no benchmark is available that reflects the challenging conditions a vehicle encounters.

The TUM benchmark [28] was developed to evaluate RGB-D based SLAM algorithms. The benchmark contains multiple small scale indoor sequences in two different environments (a small office and an industrial hall). The benchmark contains sequences that challenges an algorithm's ability to cope with badly illuminated spaces, sensor outages and changing environments. The indoor, small scale nature of the benchmark makes it unsuitable to represent SLAM methods utilized in an automotive environment, but has significantly helped in the development of visual based SLAM methods.

The EuRoC dataset [2] could be used for stereo vision based SLAM algorithms. This benchmark was initialized for the European Robotics Challenge to assess visual-inertial SLAM algorithms and 3D reconstruction algorithms for micro aerial vehicles (MAVs). The benchmark contains several stereo-vision sequences in a large industry hall recorded with a MAV. The sequences challenges algorithms to handle dynamic flying conditions and large indoor loop detection. This benchmark is specifically designed for MAVs and is therefore unsuitable for automotive representation.

The benchmark that is specifically designed for automotive applications is the KITTI benchmark [11]. The benchmark utilizes a vehicle equipped with two stereo camera setups (grayscale and RGB) and a LIDAR system. It can therefore support monocular, stereo, LIDAR and sensor fusion based SLAM methods. The KITTI SLAM benchmark contains in total 22 sequences measured during day time in sunny conditions. The sequences included measurements in an urban environment, rural area and on a highway. The benchmark started in 2012 and has a wide variety of submissions including visual feature based SLAM methods (ORB SLAM [19]), direct visual SLAM methods (LSD SLAM [8]), LIDAR based SLAM (LOAM [31]) and sensor fusion methods (VLOAM [32]).

Currently the 25 best SLAM/Visual Odometry (VO) methods have a translational error of less than 1%. However, being the best SLAM/VO algorithm in the KITTI benchmark does not mean that the algorithm per-

forms best for the automotive industry. The benchmark does not reflect important aspects of an automotive environment, like the robustness of the algorithm to a dynamic environment and robustness to different weather conditions (see figure 2.1a). This information is vital for the development of SLAM algorithms for the automotive industry and could expose weaknesses in algorithms which are currently unknown. This research will be the first to isolate the performance of SLAM algorithms with respect to a dynamic environment. This methodology could then be utilized by other researchers to test and improve the robustness of SLAM algorithms in dynamic conditions.

(a) Example of conditions of the KITTI benchmark. The benchmark is focused on providing data from long trajectories, but provides little challenging dynamic conditions [11]

(b) Example conditions of VIPER benchmark which is based on the game grand theft auto V. The game provides photo-realistic environments and challenging conditions, but only provides single camera images. [24]

(c) Example image of Mask-SLAM, which utilizes CARLA to test monocular ORB SLAM in dynamic conditions and various weather conditions. It proofs that CARLA can be utilized in combination with ORB SLAM [15]

Figure 2.1: Camera views of various benchmarks and simulators to illustrate the environment conditions.

2.2. Simulators

Creating an experiment that isolates the effect of a dynamic environment on the pose estimation of a SLAM algorithm is challenging. The resulting pose estimation of a SLAM algorithm is susceptible to changes in its environment. For example, V-SLAM algorithms are very sensitive to lighting conditions since the position estimation is based on pixel values. Hence, different shadow positions have an influence on the resulting pose estimation. Moreover, weather conditions influence the results. Even cloud formations have an impact on the result of a SLAM algorithm [15]. Therefore very specific conditions are needed to test robustness for a dynamic environment. Furthermore, according to the KITTI benchmark [12], some SLAM algorithms have a translational error of only 1%. Consequently, the vehicle needs to drive identical trajectories under different dynamic conditions.

The experiment should be performed in a simulator to completely isolate the effect of a dynamic environment on the performance of a SLAM algorithm. The gaming industry can simulate photo-realistic environments which could be utilized for computer vision research. A simulator can allow full control of environmental aspects like weather conditions and time of day. It also has the potential to create identical trajectories in different dynamic conditions. Furthermore, a simulator can provide an accurate ground truth, which is difficult to obtain for long trajectories in real life. For example, the KITTI benchmark [11] provides a ground truth with at an accuracy between 0.1 and 0.5 meters by using GPS and GLONASS with RTK corrections when available. Moreover, a simulator allows for a cheap experiment setup. A simulator is the best chance to develop a methodology that isolates the effect of a dynamic environment.

The VIPER benchmark [24] utilizes the game "Grand Theft Auto V" to produce challenging sequences with different weather conditions and a dynamic environment (see figure 2.1b). These sequences contain ground truth for semantic instance segmentation, object detection and tracking and visual odometry. However, the platform does have significant limitations. The user cannot control the environment since it utilizes a hard coded game, making it difficult to have the right conditions for an experiment. Furthermore, the sequences are images from a single camera. The benchmark is therefore only useful for monocular SLAM algorithms. For research, the environment should be preferably fully adaptable and the platform should allow simulating sensors like multiple cameras and LIDAR. These requirements make a video game not preferable for research.

There are several simulators that have been been developed to benefit computer vision research. They

are programmed on gaming development platforms (like Unreal Engine) to provide the same photo-realistic environments that high end games have. Airsim [27] is a high fidelity simulator for drones and vehicles and provides realistic visuals and physics with a RGB camera view, depth map and object segmentation view. However, the simulator focuses on realistic drone and vehicle models. Therefore it only provides the vehicle and sensors but not the environment. The environment needs to be developed from scratch, making other simulators more suitable. Another, more user friendly simulator is Sim4CV [21]. The simulator has a simple GUI for environment creation and a communication interface with Matlab, Python and C++. Like Airsim, Sim4CV provides RGB camera images, depth images and object segmented view. Furthermore, Sim4CV can simulate multiple cameras mounted on the vehicle, allowing stereo vision SLAM evaluation. However, it does not support LIDAR simulation nor weather condition control. A simulator that does provide LIDAR simulation and weather control is CARLA (Car Learning to Act)[6]. The simulator was developed to support training, prototyping and validation of autonomous driving models, including perception and control. It is an open source project which provides a python API and multiple maps in which the vehicle can be controlled. CARLA comes with an additional ROS bridge, a platform that allows easy communication between the SLAM algorithm and the CARLA sensor data.

CARLA is versatile research simulator and it could be used as a platform for a new methodology that isolates the effect of a dynamic environment on SLAM algorithms. CARLA was developed as a platform to study the performance of autonomous driving systems [6]. The simulator provides scenarios and metrics so researchers can compare their autonomous systems to other controllers. The versatility of the platform allows it to be used for other research topics. CARLA is already an accepted tool within the research community to aid in the development of computer vision algorithms. The simulator has an object segmented view which is used to train CNN classifiers for dynamic object detection [35]. Furthermore, Mask-SLAM [15] combines a semantic segmentation mask on a monocular ORB SLAM algorithm to improve robustness for a dynamic environment and weather conditions (see figure 2.1c). The complete experiment is performed in CARLA, driving a total of 100 trajectories with each trajectory driven 50 times to evaluate the methods robustness. The paper illustrates the feasibility to evaluate SLAM algorithms robustness to dynamic environment in the CARLA simulator. The difference between this research compared to the Mask-SLAM experiment is the complete isolation of robustness of SLAM algorithms to a dynamic environment, instead of a dynamic environment and weather conditions.

2.3. Performance metrics

There is not an unanimous performance metric that is used in the field of SLAM research. The TUM RGB-D SLAM benchmark [28] calculates the relative pose error over a fixed time interval,

$$E_i = (Q_i^{-1} Q_{i+\Delta})^{-1} (P_i^{-1} P_{i+\Delta}), \tag{2.1}$$

where Q is the ground truth pose and P is the estimated pose, represented in homogeneous coordinates. Δ is a constant time interval. The metric reflects the local drift caused by the algorithm. The proposed performance metric to evaluate a complete trajectory is the root mean square error of the relative pose error over all time indices,

$$RMSE(E_{1:n}, \Delta) = \left(\frac{1}{m} \sum_{i=1}^{m} ||trans(E_i)||^2 \right)^{1/2}, \tag{2.2}$$

where “trans” indicates the translational component. Only the translational component is used since rotational errors will result in translational errors. However, rotational errors early in a sequence could result into large translational errors for long trajectories. Separating rotational errors and translational errors is therefore preferred to allow closer analysis of the SLAM algorithm's output. The KITTI visual odometry benchmark [11] does separate the translation and rotation errors, using the same metric as the TUM dataset. The KITTI benchmark extends the metric by evaluating the relative pose error as a function of length and velocity. The resulting equation is

$$E_i = \frac{1}{\Delta L} (Q_i^{-1} Q_{i+\Delta L})^{-1} (P_i^{-1} P_{i+\Delta L}), \tag{2.3}$$

where ΔL is the chosen traveled distance interval. Note that the relative pose error only reflects the local drift within the time interval Δ of a visual odometry or SLAM algorithm.

The global error over the full trajectory can be measured with the absolute trajectory error,

$$F_i = Q_i^{-1} S P_i, \tag{2.4}$$

used in the TUM benchmark [28] (where S is the least squared rigid body transformation overlapping P on Q). The KITTI benchmark does not utilize the absolute trajectory error, which makes the benchmark more a visual odometry benchmark rather than a SLAM benchmark. Visual odometry does not utilize a full trajectory pose optimization while SLAM does.

The Mask-SLAM research [15] develops a V-SLAM algorithm in dynamic conditions and provides two metrics to evaluate their SLAM method in a dynamic environment. The paper indicates that ORB SLAM [20] can lose track during a trajectory. The first performance metric is called the Mean Tracking Rate, which is the average value of the position when the algorithm loses track (in [%] of total trajectory distance). Then for the trajectories with a mean tracking rate of 80% or higher, the Mean Trajectory Error is calculated. The mean trajectory error is the same as the absolute trajectory error in equation 2.4, but instead of taking the Root Mean Square Error, the mean of the error is utilized. However, the pose estimation cannot be considered successfully tracked if the algorithm tracked only 80% of a trajectory that is between 100 and 500 meters long. For this research every trajectory must be tracked for 100% to be considered successful. When the trajectory was succesfully tracked, the pose estimation is used for the performance metric.

2.4. Type of SLAM algorithms

There are several types of SLAM algorithms that can be utilized for intelligent vehicles, which observe landmarks differently to estimate the pose. The selected type of algorithms are all based on the KITTI rankings [12]. The different type of algorithms for automotive applications can be divided into four categories: direct V-SLAM, feature based V-SLAM, LIDAR based SLAM and sensor fusion SLAM. Each category has a wide selection of developed SLAM algorithms, with new ones being developed nearly every month (observed from the submission dates in the KITTI ranking [12]). Each method needs to find a way to match landmarks between images or scans, so the displacement of an agent can be estimated.

Direct V-SLAM utilize the pixel intensity values of the images directly to recognize landmarks between images. There are methods that try to match every pixel between two images like Direct Tracking And Mapping [22], but this is a very computational expensive method and will not be used for large scale outdoor SLAM methods. LSD SLAM [8] matches only the pixels that are distinctive, an example shown in figure 2.2a. These are predominantly pixels that describe edges and are close to the camera. The best direct V-SLAM method according to the KITTI benchmark is DSO SLAM [30], which estimates its pose by selecting distinctive pixels based on intensity gradient. The general advantage of direct methods is that it allows localization in low feature environments, like empty hallways. It also does not require any pre-processing of the image, which is needed for feature based V-SLAM methods.

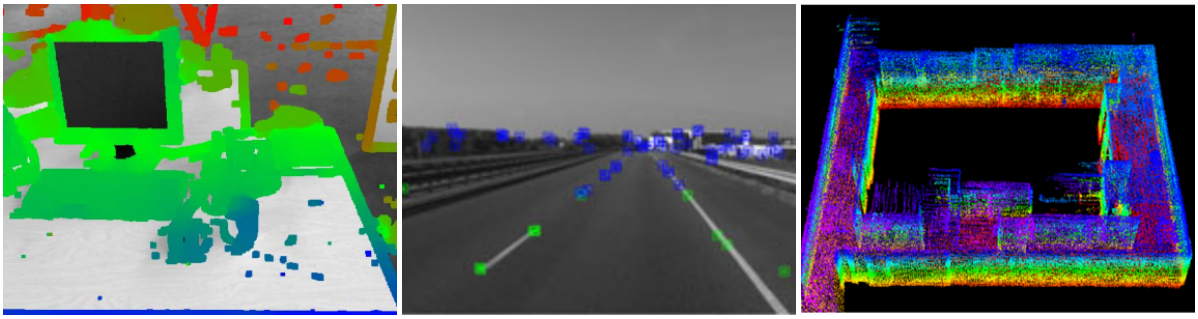
Feature based V-SLAM algorithms pre-process images, select key points and describe them in a feature vector. The feature vector is designed such that the keypoints are scale, orientation and illumination independent. ORB SLAM [19] selects corners in the image based on surrounding pixel intensities (called FAST keypoints [25]). An example is shown in figure 2.2b. The corner is described uniquely by comparing its intensity to its surrounding in a star shaped pattern and converting the results into bit-string feature for faster computation (called BRIEF features [3]). The best feature based visual odometry algorithm according to the KITTI ranking [12] is SOFT odometry [4], which detects corners through a blob and corner detector developed in [10]. However, this algorithm is not publicly available and cannot be tested in this research.

LIDAR SLAM is according to the KITTI benchmark [12] a less popular research topic but a more accurate method. LIDAR Odometry and Mapping (LOAM) [31] is the most accurate LIDAR method in the KITTI rankings (an example shown in figure 2.2c). The method estimates its pose based on detected sharp edges and planar patches, which can be detected by a smoothness term.

The most accurate method in the benchmark is the sensor fusion method V-LOAM [32], which is an extension on the LOAM method. The LIDAR that LOAM uses can complete a scan from side to side with a frequency of 1 Hz. The V-LOAM method utilizes a feature based visual odometry method (called DEMO [33]) to a pose estimation with a higher frequency. It then refines its pose estimation on a lower frequency with the LIDAR data and the pose estimation done by the LOAM algorithm.

The SLAM algorithm needed for this research needs to be open source in order to test its robustness to a dynamic environment. Although SLAM is a popular research topic, the algorithms are rarely open source. An algorithm that is versatile, state of the art and open-source is ORB SLAM [19]. The feature based V-SLAM method is able to handle monocular, RGB-D and stereo camera setups. Furthermore, an ORB SLAM based algorithm (Mask-SLAM [15]) has already been tested with the CARLA simulator. This makes ORB SLAM the

algorithm that is utilized to test the effect of a dynamic environment on the pose estimation performance.



(a) Example of direct V-SLAM method: LSD SLAM. Utilizes pixel intensity directly [7]

(b) Example of feature based V-SLAM method: ORB SLAM. Selects parts of the image and converts them to feature descriptors which allow easy matching. [19]

(c) Example of LIDAR based SLAM method: LOAM. LIDAR produces point clouds based on laser measurements [31]

Figure 2.2: Visualization of different types of SLAM algorithms

2.5. SLAM in dynamic environments

The invalidity of the static environment assumption in SLAM algorithms is known in the research community and many constructions have been developed to filter the dynamic data. Semantic segmentation could be a solution for dynamic object filtering in V-SLAM methods ([15], [35]). Mask-SLAM [15] redistributes ORB features [20] that describe vehicles or the sky, which should result into a pure static distribution of features. The results illustrate that ORB SLAM tends to lose track in a dynamic environment and different weather conditions. When the mask is applied, the SLAM algorithm does not lose track as much as the plain ORB SLAM method. The accuracy of the pose estimation does increase slightly, but Mask-SLAM does not provide consistently better results. The paper provides some examples on why the method performs worse, but fails to indicate the source of the error. The paper also performs their experiments in a dynamic environment with different weather conditions. This makes it challenging to isolate the source of the performance change.

LIDAR solutions have been proposed where scan to scan subtraction reveals the dynamic points within a point cloud, which can then be removed. In an offline method, multiple points clouds from the same trajectory can be compared to remove dynamic or semi-static objects (e.g. parked vehicles [5]). For an online method [23], two scans can be compared (from two LIDAR rotations), and subtracted to remove dynamic objects. However, this method only works on a very slow moving base (i.e. 1 m/s).

The examples illustrate that filtering dynamic objects from images and point clouds for the automotive industry is still an open problem. However, most research focus on a solution rather than understanding the problem. No research exist that analyses the effect of a dynamic environment on different SLAM algorithms. It could be that certain SLAM algorithms are very robust for a dynamic environment and only fail under very specific conditions. This would radically decrease the scope of the problem and could then potentially be solved.

In summary, no benchmark exists that tests SLAM algorithms in dynamic conditions of an intelligent vehicle. However, isolating the performance due to a moving environment is difficult unless all environment conditions can be controlled. Nonetheless, testing the robustness of SLAM algorithms in a moving environment could help identify which type SLAM algorithm is most suitable for the automotive industry. Furthermore, robust and simple solutions could be developed to ensure SLAM algorithms function in dynamic conditions. The experiment will be performed in a simulator called CARLA to control as much of the environment as possible. The methodology will be tested with the feature based V-SLAM algorithm: ORB SLAM [19]. The algorithm is state of the art, open source and has already been tested with CARLA [15].

3 Methodology

The challenge of this research is to develop a methodology that reflects the effect of a dynamic environment onto the performance of a SLAM algorithm. The experiment will be performed in a simulator in order to get fully constant environment conditions. The methodology consist of a simulation, the SLAM algorithm and the performance metrics. First the simulation in CARLA is discussed which explains the simulator choice, detailed description on how the scenarios are programmed, environment conditions and the general pipeline of the experiment. Next, the design of ORB SLAM is explained. The focus of this section is to explain how certain design choices could affect the robustness to a dynamic environment. Furthermore, assumptions to incorporate the algorithm in CARLA are explained. Finally, the performance metrics section explains the central reference system and the choice of metrics.

3.1. CARLA simulation

The objective of this research is to develop an experiment that isolates the effect of a dynamic environment on the pose estimation of a SLAM algorithm. The most basic setup would be to drive a certain trajectory in completely static conditions, drive the same trajectory with moving vehicles and compare the two pose estimations. However, there are many aspects that can influence the resulting pose estimation which need to be controlled in order to isolate the effect of a dynamic environment. For instance weather conditions should stay the same throughout the experiment. Naturally rain will imposes noise on sensor data, making repeatability difficult. Moreover, feature based V-SLAM algorithms select features based on pixel intensity [25] or pixel gradient [4]. This means that some features could describe cloud formations, which would influence the pose estimation. The experiment should therefore be performed during a clear sunny day. Due to the use of pixel intensity, many features of a V-SLAM algorithm can describe the edge of a shadow. Therefore the same trajectory under the same weather conditions but at a different time of day could lead to a different pose estimation. Furthermore, the ground truth should be reliable. Triangulation is used in most indoor SLAM experiments for ground truth but this method is difficult to use over long trajectories. The KITTI dataset [11] utilizes a very accurate variation of GPS, but the signal is not available for all trajectories. Finally, it is important that the vehicle drives exactly the same trajectory during static and dynamic conditions. SLAM algorithms can localize with centimeter precision, so there should not be any variation between the two trajectories if the effect of a dynamic environment is to be isolated. It is clear that due to these requirements, the experiment cannot be performed in real life and is therefore performed in a simulator. This gives the potential for full control of the environment, providing an accurate ground truth and driving identical trajectories.

The experiment is performed in an open-source simulator called CARLA [6]. CARLA is based on a server-client system in which the server renders the scene and a python client provides the commands. It has a set of unique features that makes the platform the appropriate choice for this experiment. First, CARLA has a elaborate sensor suite which includes the option to place multiple cameras and LIDAR sensors. This means that monocular V-SLAM, stereo V-SLAM and LIDAR based SLAM algorithms could be tested. Other simulation platforms like the VIPER benchmark [24] allow for monocular SLAM only. Monocular SLAM is sensitive for scale drift and will generally have larger pose estimation error than its stereo counterpart. Another important feature is CARLA's synchronous mode, which makes sure that all sensor data and control messages

are received. In asynchronous mode the simulator will provide a real-time experience, showing smooth and continuous images through the rendered world. However, to ensure this real-time experience a variable time step is utilized. This means sensor messages from the server will be delayed or even skipped. Synchronous mode is vital to create repeatable experiments, especially when exactly the same trajectory over time has to be driven. Furthermore, CARLA's API allows full control of weather conditions and time of day. The simulator also allows for easy access to ground truth poses. These will be more accurate than the ground truth of a real life experiment since there are no measurement errors in a simulation. Finally, CARLA provides the user with free maps representing urban environments in which the vehicle can roam. Other simulators like Airsim [27] do provide a realistic driving model and API but the user needs to create their own maps in Unreal Engine. The sensor suite, synchronous mode and free environments are the unique features that provides everything needed to perform the experiment.

Trajectory simulation On paper CARLA has all the features that are necessary to isolate the effect of a dynamic environment on the performance of a SLAM algorithm, however in reality problems occur. The biggest challenge is to get exactly the same trajectory in different dynamic conditions. This is because CARLA is designed to test and evaluate a full autonomous pipeline and expects the user to design a controller themselves. However, this research focuses solely on the computer vision and state estimation part of the pipeline. CARLA provides two solutions for this application: autopilot mode and a CARLA agent. The autopilot obeys all traffic rules and roams the map randomly at the maximum allowed speed. The CARLA agent is a developed control agent for which a fixed speed can be set. It can be implemented in your python client and has a complete pipeline with trajectory planning, collision avoidance and PID controller. Several designs were tried and failed, which include:

1. Synchronous mode, record autopilot commands, feed commands in different scenario.
2. Synchronous mode, record CARLA agent commands, feed commands in different scenario.
3. Non-synchronous mode, record CARLA agent commands, feed commands in different scenario.
4. Synchronous mode, utilize waypoints to define trajectory.

The first attempt was utilizing synchronous mode, use the autopilot on the vehicle and record all the throttle and steer commands. The recorded commands would be used as the vehicle control input to replicate the same trajectory under different conditions. However, slight deviations in the trajectory occurred over time which would result into a crash for some trajectories. Furthermore, the use of the autopilot does not allow any control of the vehicle, which makes creating dynamic scenarios difficult. The second attempt was using the designed control agent from CARLA, utilizing synchronous mode and recording the control commands. The agent allows the user to specify a destination, making it easier to create certain dynamic scenarios. However, utilizing synchronous mode and recording these control commands was not possible. The client never receives the control commands and freezes. When utilizing this same method in asynchronous mode, the system does not freeze. However, the ego-vehicle displays a lateral oscillated motion, which can be caused by either missing control messages or by the PID controller.

The final design allows the user to control the direction of the vehicle and allows for identical trajectories under different dynamic conditions. The roads on each map in CARLA can be defined by a series of waypoints. Each waypoint has a road id, lane id, is_intersection boolean, heading and position in the world. Instead of utilizing the control commands, the position of the vehicle for each simulation step can be specified by using the waypoints. After each simulation step the vehicle is "teleported" to the next waypoint. A desired constant velocity (v_{req}) can be simulated since the user can define the distance between each waypoint (d_w) and the server simulation frequency (fps):

$$d_w = v_{req} \cdot fps \tag{3.1}$$

Note that this method completely disregards any vehicle dynamics (or physics for that matter). For example, driving 100 km/h and doing a 90 degrees turn would be possible in this simulation. It is important that the chosen velocity is feasible in real life, as the simulation would accept any input.

The scenarios that are simulated in CARLA will focus on vehicle oriented scenarios instead of pedestrian oriented scenarios. The methodology will be tested with ORB SLAM [19]. The algorithm is a corner feature based V-SLAM method, which means that the algorithm selects corners from the image that are used as constraints

to optimize a pose estimation. ORB utilizes FAST keypoints [25] to select areas of the images that will be converted to a feature vector. ORB SLAM divides the image in a grid and aims to select five features in each cell. Since a vehicle covers a larger area of an image and has more corners than a human, ORB SLAM is more likely to fail due to vehicles than pedestrians. Therefore, this research focuses on vehicle oriented dynamic scenarios. Only one vehicle will be equipped with a stereo camera setup and is referred to as the ego-vehicle.

The experiment will test two different dynamic scenarios: ego-vehicle is driving behind a vehicle and vehicles driving on the opposite lane towards the ego-vehicle. The waypoints are also utilized to generate the trajectories of the vehicles that are not equipped with cameras. Before the simulation start the complete trajectory of the ego-vehicle is defined, which is a list of waypoints. This is done by selecting a starting location in the map and append the next waypoint to the list that is d_w away from the last waypoint. If there is an intersection, always choose the first waypoint. Do this until the desired distance is travelled. To create dynamic scenarios, the other moving vehicles need to have a list of waypoints that define their respective trajectories. This is done by looping through the waypoint list of the ego-vehicle and define some algorithm which selects the waypoint for that specific dynamic scenario. It is easy to select a waypoint that is on the same lane and d_x meters away. For the first dynamic scenario (driving behind a van) a waypoint is selected that is d_{iv} meters away from the ego-vehicle's waypoint (where d_{iv} is the required inter-vehicle distance).

Simulating the second scenario (vehicles driving on the opposite lane) proofs more difficult. The idea is to take the complete list waypoints of the entire map and do an exhaustive search to select the waypoint that is closest the ego-vehicle waypoint with the condition that:

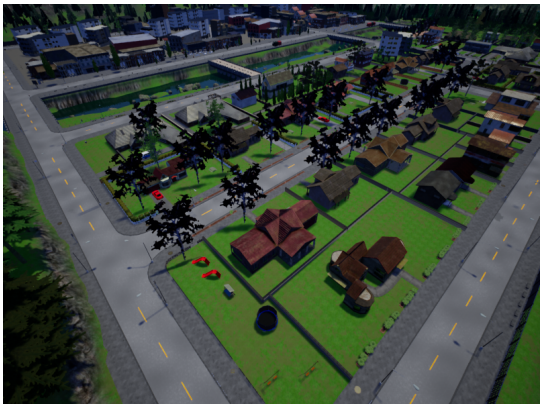
1. the waypoint has the same road id as the ego-vehicle waypoint
2. the waypoint has not the same lane id as the ego-vehicle waypoint.

This should create a list of waypoints that is exactly the same trajectory as the ego-vehicle, but in the opposite lane and the vehicle is driving backwards (since the heading is opposite). However, during intersections the road id on the opposite lane is not the same, so this method cannot be used. Instead for the waypoints of the ego-vehicle where is_intersection=True a zero is added to the list and then replaced afterwards with a waypoint. The replacement protocol works as follows: take the first waypoint after the last intersection waypoint (which is a zero on the list). Replace it by the next waypoint that is d_w meters away. When there are two waypoints, choose the waypoint that has the closest road id to the ego-vehicle waypoint so the opposing vehicle drives the same direction as the ego-vehicle. When the list consists of only waypoints, flip the list so the opposing vehicles will drive forward instead of backward. When the opposing vehicles are at the end of their trajectory, respawn them at the beginning of the trajectory.

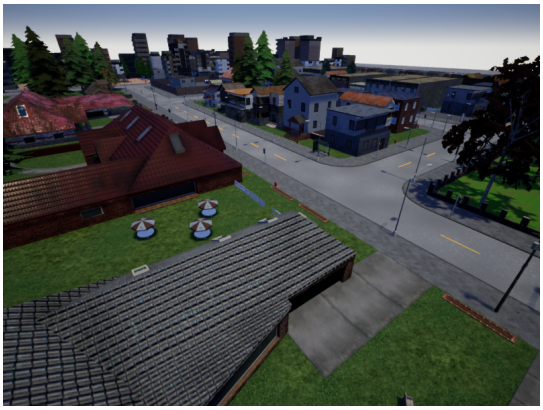
Environment description This experiment will use three different towns provided by the CARLA simulator which will represent the urban environment. Each map has a different aspects which makes pose estimation challenging for the algorithm. Each town is numbered as town 1, town 2 and town 3 and are shown respectively in figure 3.1a, 3.1b and 3.1c. Town 1 represents a suburban environment located near the woods. The area contains low-rise and mid-rise type of buildings. The roads are narrow two lane roads, which pass through populated areas as well as part of a forest. Town 2 is a smaller map, but a denser populated area. It contains more buildings and has more similar style houses as oppose to town 1 which has many unique structures. Like town 1, town 2 has a narrow two lane infrastructure. Town 3 is the biggest map of the three towns. It has two or four lane roads and contain wide lanes. The town contains a tunnel, roads underneath a bridge, roundabouts and road elevation.

The scenarios are tested on 10 different trajectories, which are divided over the three separate maps. The trajectories are through different parts of the towns, which might cause the algorithm to provide dissimilar results. Table 3.1 contains a short description of each trajectory. Some basic observations of the conditions are:

- Trajectories in town 1 and town 2 are on small two lane roads and have buildings or other landmarks near the road.
- Town 2 trajectory 1 starts in a forest and drives into the town.
- Town 3 has much wider roads. The lanes are wider and the town contains four lane roads.
- Town 3 trajectory 1 starts at the entrance of a tunnel.



(a) Aerial view of town 1. Suburban type of environment with low and mid-rise buildings and narrow two lane roads.



(b) Aerial view of town 2. Smaller suburban environment. Densely populated area with low and mid-rise buildings. Town has narrow two lane roads.

(c) Aerial view of town 3. Large urban area with wide two and four lane roads. Contains a tunnel, bridges and roundabouts as well as elevation in the roads. City has low, mid and high-rise type of buildings.

- Town 3 trajectory 2 is on four lane roads only, with large intersections and road elevation at the end of the trajectory.
- Town 3 trajectory 3 contains a small roundabout, performing a complete 360 degree turn.
- Town 3 trajectory 4 starts underneath a bridge.

Table 3.1: Short overview of the trajectories of the three towns.

Town	Trajectory	Lanes	Trajectory description	Remarks
1	1	2 (narrow)	Right, right	Suburban environment
1	2	2 (narrow)	Right, left, left	Suburban environment
1	3	2 (narrow)	Left, right, right	Suburban environment
2	1	2 (narrow)	Left, left, left	Starts in a forest, continues in urban environment
2	2	2 (narrow)	Left, left, right, left, left	Urban environment
2	3	2 (narrow)	Left, 3x right, left, left	Urban environment
3	1	2 and 4 (wide)	Long left, slight left, slight right	Starts in tunnel
3	2	4 (wide)	Left, right, long right, right	Road elevation at end of trajectory
3	3	2 (wide)	3x Left, right, long right	Full rotation on roundabout
3	4	4 (wide)	Right, long right, left, long right	Starts underneath a bridge

Experiment pipeline The ego-vehicle will be equipped with a stereo camera setup since monocular V-SLAM suffers from scale drift and RGB-D SLAM has a lower depth range. The camera images are written to the disk during the simulation, which will be converted to a rosbag afterwards. Rosbags allow sensor measurements to be recorded and to play back, so algorithms like ORB SLAM can utilize it without having to do the same experiment again. Although CARLA provides a rosbridge, communication errors occurred and a custom .png to rosbag converter was written. These rosbags will only contain the left and right image of the stereo camera and a timestamp. There will be no pose information in the rosbag, so ORB SLAM estimates its position purely based on the sequence of images.

The pipeline of the experiment for a dynamic scenario on a single trajectory is shown in figure 3.2. Note that the pipeline is very versatile due to the flexibility of the simulator. For example the same experiment can be done with LIDAR by adjusting the Python Client code. Moreover, only the rosbags and ground truth poses are needed to establish the robustness of other V-SLAM algorithms.

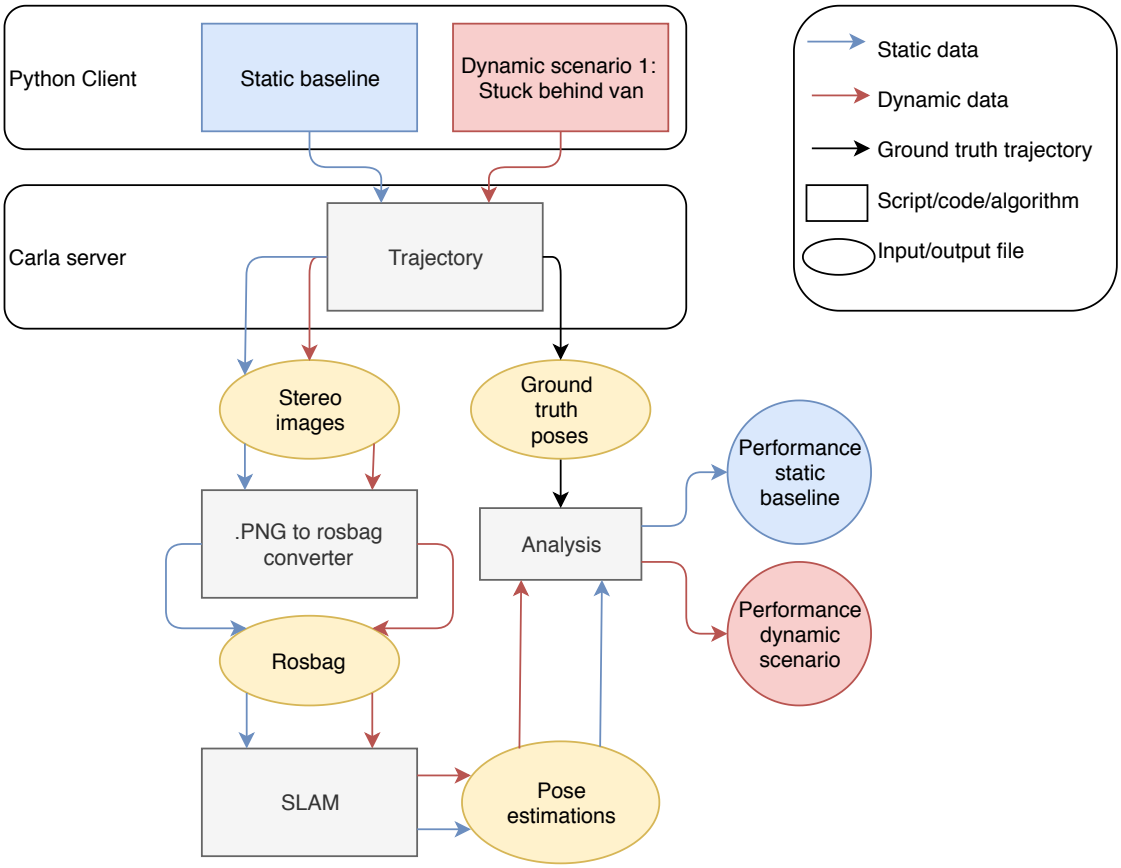


Figure 3.2: The complete pipeline of the experiment for a single trajectory. It illustrates all the steps necessary to test the robustness of the algorithm to a specific dynamic scenario.

Sensor setup and experiment details The stereo camera setup in the CARLA environment mimics the setup that is used in the KITTI benchmark as much as possible. ORB SLAM provides an example setting file of the KITTI dataset, which includes the optimized set of ORB parameters. It is assumed that by mimicking the setup of KITTI the ORB parameters do not have to be optimized. The rectified images of KITTI are 1240x376 pixels. However, the cameras in CARLA can only be defined by a single field of view, which defines both the horizontal and vertical field of view. To prevent distortion in the image, a square image is defined which equals the total amount of pixels from the rectified images of the KITTI dataset. The simulated cameras have a field of view of 90 degrees, an image size of 680x680 pixels, and a stereo baseline of 0.54 m.

The vehicle will drive exactly the same trajectory in the same conditions, except of other moving vehicles. Each trajectory is a total distance of 500 meters, travelled at a constant velocity. A low velocity is favorable for

good results as ORB SLAM has trouble estimating a position at high angular velocity. Having the speed setting set too high will cause high angular velocities in corners, increasing the likelihood that ORB SLAM fails. Furthermore, simulating a low velocity ensures that the vehicle can make tight corners since the simulation does not take into account any vehicle dynamics. A constant velocity of 15 km/h is chosen. The weather condition settings are set to "ClearNoon", hence clouds will not occur which would influence the pose estimation. ORB SLAM is performed over each trajectory. This is repeated five times to take into account the non deterministic nature of the process. Finally, the moving vehicles in the scenarios will be represented by "CARLA COLA" vans, which are the largest vehicles available in CARLA and therefore the most likely to influence the pose estimation process.

3.2. ORB SLAM

CARLA contains a very elaborate sensor suite which would allow testing monocular, RGB-D, stereo V-SLAM methods and LIDAR based SLAM algorithms. This research will however focus on the performance of a single SLAM algorithm to illustrate the depth of the analysis that is possible with this methodology. The experiment will test the performance of stereo ORB SLAM [19]; a feature based V-SLAM algorithm. ORB SLAM is unique in its completely open source software which will allow to deactivate certain features of the algorithm. This will allow an in depth analysis of the strengths of the algorithm and aid in finding the source of its robustness. ORB SLAM can operate in stereo, RGB-D or monocular setups, but stereo vision would be the more likely approach in the automotive industry. Monocular SLAM cannot perceive depth in a single frame and displays scale drift over time. RGB-D cameras have a limited range in which it can estimate depth and are more used in indoor setups. Stereo vision is able to triangulate landmarks by matching features in the left and right images. Therefore, scale drift does not occur in stereo vision SLAM methods. The purpose of this research is to assist the development of SLAM methods in the automotive industry. Hence, this research will focus on the robustness of ORB SLAM in a stereo camera setup instead of a RGB-D or monocular setup.

ORB SLAM is designed to provide reliable pose estimations for real time operation in large and complex environments [20]. Although many solutions that are implemented in the algorithm ensure these requirements, some solutions could also affect the algorithm's robustness to a dynamic environment. The main solution that ORB SLAM implements to have reliable pose estimations is to create an abundance of map points and keyframes and use strict filtering mechanisms to remove redundancy. This design could increase the robustness in dynamic environments and could be implemented on other SLAM algorithms if it proofs to be successful. The results from this experiment aids the development of stereo V-SLAM algorithms in the automotive industry. ORB SLAM is a feature based visual SLAM algorithm that utilizes ORB features to perform all SLAM tasks [19][20]. ORB features [26] in an image are easily computed, quickly matched and robust to viewpoint changes. The algorithm has three main threads: tracking, local mapping and loop closure (see figure 3.3 for a complete system's overview). The tracking thread provides the pose estimations from frame to frame. The local mapping thread optimizes the local trajectory and map points to ensure accurate reconstruction of the environment around the current frame. The loop closure thread detects previously visited locations, corrects for the accumulated error and activates the full bundle adjustment to provide a global accurate pose estimation and map. Specifically the design of the feature extraction, tracking, local mapping and place recognition module could affect the performance of ORB SLAM in a dynamic environment.

The map that is utilized to estimate the position of the vehicle consists of four elements: keyframes, map points, covisibility graph and spanning tree:

- A keyframe stores the camera pose, camera intrinsics and the ORB features extracted from that frame.
- A map point stores a 3D position, viewing direction, a representative ORB descriptor, category keypoint (monocular or stereo) and the minimum and maximum distance at which the point can be observed.
- The covisibility graph connects the keyframes that share observations with a minimum of 15 matched observations. The covisibility graph provides the keyframes and map points which are utilized for the local trajectory and map optimization.
- The spanning tree is the essential version of the covisibility graph with the minimum number of connections and is used for fast identification of keyframes and map points.

Pre-processing The first step that could influence the algorithms behavior in a dynamic environment is the keypoint selection in an image. ORB SLAM selects 2000 keypoints per image based on FAST corner detector

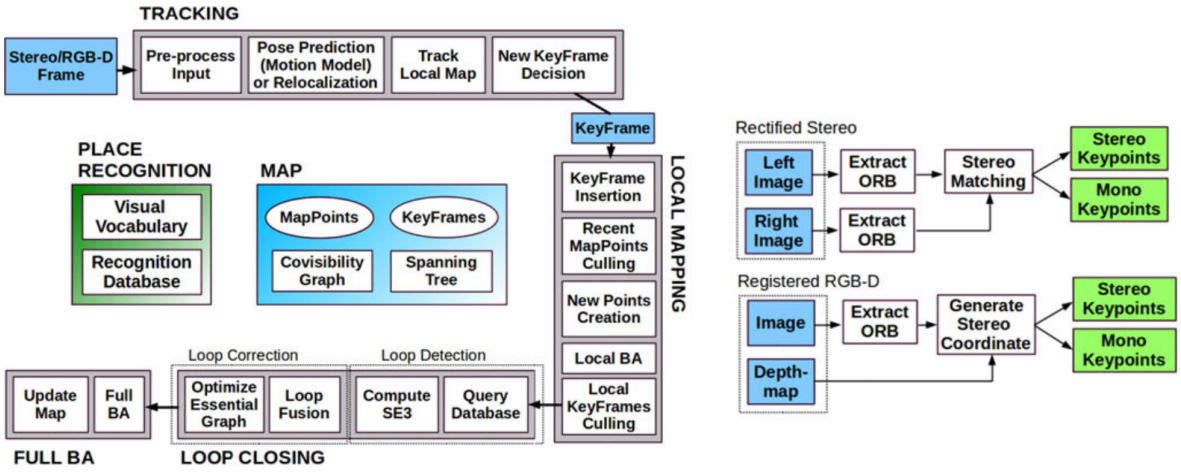


Figure 3.3: The system overview of stereo ORB SLAM. [19]

[25]. The FAST algorithm detects corners by comparing the pixel intensity C with the pixel intensity of a circle around C (see figure 3.4). If there are n contiguous pixels which are either darker or lighter than C, it is classified as a corner. ORB SLAM extracts the corners from eight scale levels to ensure scale independent features. The scaled images are divided into a scale grid and select five features from each cell to ensure homogeneous distribution of features. However, this distribution is adapted when a cell contains no corners. The corners are described with oriented BRIEF features [3], which describe the image patch with a bit string constructed by a binary intensity test[26]. The FAST keypoints and BRIEF features allow for fast feature extraction and matching, which allows real time operation of the SLAM algorithm.

After the ORB features are extracted, a match of every feature is searched in the right image along the epipolar lines [19]. Features that are matched are classified as close or far keypoints. Close stereo points do not need multiple frames for triangulation since depth can be perceived from the stereo setup. Far keypoints do not provide accurate translation information but can be utilized for orientation estimation. Close keypoints are the points which depth is less than 35 times the stereo baseline. The far keypoints and keypoints that are not matched along the epipolar line are classified as monocular keypoints, close keypoints are classified as stereo keypoints. Note that the feature choice and implementation influences how an algorithm would react to a moving vehicle. For example the algorithm might select many features on a vehicle that is close to the camera. The ORB features might describe the outside silhouette of the vehicle, edges of the window, edge of a number plate or even the edges of the tire. The estimated pose of the camera will be based on the selected features, hence the feature choice might be a negative influence on the pose estimation in a dynamic urban environment.

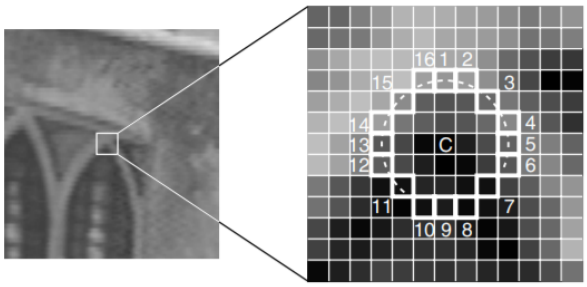


Figure 3.4: The selection of a FAST keypoint. The intensity of pixel C is compared to the pixel intensity of 16 pixels surrounding pixel C. When n-contiguous points are either darker or lighter than pixel C, the pixel is categorized as a corner. [25]

Tracking The next step is the tracking process in which the algorithm estimates the pose of the vehicle for every single frame. If tracking was successful for the last frame, a constant velocity motion model predicts the new camera pose [20]. This will aid a guided search of the map points from the last image to match with the current image. If this fails a wider area of the estimated position is searched. Note that a guided search

could influence the robustness to a dynamic environment. If the motion model is correct, moving objects might not be matched because it. When there is an initial set of matched features, a rough pose estimate is performed with motion only bundle adjustment. With this estimate, map points that are expected to be seen can be projected on the current frame and more features can be matched. This will lead to the final pose estimation via motion only bundle adjustment, which is defined as:

$$\{\mathbf{R},\mathbf{t}\} = \underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_{i\in\chi} \rho\left(\left|\left|\mathbf{x}^i_{(.)} - \pi_{(.)}(\mathbf{R}\mathbf{X}^i + \mathbf{t}\right|\right|^2_{\Sigma}\right) \tag{3.2}$$

$$\pi_m\left(\begin{bmatrix}X\\Y\\Z\end{bmatrix}\right) = \begin{bmatrix}f_x\frac{X}{Z}+c_x\\f_y\frac{Y}{Z}+c_y\end{bmatrix},\ \pi_s\left(\begin{bmatrix}X\\Y\\Z\end{bmatrix}\right) = \begin{bmatrix}f_x\frac{X}{Z}+c_x\\f_y\frac{Y}{Z}+c_y\\f_x\frac{X-b}{Z}+c_x\end{bmatrix} \tag{3.3}$$

$\mathbf{R}$ is the camera's orientation in SO(3) and $\mathbf{t}$ is the translation in 3D space. ρ is the Huber loss function, χ are the matched features, $\mathbf{x}^i$ are the matched features in image coordinates and $\mathbf{X}^i$ are the matched features in world coordinates. π are the projection functions, where π_m is the projection function of the monocular keypoints and π_s the projection function of the stereo keypoints. For the projection functions f_x and f_y are the focal lengths, c_x and c_y are the principal points and b the baseline[19]. The equation does not contain a time variable which indicates that the equation assumes that all matched features ($\mathbf{x}^i$) and landmarks ($\mathbf{X}^i$) remain static. The equation is optimized with a non-linear optimization framework g2o [16], which utilizes the Levenberg Marquardt optimization algorithm.

After this process, it is decided if a keyframe needs to be inserted. The features of the current frame are always matched with the features of the last keyframe (reference keyframe). When a new keyframe is inserted, the features are redistributed as explained in the pre-processing paragraph. The philosophy is to insert as many keyframes as possible for robustness and have a separate process that removes redundant keyframes. A keyframe is inserted when more than 20 frames have passed after the last keyframe insertion, more than 50 points are tracked and less than 90 % of the points are tracked from the reference keyframe (which in this case is the last keyframe). A keyframe is also inserted when the number of total matched close stereo keypoints are lower than 100 and a new keyframe would add 70 new close stereo keypoints. The placement of keyframes could have a role in the robustness to a dynamic environment, since the pose estimation of each frame is based on the features that are created by the keyframes. Hence, when a keyframe is inserted and a moving vehicle is close, many features that describe the vehicle could be tracked which would negatively influence the motion only bundle adjustment.

Local mapping The mapping process is triggered after a keyframe is inserted. The first step is to update the covisibility graph, which includes updating the connections of keyframes which share observations. Next, the bag of words representation of the image is created which will be explained in the next paragraph. After the keyframe is inserted, map points can be be culled. Map points are retained when the feature is tracked in more than 25 % of the frames in which it was predicted to be visible. Furthermore, the map point has to be visible in at least three keyframes. After three keyframes have past, these map points can only be eliminated when a keyframe was removed by the keyframe culling method at the end of the mapping process. When a keyframe is removed it could be possible that a map point is no longer visible in three consecutive keyframes and therefore removed. One could imagine that this culling method results in the removal of various dynamic map points since matching the same features in three consecutive keyframes could be challenging for moving features. After recent map points are removed, the features of the keyframe are projected onto the map. A depth estimate is provided by either the stereo setup or with covisibility of multiple keyframes.

Next, local bundle adjustment is performed which optimizes all the keyframes that are connected in the covisibility graph with the current keyframe and all the map points contained by these keyframes. Let the set of keyframes that are connected by covisibility to the current keyframe be K_L and the points within those keyframes be P_L. There are a set of keyframes that do not have covisibility with the current keyframe, but do have covisibility with map points in keyframes K_L. These keyframes are included in the bundle adjustment, however the position of these keyframes are fixed. The total set of keyframes that are included in the bundle adjustment are referred to as K_F. The local bundle adjustment results in:

$$\{\mathbf{X}^i, \mathbf{R}_l, \mathbf{t}_l | i \in P_L, l \in K_L\} = \underset{\mathbf{X}^i, \mathbf{R}_l, \mathbf{t}_l}{\operatorname{argmin}} \sum_{k \in K_L \cup K_F} \sum_{j \in \chi_k} \rho(E(k, j)) \tag{3.4}$$

$$E(k, j) = \left|\left| \mathbf{x}^j_{(.)} - \pi_{(.)}\left(\mathbf{R}_k \mathbf{X}^j + \mathbf{t}_k\right) \right|\right|^2_{\Sigma} \tag{3.5}$$

where k are the keypoints in the keyframe and j are the matched map points. In the final step redundant keyframes are removed. Keyframes are defined as redundant when 90 % of the map points have covisibility with at least 3 other keyframes. This leaves a sparse web of keyframes that contain reliable and robust features.

Loop detection Relocalization is caused by the place recognition module of ORB SLAM. It utilizes DBoW2 [9] to recognize its position, which describes images as a bag of visual words. The visual vocabulary is created in an offline step and trained on the Bovisa dataset [1] which contains real indoor and outdoor images. ORB features are extracted from the dataset and are used to create the vocabulary [18]. Each keyframe is described with a bag of words vector that contains this vocabulary. The vector is stored in the recognition database so it can be utilized for place recognition. However, since there could be visual overlap between the keyframes, the similarities between the bag of words vector and keyframe should be summed over a period of time. ORB SLAM's place recognition model needs a high similarity score over several keyframes that are connected in the covisibility graph to recognize the location. This makes the model more robust against visual overlap. The design of the place recognition model could have significant impact on the robustness to a dynamic environment.

Localization mode In localization mode, ORB SLAM estimates its position within a previously created map [19]. Localization mode deactivates the local mapping and loop closing threads and uses visual odometry and relocalizition to accurately localize the cameras. For visual odometry, the algorithm only utilizes the matches from the previous frame to estimate a position. These estimations will accumulate drift which is mitigated by feature matches in the existing map.

ORB SLAM in CARLA A set of assumptions are needed to incorporate ORB SLAM with the CARLA simulated stereo camera images. ORB SLAM needs to have rectified images and the camera calibration parameters as an input. The images are automatically rectified by placing the cameras in the vehicle on the same horizontal line. The documentation of ORB SLAM advises to use the OpenCV calibration model to provide the required camera intrinsic parameters. It utilizes Zhang's calibration method [34] which requires a checkerboard pattern in front of your cameras, which is not possible in CARLA. ORB SLAM requires focal lengths (f_x, f_y), center of the image (c_x, c_y), radial distortion parameters (k_1, k_2) and tangential distortion parameters (p_1, p_2). The cameras in CARLA can be specified by defining the image size and the field of view (FOV) of the camera. The distortion parameters can be set to zero, assuming that there is no distortion in the lens. The center of the images are half of the image size and are therefore known. If it is assumed that the lens is a thin fixed focal length lens, so the model in image 3.5 can be used. Hence, the focal length can be calculated by

$$f = \frac{h}{2\tan(\frac{FOV}{2})}, \tag{3.6}$$

where h is the size of the image and FOV is the field of view angle. To test the validity of these assumptions, the static baseline results are compared to the pose estimation accuracy given by the KITTI benchmark [12]. The static baseline accuracy are in the same order of magnitude as the KITTI benchmark, thus the assumptions seem to be valid.

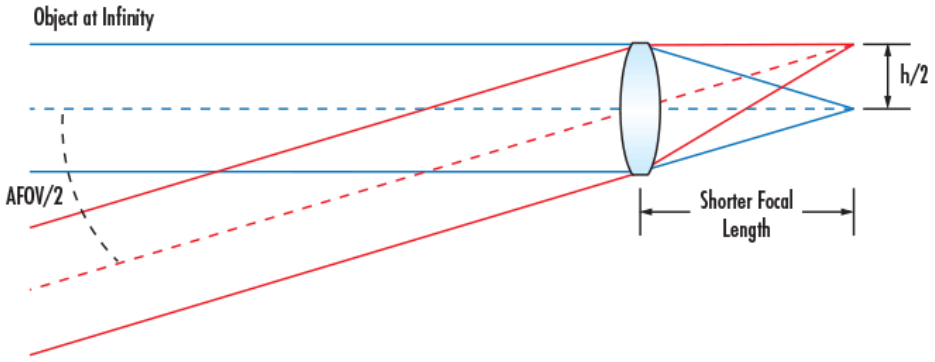


Figure 3.5: Visual derivation of field of view to focal length [14]

3.3. Performance metrics

Before the pose data can be analyzed the ground truth data and ORB SLAM data need to be converted to a central reference system and orientation convention. The two data sources utilize completely different methods to describe its pose. ORB SLAM expresses its pose with:

- Position in Cartesian coordinates
- Orientation in unit quaternions
- A right handed system
- Origin is in its first established key-frame

While the ground truth expresses its pose as:

- Position in Cartesian coordinates
- Orientation in Euler angles
- A left handed system (shown in figure 3.6a)
- Origin is an arbitrary fixed position (based on how the town was build up in the UnrealEngine axis system)

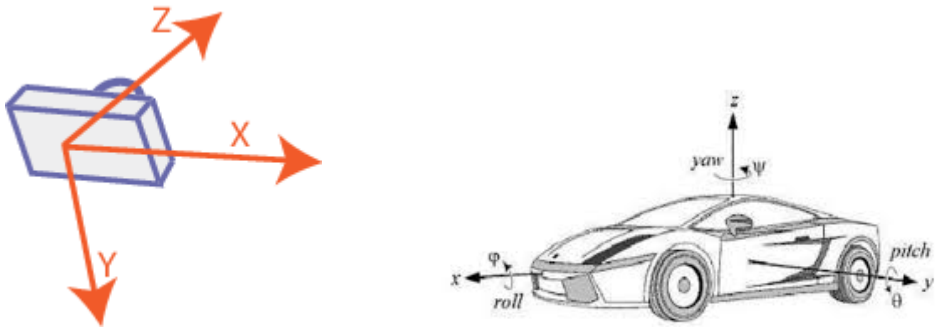
Both the ground truth and ORB SLAM data are transformed to a right handed axis system (shown in figure 3.6b) with the origin at the vehicle position at t=0 s. The poses will be represented in homogeneous coordinates:

$$Q_i = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix}, \tag{3.7}$$

where **R** is the rotation matrix and **t** is the position vector. The origin will be the starting location of the vehicle and the x direction will be pointing forward relative to the vehicle (seen in figure 3.6b).

The performance metric that will be used to compare different scenarios needs to fulfill a set of requirements. First, it is important that pose estimation errors can be traced back to a point in time. If a certain situation, or location in the town gives ORB SLAM difficulty in estimating the pose, than the metric should allow traceability. The absolute trajectory error (ATE) [28] used in the RGB-D SLAM benchmark does not fulfill this requirement. The ATE metric transforms the estimated trajectory onto the ground truth. This could lead to a large pose estimation error late in the trajectory but was caused by an error at the beginning of the trajectory. Hence, there is no trace-ability. For trace-ability it is important that estimation errors in the beginning of the trajectory do not influence the performance metric later in the trajectory. Furthermore, the metric should allow a sanity check. The pose estimation performance in CARLA should be compared to an independent source to check whether the numbers are in the right ballpark.

The performance of ORB SLAM over a trajectory will be evaluated with root mean square error (RMSE) of the relative pose error (RPE) [28] over a distance of 100 meters. The relative pose error measures the accuracy



(a) ORB SLAM right handed coordinate system

(b) Axis system that will both ground truth as ORB SLAM is converted to.[17]

Figure 3.6: Visualization of coordinate systems

of the pose over a local interval and therefore corresponds to the local drift. Hence, a pose estimation error made in the beginning will not influence the pose estimation near the end of the trajectory. The relative pose error is expressed with,

$$E_i = \frac{1}{\Delta_L}(Q_i^{-1}Q_{i+\Delta})^{-1}(P_i^{-1}P_{i+\Delta}), \tag{3.8}$$

where Q is the ground truth poses, P the estimated poses, Δ is a chosen frame number interval, Δ_L is the distance normalization and i is the frame number. E_i, P and Q are matrices representing homogeneous coordinates. In this case Δ is the frame number where the vehicle has travelled 100 meters and therefore Δ_L = 100 meters. The KITTI benchmark [11] utilizes the same metric with a minimum distance of 100 meters. Therefore, the performance measured in CARLA can be compared to the performance in the KITTI standings [12]. The relative pose error can be divided into a translational component and rotational component. The translational component is the magnitude of the position vector **t** of the error E_i represented in homogeneous coordinates (see equation 3.7). The magnitude of the rotational component is calculated by

$$|\theta| = \arccos\left(\frac{Tr(\boldsymbol{R})-1}{2}\right). \tag{3.9}$$

where Tr(**R**) is the trace of the rotation matrix within the homogeneous E_i. To establish the performance over a complete trajectory, the root mean square error (RMSE) is calculated:

$$RMSE_{trans}(E_{1:n},\Delta) = \left(\frac{1}{m}\sum_{i=1}^{m}||\text{trans}(E_i)||^2\right)^{\frac{1}{2}}, \tag{3.10}$$

$$RMSE_{rot}(E_{1:n},\Delta) = \left(\frac{1}{m}\sum_{i=1}^{m}||\text{rot}(E_i)||^2\right)^{\frac{1}{2}}, \tag{3.11}$$

where m are the total amount of evaluations. Since pose evaluation process is non deterministic, ORB SLAM will be performed several times. The performance of each trajectory will be expressed in a mean RMSE value, as well as a standard deviation.

The relative pose error can also be utilized for a closer analysis on a scenario and a certain trajectory. By shortening the evaluation distance of equation 3.8 from 100 to 10 meters and plot it over time, the effect of certain environmental conditions can be analyzed. For example the effect of a right hand turn on the pose estimation can be analyzed or the effect of an approaching vehicle.

There is a possibility that ORB SLAM fails to provide a pose estimation. There are two types of failures: tracking failure or false relocalization. When ORB SLAM fails to track features, it cannot estimate a pose and will stop providing pose estimation over time. It will try to relocalize itself as soon the algorithm recognizes its environment. However, since the trajectories do not revisit the locations, this will most likely not happen. Additionally, it could be possible that the algorithm falsely recognizes its environment and relocalizes its position. This is referred to as false relocalization. This will create a large peak in the relative pose error and distorts the mean value over the complete trajectory. The relocalization will be quite random. Therefore, trajectories where false relocalization occurs is filtered out and represented as a separate metric. The additional

metrics are the percentage of tracking failure and false relocalization with respect to the total number of tries per trajectory. These pose estimations will be filtered out from the RMSE values.

4

Experiments

The experiments are divided into the two dynamic scenarios. First scenario the ego-vehicle is driving behind a van during the complete trajectory. In the second dynamic scenario vans are driving in the opposite lane towards the ego-vehicle. There are three variations of ORB SLAM that are tested on each scenario and have a strict color convention throughout this report:

1. ORB SLAM - the algorithm according to the original paper [19], which is represented in green.
2. ORB VO - loop closing is bypassed so the algorithm cannot trigger relocalization, which is represented in blue.
3. ORB MC - map point culling is bypassed to analyze the effect on pose estimation performance in a dynamic environment, which is represented in red.

Each section in this chapter contains the results and analysis of a dynamic scenario. Each dynamic scenario contains the results of all ORB SLAM variations. For each section, first the general results are discussed, next a detailed analysis is performed to explain the results. The final section contains the results of the localization research question.

4.1. Driving behind a van

The results of the first scenario are presented like figure 4.1. Each figure contains three subplots that illustrate the overall performance of the pose estimation for each town. The first two plots show the average and standard deviation of the root mean square error (RMSE) of the relative pose error (RPE) for each trajectory. The top shows the translational component and the middle shows the rotational component. The RMSE value of all five pose estimations are indicated with an 'x' and the line indicates the mean values. Furthermore, uncertainty is represented in the colored area by adding and subtracting two times the standard deviation from the mean. This area represents the 95% confidence interval and creates a clear difference between significant and insignificant performance differences. When the confidence interval of the pose estimation of two different scenarios do not overlap, it can be referred to as a significant performance difference.

The bottom graph shows the percentage that is filtered out by either overall tracking failure or false loop closure. These results are filtered out since both failure modes influence the average root mean square error immensely. The x-axis indicates the location and the scenario. For example "T3Nr3 Dist: 15" shows the performance at the third town and third trajectory, where the distance between the van and the ego-vehicle is 15 meters. Note that the results are based on five pose estimations performed by ORB SLAM. Ergo, if 0.8 of the data is filtered out due to false loop closure, the displayed average error is of only one data point and has therefore no standard deviation.

4.1.1. Results

ORB SLAM Figures 4.1, 4.2 and 4.3 show the performance of ORB SLAM in town 1, 2 and 3 respectively. The figures show that ORB SLAM biggest mode of failure is false relocalization. In total, 40 of the 45 estimated trajectories are falsely relocalized with an inter-vehicle distance of 10 meters. Furthermore, figure 4.2 shows that ORB SLAM falsely relocalizes the vehicle three out of five times times at town 2 trajectory 1 in static

conditions. False relocalization in static conditions does not occur in the other nine trajectories. Moreover, a difference in behavior can be observed between town 3 and the other two towns. In town 1 and 2, ORB SLAM has successfully estimated the trajectory with a vehicle in front of the cameras at a distance of 15 meter. However, in town 3 ORB SLAM fails to provide accurate pose estimations at this distance in three out of four trajectories. Note that according to figures 4.1 and 4.2 ORB SLAM has a slight increase in average RMSE values when the inter-vehicle distance decreases in town 1 and 2. Nevertheless, the increase usually is within the 95% confidence interval. The variance of the translational component does increase as the van is closer to the cameras.

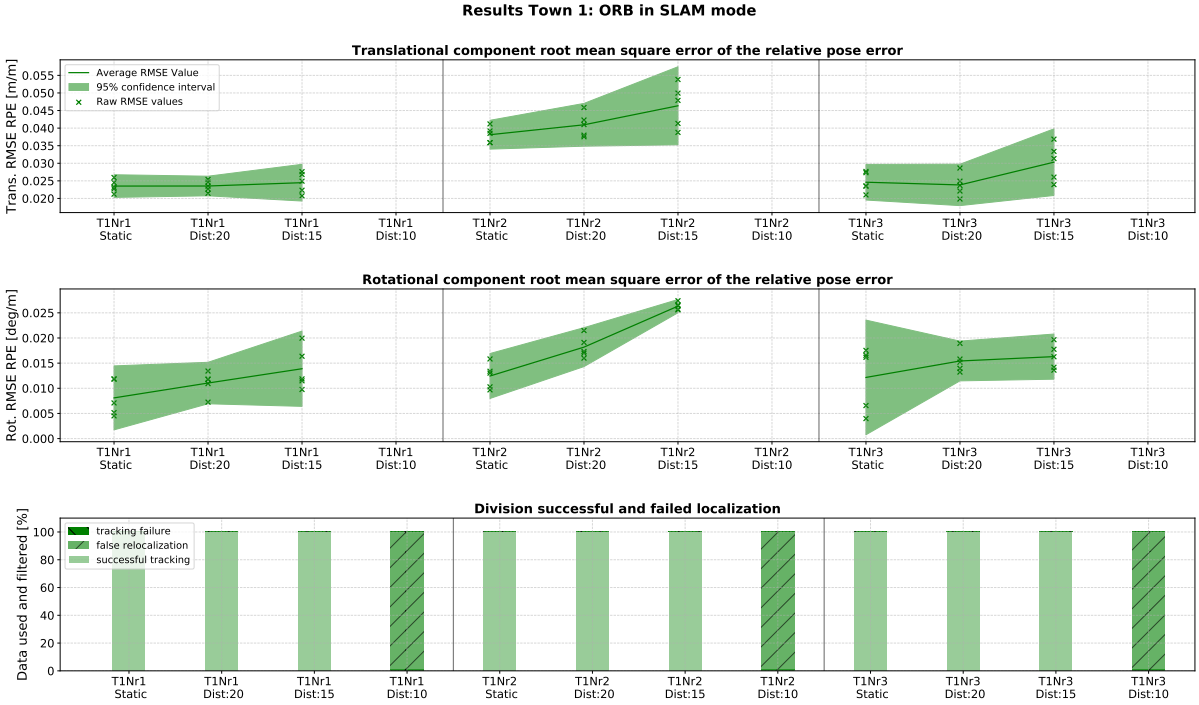


Figure 4.1: Performance ORB SLAM in town 1. False relocalization occurs at an inter-vehicle distance of 10 meters. The difference from the static scenario to dynamic scenario is within the confidence interval. The variance of the translational component increases as the van is closer to the cameras.

ORB VO Figures 4.1, 4.2 and 4.3 show false relocalization is the root cause of pose estimation failure, which is triggered by the place recognition module. If the place recognition module is bypassed, the algorithm becomes a visual odometry algorithm, referred to as ORB VO. Bypassing the place recognition module gives more insight on the effect of the violation of the static environment assumption on the pose estimation. Figures 4.4, 4.5 and 4.6 show the pose estimation performance of ORB VO in the three towns. Without the place recognition feature, ORB VO is now able to estimate poses on the trajectories which previously had structurally occurring false relocalization (e.g. town 2 trajectory 1). According to figure 4.2 ORB SLAM would relocalize in town 2 trajectory 1 for all conditions. However without the place recognition module the algorithm can accurately localize itself up and including an inter-vehicle distance of 15 meters. ORB SLAM also relocalizes at an inter-vehicle distance of 15 meters for town 3 trajectory 1, 2 and 4 (see figure 4.3). As can be seen in figure 4.6 ORB VO provides accurate pose estimations for trajectory 1 and 4 at 15 meters distance between the vehicles, but fails to provide an accurate pose estimation for trajectory 2. Finally, the algorithm provides a pose estimation at an inter vehicle distance of 10 meters. However, the root mean square error drastically increases in this scenario, providing unreliable pose estimations.

ORB MC ORB SLAM has a map point culling method which filters out map points that are not matched over multiple positions [20]. This filter mechanism is designed to keep reliable map points, which are used for the pose estimation. Moreover, this filter mechanism could have significant effect on the robustness of ORB SLAM in dynamic environments. ORB MC is the ORB SLAM algorithm which bypasses the place recognition module and the map culling method to test the effect of this filter mechanism on a dynamic environment.

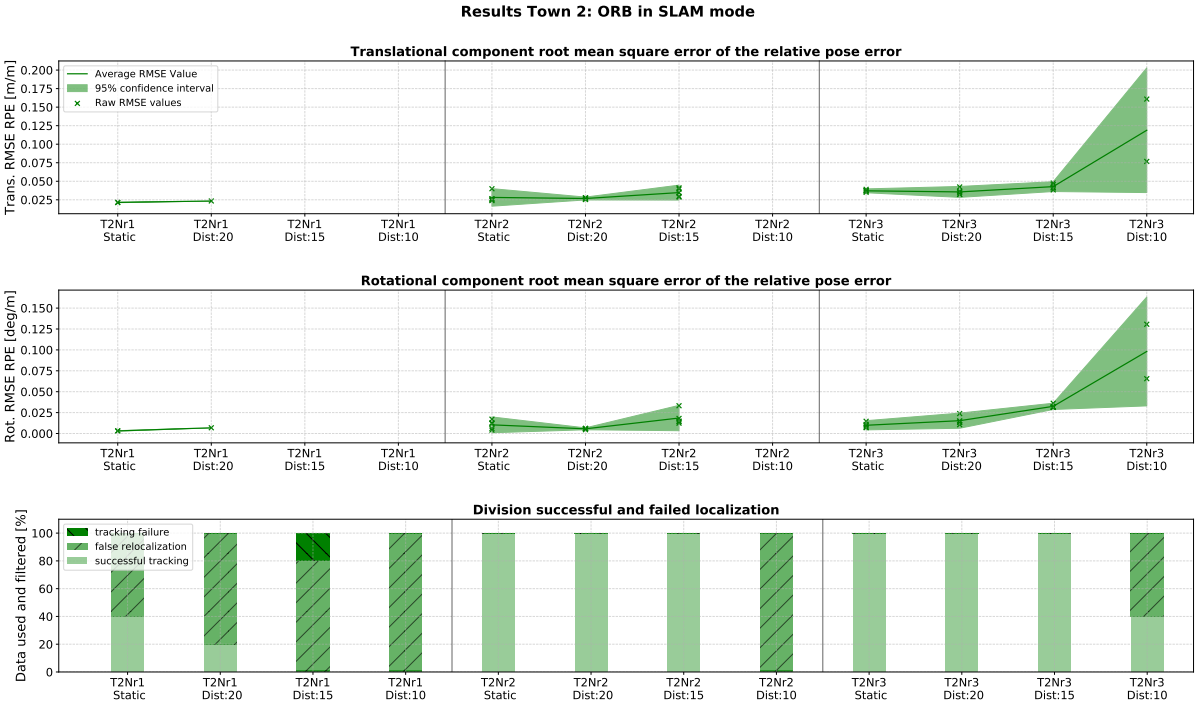


Figure 4.2: Performance of ORB SLAM in town 2. False relocation occurs in static and dynamic conditions for trajectory 1. Trajectory 2 produces false relocalization when there is an inter-vehicle distance of 10 meters. Trajectory 3 has two instances that successfully tracked the complete trajectory. The instances show a significant increase in error but also a large variance.

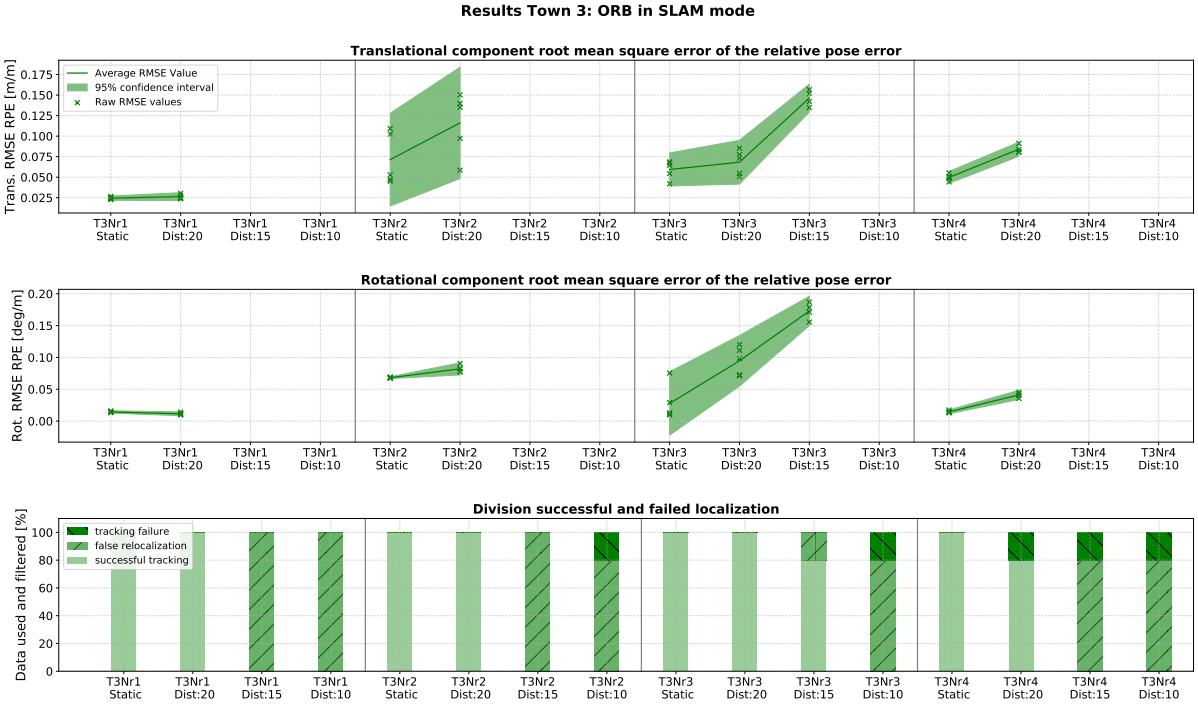


Figure 4.3: Performance of ORB SLAM in town 3. False relocalization occurs at an inter-vehicle distance of 15 meters or closer for trajectories 1,2 and 4. Dynamic scenarios have significant impact on the performance of the algorithm.

Figures 4.4, 4.5 and 4.6 show the results. Figures 4.4 and 4.5 illustrate that by bypassing the map point culling method the average root mean square error increases in town 1 and 2 with an inter-vehicle distance of 10 meters. This suggests that the map culling method does influence the robustness to a dynamic environment. Figure 4.6 shows similar performance for ORB VO and ORB MC, suggesting that the map culling method does

not affect the performance in town 3.

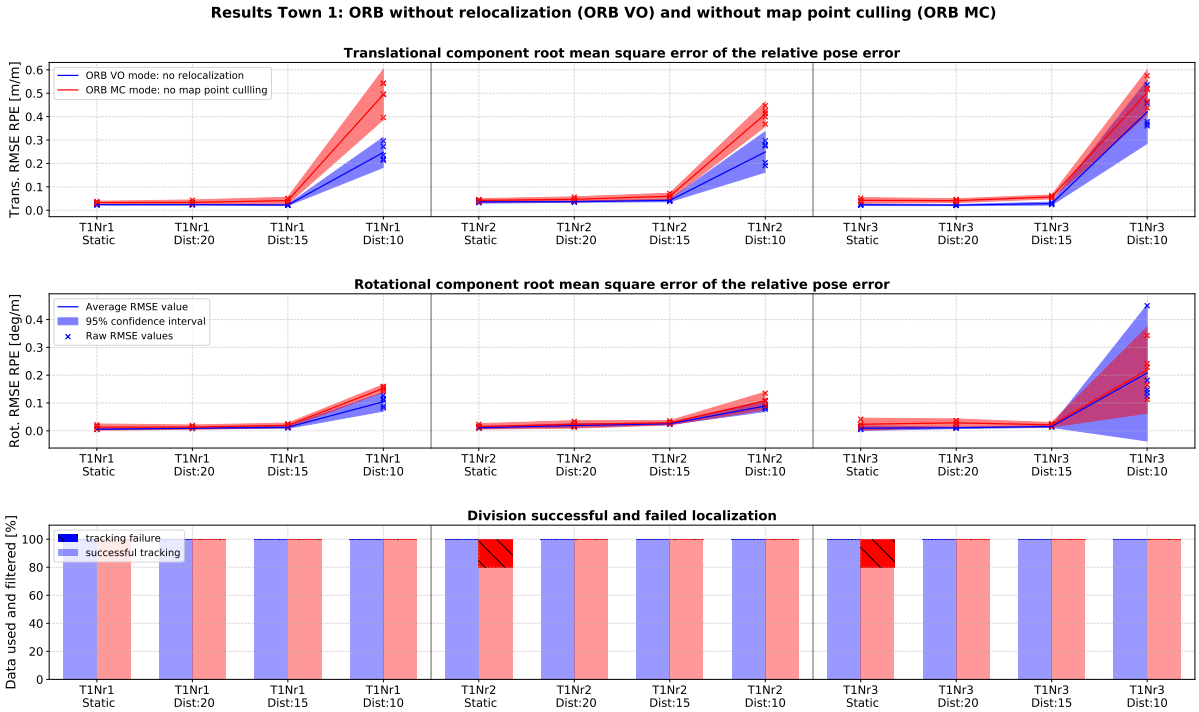


Figure 4.4: Performance of two variations of ORB SLAM in town 1: ORB VO (relocalization is not possible) and ORB MC (no map culling module). Without relocalization a significant increase in pose error occurs at an inter-vehicle distance of 10 meters. Disabling the map culling module increases the translational error at an inter-vehicle distance of 10 meters.

Figure 4.5: Performance of ORB VO and ORB MC in town 2. Both algorithms have successfully tracked the poses over the all trajectories. The pose error and variance increases significantly at an inter-vehicle distance of 10 meters. Disabling the map point culling increases the pose error and variance even more at 10 meters inter-vehicle distance.

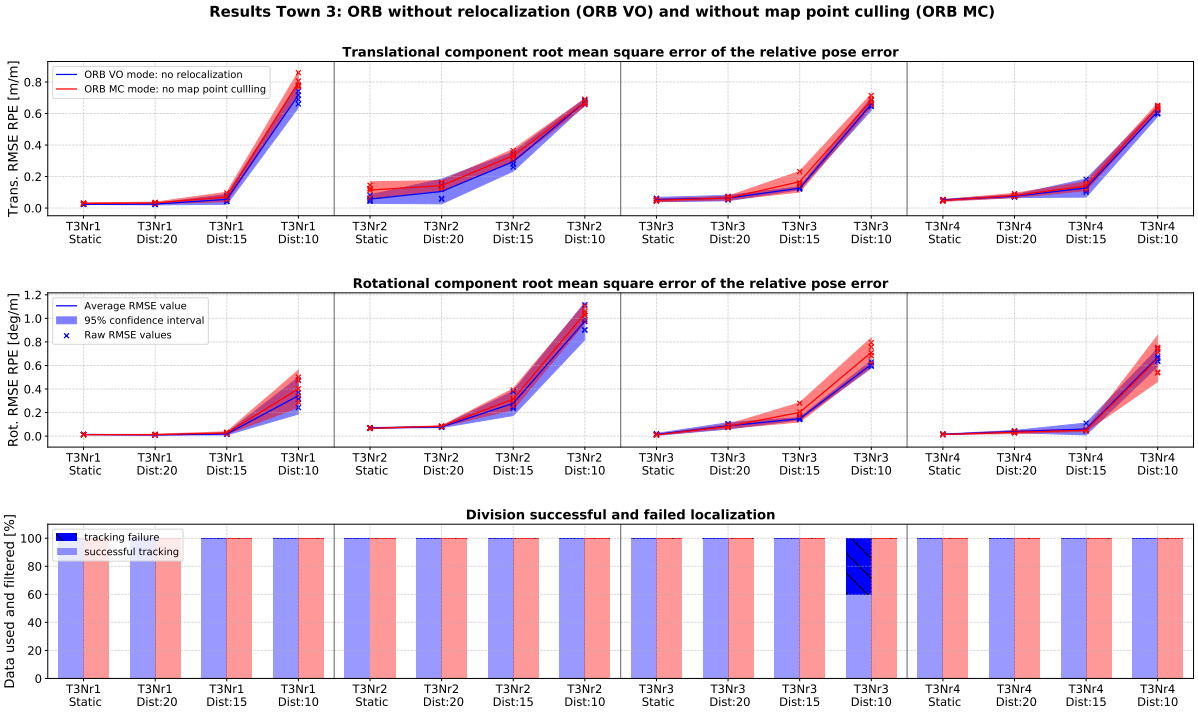
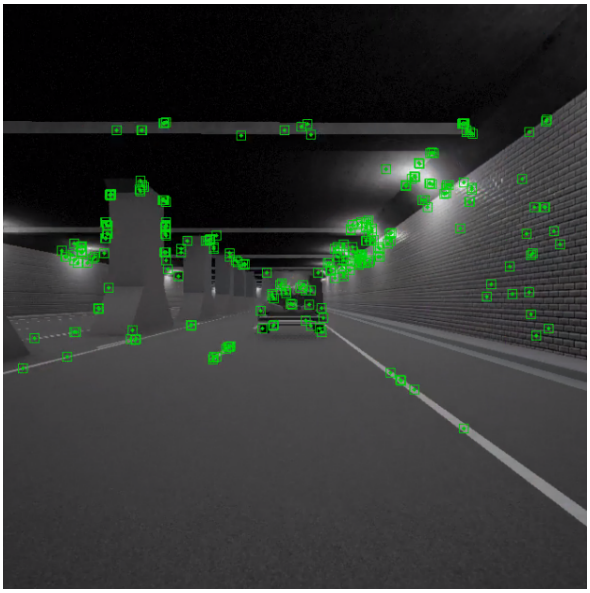


Figure 4.6: Performance of ORB VO and ORB MC in town 3. The pose error significantly increases at an inter-vehicle distance of 10 meters. Disabling the map point culling does not seem change the resulting pose estimation error.

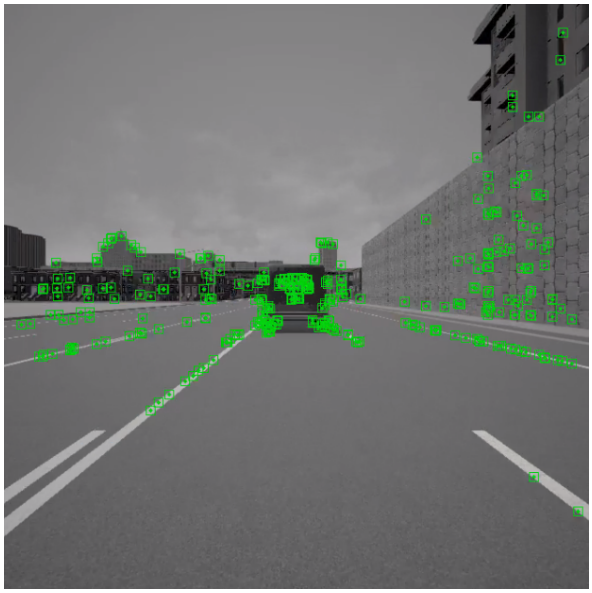
4.1.2. Analysis

ORB SLAM The main mode of failure of the original ORB SLAM algorithm when driving behind another vehicle is false relocalization. The closer the vehicle in front is to the ego-vehicle, the more likely it is that false relocalization occurs. According to figures 4.1, 4.2 and 4.3, the ego-vehicle is falsely relocalized 40 out of the 45 pose estimations with an inter vehicle distance of 10 meters. ORB SLAM describes images as a bag of words with a vocabulary of offline created ORB features [9]. As a large percentage of the image are features that describe the van in front (an example shown in figure 4.7a), it is likely that the bag of words vector has a close match with other images in the video sequence. The DBoW2 algorithm requires a high matching score over a long period of time to prevent that the algorithm matches similar looking environments. However, since the van is in front of the ego-vehicle for the complete trajectory, the bag of words will have a lot of similarities over time triggering false relocalization. From the result can be concluded that false relocalization is likely to occur when driving closely behind a vehicle over a long period of time due to the design of the place recognition module.

According to figure 4.3, the ORB SLAM algorithm will falsely relocalize at an inter-vehicle distance of 15 meters in town 3 at trajectory 1 and 2. This is odd, since false relocalization usually happens when the vehicle in front is closer 10 meters from the camera. Figure 4.6 shows the pose estimation when the algorithm would not relocalize in town 3. It illustrates that ORB VO is able to provide a reliable pose estimation in town 3 trajectory 1, which is a trajectory in a tunnel. Since the tunnel is a very repetitive environment, the place recognition module will match the image with a place it has visited before and will falsely relocalize the vehicle (an example of the environment is shown in figure 4.7a). This explains why this particular trajectory activates relocalization already at an inter-vehicle distance of 15 meters instead of 10 meters. Town 3 trajectory 2 is a wide road with difficult textured environments for the algorithm, shown in figure 4.7b. Large walls on the side of the road provide little texture to match features. Most of the features describe the vehicle in front, increasing the chance of false relocalization. The same phenomenon occurs with trajectory 4. Finally, according to figure 4.2 relocalization happens at static conditions in town 2 trajectory 2. This town is smaller and many buildings are the same, which would explain why the static environment triggers the relocalization. This might not seem realistic however in reality, some districts use the same design for many houses, which could recreate this behavior. These results suggest that the place recognition module needs a redesign before the algorithm can be applied in automotive applications.



(a) Image of ORB SLAM in town 3 trajectory 1 with an inter-vehicle distance of 15 meters. The long tunnel is a very repetitive environment, which can trigger the place recognition module



(b) Image of ORB SLAM in town 3 trajectory 2 with an inter-vehicle distance of 15 meters. The walls provide difficulty for pose estimation. The majority of the image describes vehicle in front, which triggers false relocalization

Figure 4.7: Demonstrating situations that cause false relocalization when a vehicle drives 15 meters in front of the ego-vehicle

ORB VO Figures 4.4, 4.5 and 4.6 show a noticeable difference in the magnitude of the pose error when comparing town 1 and 2 with town 3. For town 1 and 2, the error does not significantly increase when a vehicle drives in front of the cameras. However, a clear increase in error occurs when a vehicle drives in front of the ego-vehicle in town 3. The layout of the infrastructure could be an explanation for this. The static features are further away from the camera due to the wider roads, which could influence the accuracy of the translational component. Stereo ORB SLAM labels key points as "close" or "far" [19]. Close key points are safe to use for depth estimation, while far key points can be used to establish orientation. In this experiment setup, close key points are defined as 35 times the stereo baseline (0.54 meters), which is 18.9 meters. At an inter-vehicle distance of 15 meters, the vehicle in front is close enough to influence the translational pose estimation of the ego-vehicle. However, in town 1 and 2 there are plenty of close static landmarks which allows for a reliable pose estimation. In town 3, the static landmarks are further away. Therefore, when the vehicle in front is close, a high percentage of close key points are dynamic. This decreases the reliability of the pose estimation and results in either a higher error, false relocalization or tracking failure (see figure 4.6).

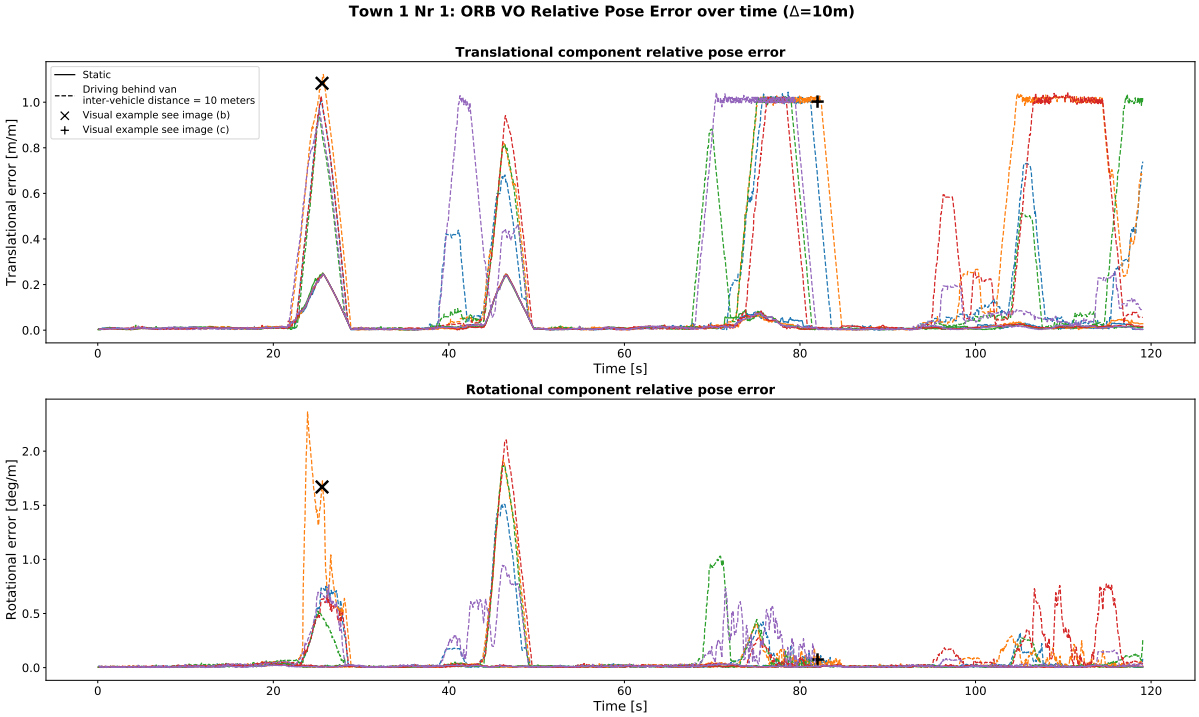
All trajectories see a significant increase in the average root mean square error at an inter-vehicle distance of 10 meters when the algorithm is in VO mode. Not only the average error, but also the standard deviation increases. This suggests that at 10 meters the static environment assumption is violated such that the pose estimation becomes unreliable. A closer analysis gives more information to when and why the error increases. Figure 4.8a shows the relative pose error off all ORB VO pose estimations over time of trajectory T1Nr1, evaluated over 10 meters instead of 100 meters, with an inter-vehicle distance of 10 meters. This allows to trace the origin of an error to a certain event in the trajectory, for example a right hand turn. There are three aspects that stand out. First, during the majority of the straight sections the pose error of the dynamic scenario are identical to the static scenario, demonstrating a robust behavior to a dynamic environment. The second and third aspect is that the figure shows two major sources of an increase in error: two large peaks at t= 25 s and t=45 s and two plateaus starting at t = 68 s and t = 102 s. Understanding this behavior is key in developing better stereo based V-SLAM methods for the automotive industry.

The lack of difference in performance between the static conditions and dynamic conditions on the straight road could be explained by the feature tracking method the algorithm uses. The tracking of features is assisted with a constant motion model [20], which the features from the vehicle in front violates. The features will therefore not be tracked and removed from the map. During a straight trajectory ORB features

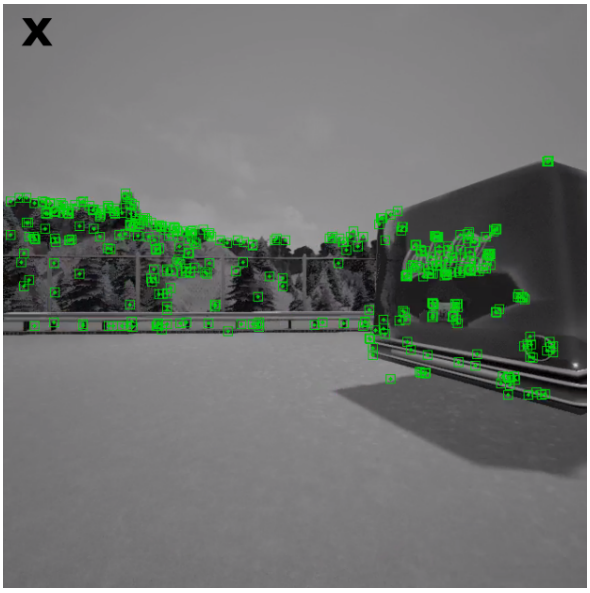
appear and disappear on the vehicle in front over time. When a new keyframe is inserted, the ORB features are redistributed over the image, where some of them select the vehicle in front. As the algorithm will try to track the features with the help of the motion model, it will loose track of the features that describe the vehicle. When enough features in the image have changed, a new keyframe is inserted and the ORB features on the vehicle will appear again. Since all the features that describe the vehicle in front are removed on a straight trajectory, the pose estimation will be very similar for the static and dynamic scenario.

The error peaks in figure 4.8a coincide with the two right hand turns in the trajectory. The increase in error is because the majority of the features describe the back of the van (see figure 4.8b), which does not adhere to the static environment assumption. As a result the pose estimation error increases. Why the algorithm suddenly tracks the vehicle in front could have several reasons. An explanation could be that the majority of the image represents the back of the van, so it is likely that the algorithm tracks the vehicle features. In corners, the front vehicle is actually closer in the simulation than during straights. This is because the inter-vehicle distance is defined by the distance between the waypoints. However, the magnitude between two waypoints in a corner is smaller than on a straight road. Another explanation could be that during corners the constant motion model is violated. This will cause the algorithm to broaden the search for matching features which allows it to match features from the van. What can be concluded that taken a turn with a vehicle closely in front of the cameras heavily increase the pose estimation error.

Another major factor in the increase of the average root mean square error are the two plateaus at t = 68 s and t = 102 s. During that time the majority of the ORB features select the vehicle in front and are "locked in", an example is shown in figure 4.8c. The majority of the features describe the van. The distance between the two vehicles remains constant, therefore the algorithm thinks the vehicle is standing still. Only when less than 90 % of the ORB features of the last keyframe are tracked, a new keyframe is inserted. Up and until that point, the algorithm thinks the ego-vehicle is standing still, which looks like the ORB features are "locked" onto the vehicle in front. This phenomenon causes large errors in the pose estimation.



(a) The relative pose error of ORB VO evaluated with Δ=10 m over town 1 trajectory 1 with an inter-vehicle distance of 10 meters. During straight roads the pose estimation error is identical for static and dynamic conditions. The peak errors occur during corners. The plateau errors at t = 68 s and t = 102 s occur when the ORB algorithm "locks" onto the vehicle in front. See figure 4.8c.



(b) At an inter-vehicle distance of 10 m, a large percentage of the image describes the vehicle in front. This does not only mean that a large percentage does not adhere to the static environment assumption. Also the chance of place recognition increases, which triggers false relocalization

(c) Image of ORB SLAM in town 1 trajectory 1 with an inter-vehicle distance of 10 meters. Illustration of ORB features locking onto the vehicle in front and therefore expecting the ego-vehicle to stand still.

Figure 4.8: Vehicle 10 meters in front of ego-vehicle. An in-depth analysis of the source of increase in pose estimation error.

ORB MC Figures 4.4 and 4.5 illustrate that bypassing the map point culling method that is integrated in the algorithm increases the root mean square error at town 1 and town 2 with an inter-vehicle distance of 10

meters. This suggests that the map point culling feature does increase the robustness to a dynamic environment. Generally, it can be seen that without the map point culling method, the algorithm tends to "lock" its features on the vehicle in front more often. This causes the ORB algorithm to think the vehicle is standing still, as explained in the previous paragraph. The map culling method removes the map points that describe the vehicle which were created in keyframes. When many keyframes are inserted (for example during a turning manoeuvre) these features are not removed when the culling method is disabled. An explanation why this "locking" occurs could be that the features that describe the vehicle are now in the area where the motion model is looking to match. When too many van features match, "locking" happens and the algorithm thinks the vehicle is standing still.

There is little to no difference in the performance of ORB VO when disabling the map culling algorithm in town 3 as can be seen in figure 4.6. An explanation could be because it is more difficult to gather close features as the roads are very wide, causing the majority of the ORB features to describe the vehicle in front. Stereo ORB SLAM requires a certain amount of close matched keypoints to provide a reliable translation estimate [19]. Therefore it is likely that on wide roads, the vehicle in front is selected. This causes the feature "locking" phenomena, increasing the pose estimation error as the algorithm thinks the vehicle stands still. Since this "locking" phenomena already happens often with the map culling feature on, disabling it would have little effect.

4.2. Vans driving on opposite road

In the second scenario the ego-vehicle encounters vans on the opposite lane. In total the ego vehicle encounters 17 vans. In this situation, only the dynamic scenario is compared to the static scenario. Each figure contains the translational and rotation component of the root mean square error of the relative pose error. All starting locations of all towns are represented in one figure. Furthermore, ORB VO is not tested since no false relocalization occurs and therefore the results of ORB VO will be similar to ORB SLAM.

4.2.1. Results

Figure 4.9 shows that in general the pose estimation error marginally increases when vehicles drive on the opposite lane. The magnitude of this error is much lower compared to the first dynamic scenario: driving behind another vehicle. Also note that no false relocalization occurs due to the dynamic conditions (except for town 2 trajectory 2, where false relocalization happens in static conditions).

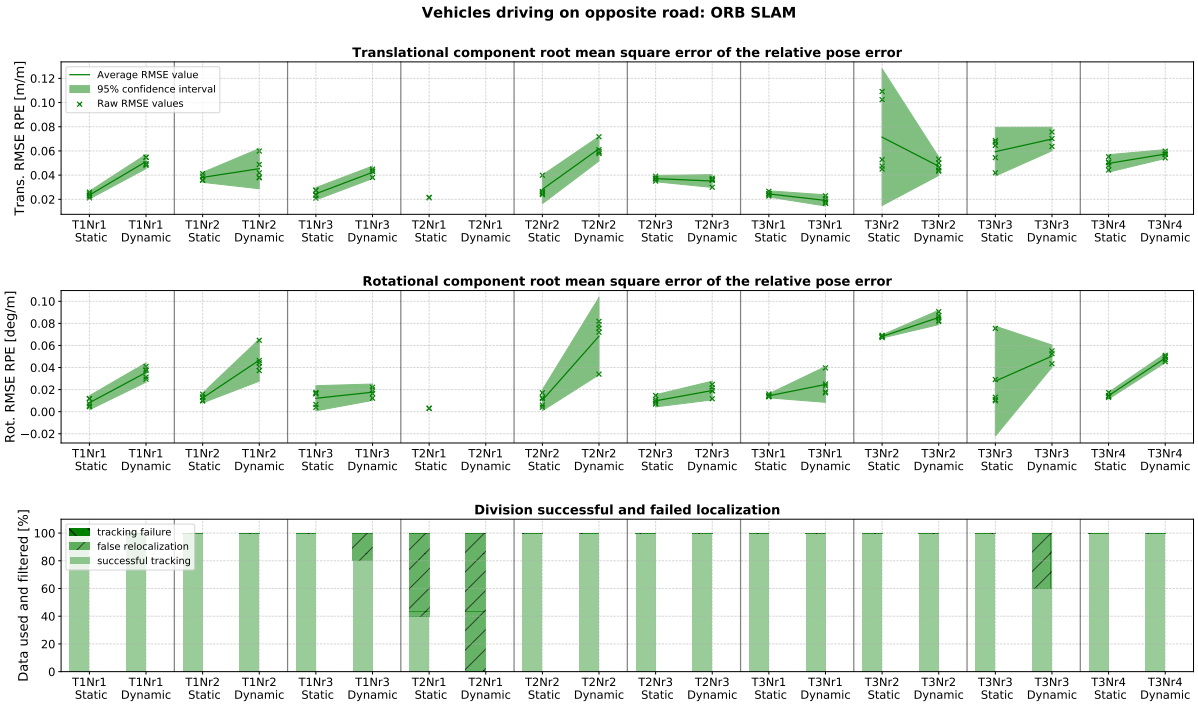


Figure 4.9: Performance of ORB SLAM when vehicles drive on the opposite road for all trajectories across the three towns. For the majority of the trajectories the error increases marginally.

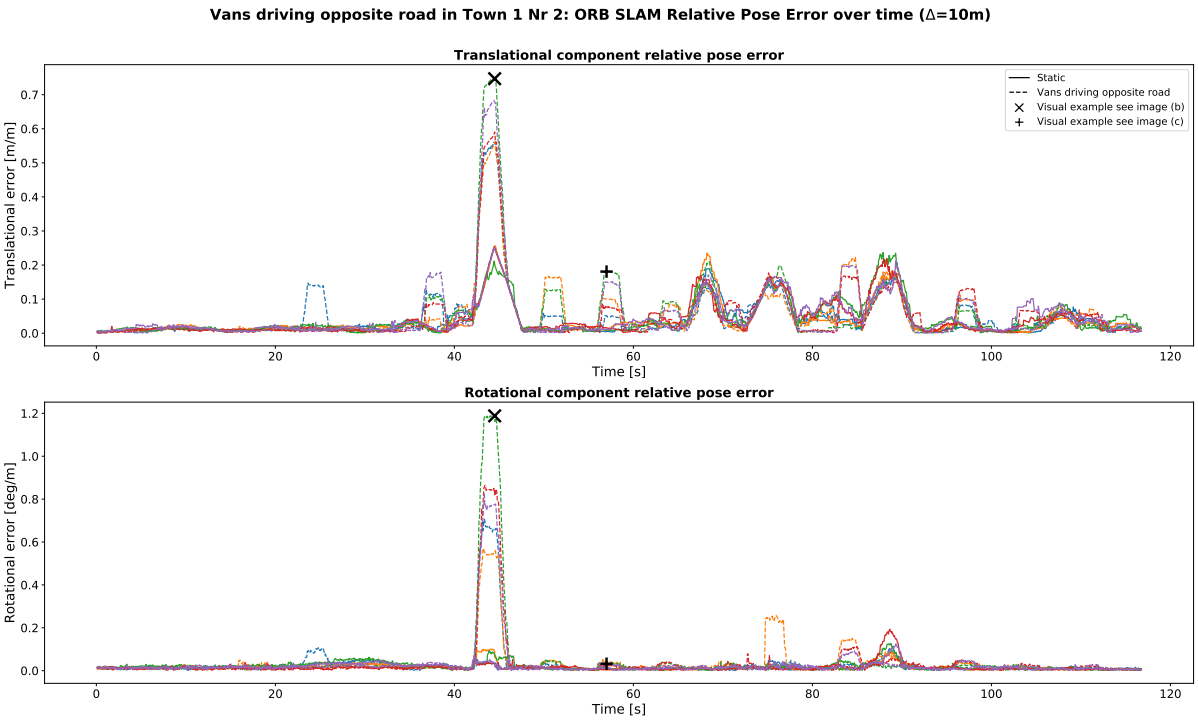
Figure 4.10 compares the performance of ORB SLAM to the performance of the algorithm without the map culling method when vehicles drive on the opposite side of the road. The figure illustrates that the variance marginally increases in some trajectories, while in other trajectories it does not seem to have effect.

Figure 4.10: Performance of ORB SLAM and ORB without map point culling method (ORB MC) with vehicles driving on the opposite road. The variance seems to slightly increase, although its significance can be argued.

4.2.2. Analysis

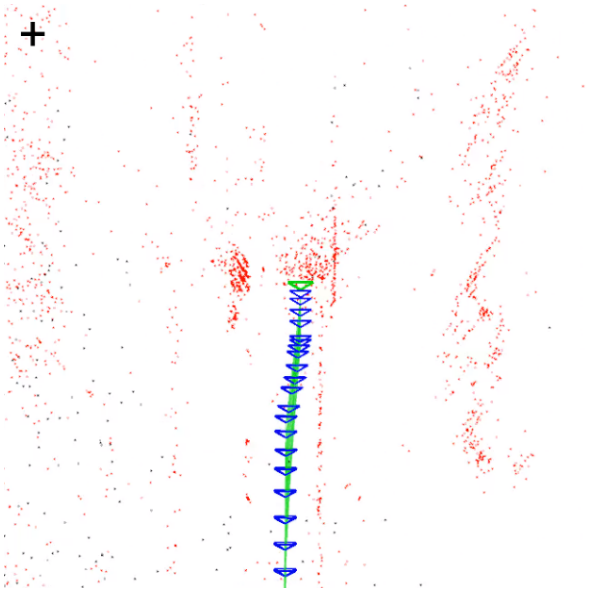
ORB SLAM Although the increase in error is marginal, a closer analysis in a single trajectory provides more insight in the behavior of the algorithm in dynamic conditions. Figure 4.11a shows the relative pose error over time of trajectory T1Nr2 evaluated over 10 meters. There is a remarkable peak in error at the first right hand turn, indicated with an x. During this right hand turn, a vehicle on the opposite road approaches to take a left hand turn, as can be seen in figure 4.11b. This explains why the error on the first turn in the dynamic scenario is larger than the errors in the turns after (see t= 68 s for example). As the algorithm establishes new keypoints during the turn, it mainly finds keypoints that describe the opposite vehicle. This increases the relative pose error, as the algorithm expects the keypoints to be static. This affects the translational component as well as the rotational component.

At the straight sections, figure 4.11a shows that sometimes the algorithm does have an increase in error due to an encountered vehicle and sometimes it does not. Figure 4.11c shows what happens in the map of ORB SLAM. As the vehicle comes closer more features describe the vehicle in front. The algorithm assumes that these features are static which causes the pose estimation to shift to the right. However, this phenomenon does not occur with every encountered vehicle. What could be possible is that the timing of a keyframe insertions plays a role. Keyframe insertion is a non-deterministic process and explains why not every run shows identical results. It could be that the new keyframe contains ORB features which describe the opposing vehicle, which are mapped in the local map. This local map is used to establish a pose estimation frame by frame. This will make it likely that the opposing vehicle features described in the image will match with the features described in the map, affecting the pose estimation. Note that the opposing vehicles on the straight road sections do not seem to effect the orientation estimation. This can be explained since many static keypoints are far away which are utilized to optimize for an orientation estimation. However, only the close keypoints are utilized for the translation estimation. Therefore the opposing vehicle only influences the translation approximation and barely influences the orientation estimation.



(a) Relative pose error evaluated over 10 meters for trajectory Town 1 Nr 2 comparing the performance of ORB SLAM in static conditions versus vehicles driving on the opposite side of the road. A large error peak occurs at t = 45 s caused by an opposing vehicle while taking a left hand turn (see figure 4.11b). Smaller translational error peak occurs when encountering a vehicle on the straight at for example at t = 57 s. (visual example in figure 4.11c.

(b) Source of the large error peak at t = 45 seconds. Many features describe the opposed moving vehicle which cause an error in the pose estimation, since the algorithm assumes matched features are static.



(c) Source of the translational error peak at t = 57 s. Image shows the estimated keyframe poses (triangles)and map point cloud (red dots) created by ORB SLAM. The estimated trajectory shifts to the right as the opposing vehicle comes closer. Algorithm assumes the matched features are static, which is the dense point cloud to the left of the last triangle. Therefore expects that the ego-vehicle is moving laterally.

Figure 4.11: Vehicles driving towards the ego-vehicle. An in-depth analysis of the source of increase in pose estimation error.

ORB MC Figure 4.12 illustrates the same trajectory, but now without the map culling method. The figure shows an increase in effect of the vehicles driving in opposite direction. Since the mapped features are not removed, it is more likely that features from the opposing vehicle in the image are matched with the mapped features. This influences the pose estimation negatively and therefore increases the error. This is the same effect discussed in the previous scenario.

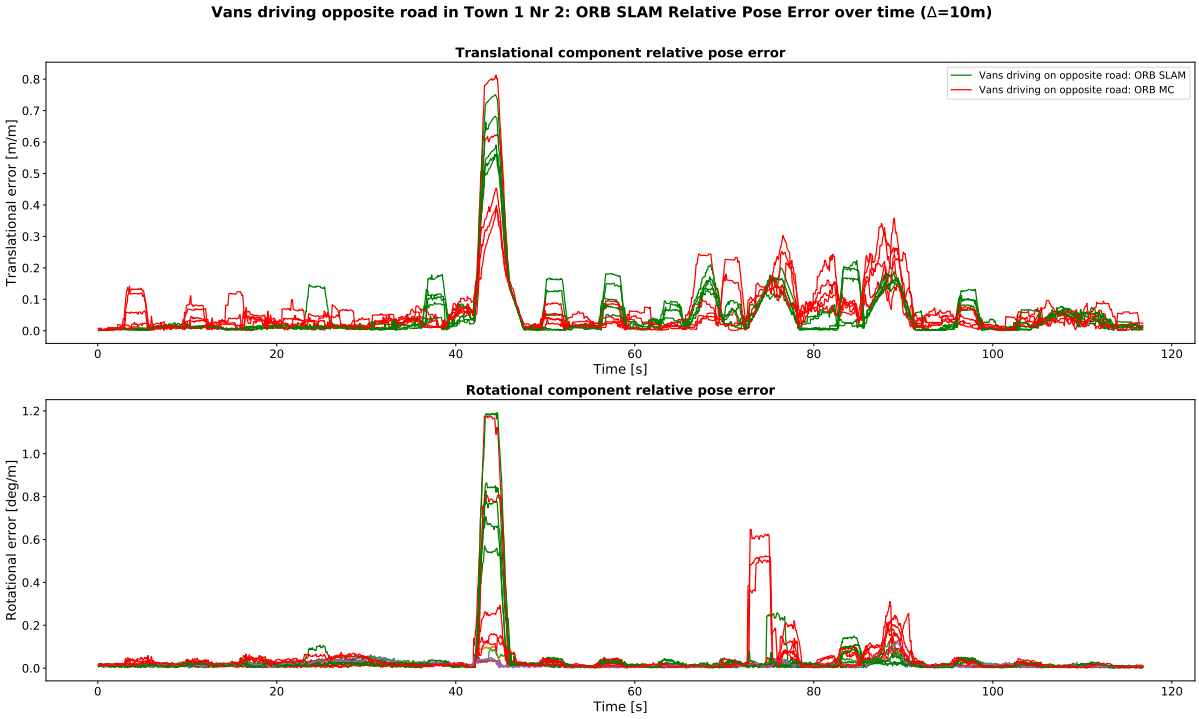


Figure 4.12: Relative pose error evaluated over 10 meters for trajectory Town 1 Nr 2 comparing ORB SLAM (green) with ORB MC (red) (map points are not culled). Translational error shows spikes in ORB MC in the beginning of the trajectory. This suggest that map point culling increases the robustness to a dynamic environment.

4.3. Localization only

ORB SLAM is able to utilize the maps that are produced in a previous SLAM session and localize itself within this map. When an accurate map is produced, any vehicle should be able to localize itself within this map. However, does the situation in the mapping process influence the pose estimation in localization mode? Three scenarios are used to answer this experiment: creating a map in static conditions, creating a map while a vehicle drives 15 meters in front and finally creating a map while vehicles drive towards the ego-vehicle in the opposite lane. The results represent the pose estimations of ORB SLAM in localization mode utilizing a map created in various scenarios.

4.3.1. Preliminary results

Figures 4.13, 4.14 and 4.15 show the performance of ORB SLAM in localization only mode with maps created under three different conditions: static environment, vehicle in front with an inter-vehicle distance of 15 meters and vehicles coming from the opposite lane. The results are the pose estimation from three trajectories from town 1. This is not enough data to give a definitive conclusion, but does provide an indication of the algorithms behavior. Therefore, this subsection is referred to as preliminary results. Each figure illustrates the pose estimation of a single scenario with maps created in various conditions, except the first black bar which illustrates the pose estimation when no map is used (the results from the ORB SLAM pose estimation).

Figure 4.13 shows the pose estimation in localization mode in a static scenario. The figure illustrates that the pose estimation error and uncertainty increases when a map is utilized under different circumstances. Figure 4.14 illustrates the pose estimation when a vehicle drives 15 meters in front. The figure shows that the algorithm can accurately localize itself when a map from either the same scenario or a static scenario

is utilized. However, the pose error significantly increases using a map that was created when vehicles are driving towards the ego-vehicle. Figure 4.15 shows the pose estimation when vehicles are driving towards the ego-vehicle. The figure shows similar pose estimation performance with maps created in all three scenarios.

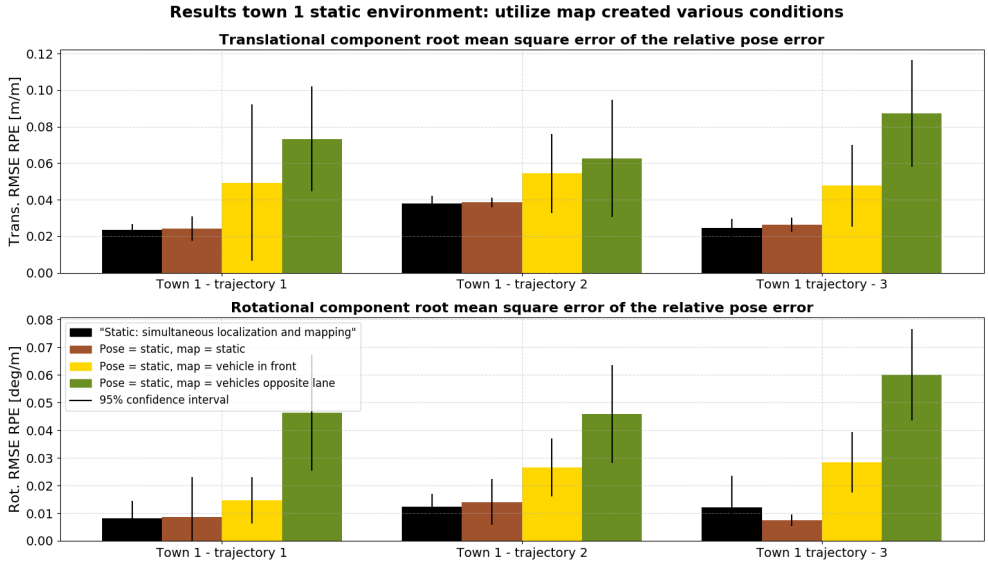


Figure 4.13: Pose estimation during static conditions utilizing a map that was made in various scenarios. The average pose estimation error and uncertainty increases when a map is utilized that was created under different scenarios.

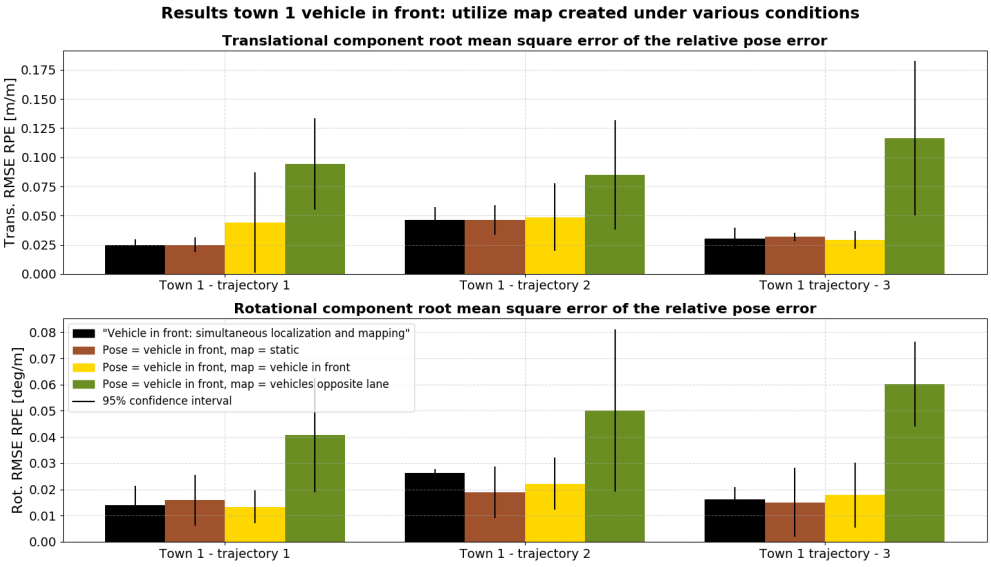


Figure 4.14: Pose estimation with a vehicle 15 meters in front utilizing a map that was made in various scenarios. Utilizing a map that was made in the same conditions or made in static conditions perform similar. However, utilizing a map when vehicles are driving towards the ego-vehicle has an increase in pose estimation error.

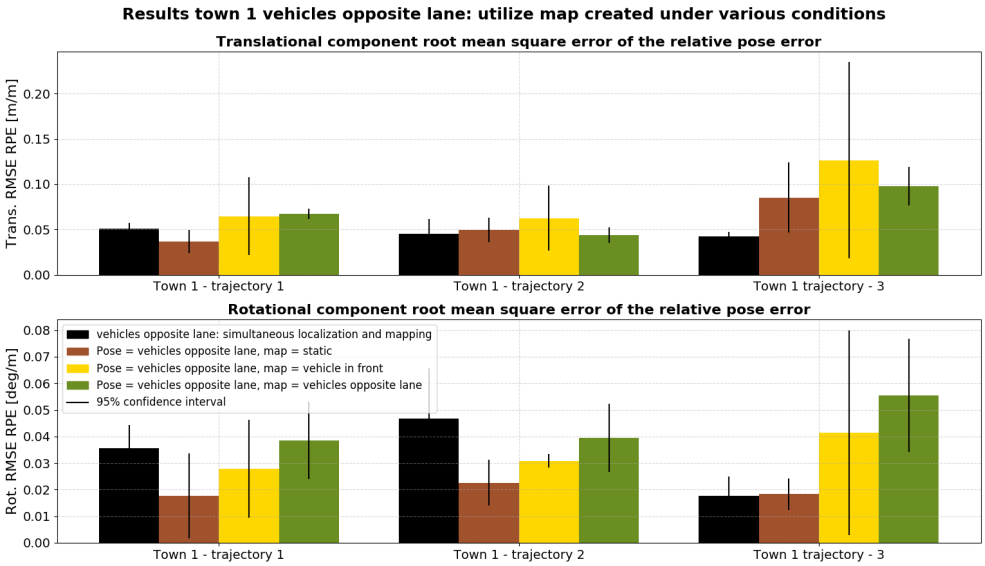


Figure 4.15: Pose estimation with vehicles driving towards the ego vehicle. The pose estimation performance seems very similar in the three created maps.

4.3.2. Analysis

Figures 4.13, 4.14 and 4.15 illustrate the pose estimation in localization mode with maps created under various conditions. The figures show that:

1. When driving in a static environment, having a map that is created in different conditions will increase the pose estimation error.

2. When driving behind a vehicle, having a map created in exactly the same scenario or in a static environment provides similar pose estimations. However, utilizing a map where vehicles are driving on the opposite lane will increase the pose error.

3. When driving with vehicles on the opposite lane, the environment of the map seems to have little effect.

The difficulty of the scenarios should be ranked in order to draw conclusion from these results. Figure 4.16 shows the pose estimation error of the three scenarios in SLAM mode. It shows that the performance of ORB SLAM are pretty similar in the three scenarios, but the error is slightly higher when vehicles are driving towards the ego-vehicle. The above observations suggest that you can utilize a map that was made under different circumstances as long as the map was created in an equal or less challenging environment. For example, you can use a static map when you try to estimate the pose when vehicles are driving towards you. However, your pose error will increase when you use a map that was created when vehicles are driving towards you and estimate a pose in static conditions. Note that this is an analysis on preliminary results, more data is needed to guarantee this conclusion.

Figure 4.16: The pose estimation from ORB SLAM during three different scenarios: static conditions, driving 15 meters behind a vehicle and vehicles driving on the opposite road. This indicates the general difficulty of the three scenarios compared to each other.

5 Conclusions

This research aims to develop an experiment setup that allows to measure the effect of an urban dynamic environment on a SLAM algorithm. The setup is tested with a stereo feature based V-SLAM algorithm called ORB SLAM [19]. SLAM frameworks assume a static environment to estimate the pose of the vehicle. The dynamic environment of intelligent vehicles affects the algorithms ability to localize the vehicle. The knowledge gained by this research helps the development of SLAM technology in an automotive environment. This chapter contain the conclusions that can be drawn from the results and analysis, a discussion on the effectiveness of the methodology and formulate recommendations for future work.

5.1. Conclusion

To get a deeper understanding on the effect of this static environment assumption the following main research question was formulated:

How does a dynamic urban environment influence the pose estimation accuracy of stereo ORB SLAM?

The dynamic urban environment was categorized into two distinct dynamic scenarios, leading to the following two sub-research questions:

1 *How does driving behind another vehicle influence the pose estimation accuracy of stereo ORB SLAM?*

2 *How do vehicles driving on the opposite lane influence the pose estimation accuracy of stereo ORB SLAM?*

Two other variations of ORB SLAM were tested on these scenarios to get deeper knowledge on how certain internal mechanisms influence the results. The first variation is ORB VO, which bypasses the place recognition module so the algorithm does not relocalize the vehicle. This allows insight in the behavior of the algorithm when the vehicle in front is close. The second variation is ORB MC, which bypasses the place recognition module and the map culling method. This allows insight in how this filter mechanism influences robustness to a dynamic environment. The two sub-research questions are:

3 *How do the dynamic scenarios influence the pose estimation accuracy of stereo ORB VO?*

4 *How does the map culling method influence ORB VO's pose estimation in the dynamic scenarios?*

Finally, ORB SLAM is tested in localization mode to research the effect of using a map that was created in different conditions. The final sub-research question is:

5 *How do the scenarios in which a map is created influence the pose estimation of ORB SLAM in localization mode?*

Driving behind another vehicle For the first dynamic scenario (driving behind another vehicle), the three variations of ORB SLAM are tested on 10 different trajectories with a vehicle driving in front of the ego-vehicle. The distance between the two vehicles are varied between 20, 15 and 10 meters and compared to a static baseline. For the ORB SLAM algorithm (without any adaptations) it can be concluded that driving behind a

vehicle triggers the the place recognition module. It falsely relocalizes the vehicle when the vehicle in front is somewhere between 10-15 meters away from the stereo camera setup. This behavior demonstrates the algorithm's fragility for this specific scenario.

If relocalization is bypassed (ORB VO), the pose estimation error generally increase as the vehicle in front is closer to the cameras. However, the magnitude of this error is dependent on how close the static landmarks are to the ego-vehicle. When many static landmarks are close to the ego-vehicle and the distance between the two vehicles is 15 meters or larger, the error does not increase significantly. However, when the static landmarks are too far to give a proper depth estimate, the translation error can increase noticeably. When the inter-vehicle distance is decreased to 10 meters, the pose estimation error increases drastically. This behavior happens in all three towns. The origin of this error lays in corner manoeuvres and a "locking" phenomena. The algorithm is surprisingly robust when both vehicles drive straight. However, the error increases severely when the ego-vehicle takes a turn while the vehicle in front is in the image. The majority of the features describe the vehicle in front, negatively influencing the pose estimation. Furthermore, at this distance of 10 meters, the algorithm tends to "lock" the majority of the ORB features on the vehicle in front. "Locking" happens when the algorithm estimates that the ego-vehicle is standing still because the distance between the two vehicles remains constant and most features describe the vehicle in front. It looks like the features are "locked" on the vehicle in front. The features are redistributed when a new keyframe is inserted, which means 10 % of the tracked features need to change. However, the ego-vehicle could have travelled a significant distance before the redistribution of features happen. This increases the pose estimation error drastically.

ORB MC bypasses the map culling module and place recognition feature. This sees a significant increase in pose estimation error when the vehicle is 10 meters in front of the cameras and static landmarks are close. Without the map culling, the algorithm tends to "lock" more on the vehicle in front, since map points that describe the vehicle in front are not removed. The algorithm will match these map points with the vehicle features of the current image. This will allow the algorithm to track these features and therefore "locking" onto the vehicle in front. This suggests that the map culling method does increase the robustness of the algorithm to a dynamic environment.

Vehicles driving on the opposite lane The second dynamic scenario (vehicles driving on the opposite lane) does not influence the pose error as significant. The algorithm illustrates robust behavior to this situation. The dynamic scenario does not cause any relocalization problems. Furthermore, the dynamic scenario causes only a slight increase in the pose error over the full trajectory. The source of the error can be traced back to encountering other vehicles on the opposite road. However, on straight roads this phenomena does not happen structurally with every encountered vehicle. It suggests that the timing of keyframe placement plays role in this behavior. When a keyframe is inserted and an opposing vehicle is close, many features of the opposing vehicle are mapped. As the vehicle comes closer, the vehicle features are tracked causing the pose error to increase, since it expects that the tracked features are static. However, when a keyframe is inserted and the opposing vehicle is far away, no dynamic features are tracked and the opposing vehicle has little influence as it is driving by. The error increases drastically when an opposing vehicle is encountered during a turn. The majority of the features will describe the opposing vehicle and negatively influence the pose estimation.

When the map culling method is disabled, the likelihood becomes larger that the pose error increases when it encounters a vehicle on the opposite road. The culling method normally removes the opposing vehicle features, since they do not match over three keyframes. However when map points are retained, the likelihood that the opposing vehicle features are matched increases, resulting in a larger pose error. This suggest that also in this scenario, the map culling method influences the algorithm's robustness to a dynamic environment.

Localization only The experiment that was performed in localization mode did not provide sufficient data to give a definitive conclusion. The data can be interpreted to give an indication of what to expect, but more research is needed to confirm the hypothesis. The data suggests that a map can be created in a different scenario as long as the current environment is less challenging compared to the mapped environment. Hence, the pose estimation error will be defined by the most challenging situation in either the mapping process or localization process.

Concluding, ORB SLAM without loop closure shows robustness to an urban dynamic environment when

sufficient static landmarks are close to the ego-vehicle. In these conditions the algorithm is able to provide relative accurate pose estimations when vehicles are driving towards the ego-vehicle. On this stereo setup, ORB SLAM is robust to driving behind another vehicle with an inter-vehicle distance equal or larger than 15 meters. The algorithm shows robustness on straights roads, even when decreasing the inter-vehicle distance to 10 meters.

However, ORB SLAM needs improvement on several aspects before applying it on intelligent vehicles.

- The place recognition module is too sensitive causing false relocalization, especially when driving behind another vehicle.
- The algorithm is very unstable during turning manoeuvres. The pose estimation error increases drastically when moving objects occur during a turn.
- A solution should be designed to prevent features to "lock" onto the vehicle in front.
- Creating a map in a dynamic environment influences the pose estimation in localization mode.

The designed methodology demonstrates to be valuable for an in-depth analysis of the SLAM performance in a dynamic environment. The experiment and analysis allows to exactly identify the points of improvement for the algorithm if it is to be used by intelligent vehicles. The methodology is flexible enough to perform similar research on other type of SLAM algorithms.

5.2. Discussion

The methodology in this research has successfully utilized a simulated environment that completely isolates the effect of a dynamic urban environment on the performance of a SLAM algorithm. The designed set-up provides constant weather conditions, constant lighting conditions and most importantly allows the ego-vehicle to be exactly at the same position under different dynamic scenarios. Due to these constant conditions, a detailed analysis can be performed on the resulting pose estimation in which the strengths and weaknesses of the algorithm are exposed. The designed experiment can be utilized to test the robustness of monocular, stereo and LIDAR based SLAM algorithms to a dynamic urban environment. This research focuses on the analysis of a stereo feature based V-SLAM method, but the experiment can be performed with other sensors thanks to the elaborate sensor suite of CARLA [6]. Furthermore, for testing monocular and stereo based V-SLAM methods, researchers would only need the ground truth file and stereo rosbags to test their algorithms robustness and would not need to install CARLA. However, since the experiment is in a simulation, the actual relevance to the real world can be questioned. It is important to discuss and understand the realistic and unrealistic aspects of this experiment, so results can be interpreted correctly.

The general design of the experiment does not completely represent the use of ORB SLAM in an intelligent vehicle. Doing the experiments in CARLA provides a lot of practical advantages, but the simulator does have an influence on the results. The design of CARLA influences the results on several factors, which could have an effect on how ORB SLAM behaves in the real world. The following aspects of the experiment should not influence the results or should improve the results.

For the performance of ORB SLAM, the quality of the simulation might not actually be that important. What is important in the simulation is the lighting conditions and the simulated reflectivity of materials. ORB features select keypoints by finding corners in an image based on light intensity and describes them in a bit-string [26]. The simulation should stimulate real life behavior from the algorithm as long as the intensity of the sun and the physics of reflection are simulated realistically. Unreal Engine provides extraordinary results regarding these light and reflection simulation as can be seen in figure 5.1. The image illustrates that the vehicle in front shows reflections of the environment on the vehicle itself, accurate shadow simulation and light flares in the camera caused by the sun. Although humans clearly see that the image is from a simulation, ORB SLAM only utilizes pixel intensity and the simulation should be close enough to reality to provide reliable results.

In reality, ORB SLAM should be more accurate than the results now suggest. As section 3.2 explains, the camera focal points are estimated by assuming a thin fixed focal length lens. The camera intrinsic values are calculated from this assumption. When a real camera is used, Zhang's calibration method [34] is utilized to estimate the camera's intrinsic values. This requires a chessboard pattern in front of the camera, which is not a feature in CARLA and therefore requires to make assumptions. The overall pose estimation accuracy is expected to increase when the calibration method is used. The calibration method will provide more accurate

Figure 5.1: Example image of CARLA simulation that illustrates the realistic lighting and reflection physics of the simulator. These are important aspects for ORB SLAM which rely on pixel intensity to provide a pose estimations.

camera intrinsic values, which will result into more accurate projection functions in the bundle adjustment of ORB SLAM (see equation 3.3).

The simulation setup has great influence on the application relevance in the real world. The following aspects limit the relevance to reality. For the first dynamic scenario, the velocity of both vehicles are set to a constant speed of 15 km/h. In a simulation this results in a constant inter-vehicle distance. In reality this distance will slightly vary, since driving at constant speed is rather difficult. This might lead to different results in the dynamic scenario where the vehicle drives in front when the inter-vehicle distance is 10 m. Performing this scenario in real life should influence the frequency of the "locking" phenomena. In reality it is more likely that the ORB SLAM features redistributes since the distance between the two vehicles will not be constant. The resulting pose estimation error should be lower.

The tested velocity in this experiment is rather slow: 15 km/h. Therefore, the results are optimistic when compared to real world applications. Increasing the velocity provides a new set of problems. Mostly because the designed simulation does not incorporate any vehicle dynamics. Therefore driving at a higher velocity would not guarantee that turning manoeuvre would actually be possible. Furthermore, driving at a low speed guarantees a lower angular rate which increases the performance of ORB SLAM. CARLA does provide an autopilot which obeys traffic rules and lowers its speed when driving a corner. However, it does not allow control of the vehicle. This means that simulating specific dynamic scenarios would be impossible and therefore this function was not utilized. The effect of the velocity in different dynamic conditions would be an interesting topic to research.

The length of the stereo baseline has an effect on the results. The results show a difference between the effect of certain scenarios in town 1 and 2 compared to town 3. This could be because of what the algorithm defines as "close" and "far" stereo points, which is dependent on the distance between the two cameras. Increasing the length of the stereo baseline should allow the vehicle to provide the depth estimate of features that are further away from the camera. This should increase the robustness of the algorithm to a dynamic environment in town 3.

Finally, only very specific scenarios are simulated in CARLA, since the experiment is designed to isolate the effect of certain conditions. It would be difficult to trace the source of performance decrease when more complex environments are tested. The simulation therefore contains unrealistic constant environments, for example the same type of van passing over and over again on the other side of the lane. In real life, more complex scenarios would occur which would include different type of road users, like cyclists and pedestrians. It was assumed that large vehicles would have the biggest impact on the results of ORB SLAM. The effect of different road users and more complex scenarios should be tested in the future.

5.3. Future work

SLAM algorithms play an important role in the development of fully autonomous vehicles, as they provide the information necessary to plan a safe trajectory. Important knowledge was gained during this research which aids the development of SLAM algorithms in intelligent vehicles. A deeper analysis of stereo ORB SLAM in dynamic conditions was performed which helps the development of robust stereo V-SLAM algorithms in urban environments. Moreover, the research illustrated the feasibility of utilizing the open source simulator CARLA for essential computer vision research in the automotive industry. The results from the experiment allow interesting research opportunities which would push the development of SLAM applied on intelligent vehicles.

The data for the localization mode experiment is currently too little to draw a definitive conclusion from. Currently, the data only represent the three trajectories from town 1, the other two towns remain untested. A definitive conclusion can be drawn if the maps and pose estimation data also includes the seven other trajectories from the other two towns. Another interesting proposition is to test whether the pose estimation improves if the vehicle drives a normally difficult scenario in localization mode with an accurate predefined map. For example, create an accurate map by driving a trajectory in a static scenario multiple times. Then utilize this map in localization mode to localize the vehicle in a difficult dynamic scenario. Furthermore, the maps in the experiment were created in a single sweep SLAM session. It is recommended to do multiple SLAM sweeps to create reliable maps. It would be interesting to test whether the pose estimation becomes more reliable when the map is created in an array of different conditions.

Verification of the results remain an open issue and should be tested before more experiments are performed in the simulator. The general patterns that stereo ORB SLAM displays in the simulator should also appear in reality. A vehicle with a stereo camera setup should drive behind a vehicle, varying the distance between the two vehicles. Essential is that the experiment shows similar behavior as the simulated experiment. At an inter-vehicle distance of 10 meters ORB SLAM should show false relocalization, ORB VO should demonstrate a strong increase in error during corners and the features should tend to "lock" onto the vehicle in front. If this is the case, the results in this research are verified and more research can be done in the simulator.

The gained knowledge in the strength and weaknesses of ORB SLAM in an urban environment can be used to develop solutions which would increase the algorithm's pose estimation accuracy in dynamic environments. This research has shown specific points of improvement to increase the robustness for stereo ORB SLAM in an urban dynamic environment. The same experiment can be utilized to test the effectiveness of these solution. Examples of specific solutions that would aid the robustness of ORB SLAM in a dynamic environment are improving the place recognition module so false relocalization is prevented or preventing feature selection on close moving vehicles.

This research has tested the robustness of stereo ORB SLAM in dynamic scenarios, but under very specific conditions. There is a wide array of variables that can be tested which would increase our understanding of ORB SLAM and aid the development of V-SLAM in intelligent vehicles. Examples of variables could be testing the effect of the driving velocity, different weather conditions or camera baseline length. Other aspects would be to test different dynamic scenarios which could include pedestrians and cyclists.

The methodology used in this research is unique in testing the robustness of SLAM algorithms to a dynamic environment. Since CARLA has an elaborate sensor suite, LIDAR SLAM algorithms could use the same methodology to isolate the effect of certain dynamic scenarios to the pose estimation. Other V-SLAM algorithms only need to have access to the rosbags and ground truth files to test their robustness and do not even need to download CARLA. CARLA can be used as a powerful tool in the development of SLAM algorithms for intelligent vehicles.

This research not only highlighted the points of improvement for stereo ORB SLAM, but also highlighted the strong points of the algorithm. By removing the features from the map that are not detected in consecutive keyframes the error was significantly reduced. This insight could help the development of other SLAM methods to improve on the pose estimation accuracy in dynamic environments.

This research has developed a flexible experiment setup that completely isolates the effect of a dynamic urban environment on the performance of a SLAM algorithm. The methodology will aid in obtaining a deeper understanding on the strength and weaknesses of SLAM in dynamic conditions, since SLAM frameworks assume a static environment. The experiment setup can be utilized to analyze robustness to a moving environment of monocular, RGB-D, stereo V-SLAM algorithms as well as LIDAR based SLAM methods. The analysis in the strength and weaknesses of ORB SLAM is an example of the depth of the research that is possible with this

methodology. The lesson learned from this analysis can be utilized to improve ORB SLAM or develop even better V-SLAM algorithms. The methodology and the lessons learned from the results could be an important tool in the development of SLAM for automotive applications.

Bibliography

[1] A. Bonarini, W. Burgard, G. Fontana, M. Matteucci, D.G. Sorrenti, and T.D. Tardós. Rawseeds: Robotics advancement through web-publishing of sensorial and elaborated extensive data sets. *Intelligent Robots and Systems (IROS) Workshop on Benchmarks in Robotics Research*, 2006.

[2] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The euroc micro aerial vehicle dataset. *The International Journal of Robotics Research*, 35:1157–1163, 2016.

[3] M. Calonder, V. Lepetit, C. Stretcha, and P. Fua. Brief: Binary robust independent elementary features. *European Conference on Computer Vision*, 11:778–792, 2010.

[4] I. Cvišić and I. Petrović. Stereo odometry based on careful feature selection and tracking. *2015 European Conference on Mobile Robots (ECMR)*, pages 1–6, 2015.

[5] X. Ding, Y. Wang, H. Yin, L. Tang, and R. Xiong. Multi-session Map Construction in Outdoor Dynamic Environment. *ArXiv e-prints*, jul 2018.

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. *CoRR*, 2017.

[7] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for monocular camera. *2013 IEEE International Conference on Computer Vision*, pages 1449–1456, 2013.

[8] J. Engel, J. Stückler, and D. Cremers. Large-scale direct slam with stereo cameras. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942, 2015.

[9] D. Galvez-López and J. D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[10] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968, 2011.

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[12] A. Geiger, P. Lenz, and R. Urtasun. Visual odometry/ slam evaluation. `http://www.cvlibs.net/datasets/kitti/eval_odometry.php`, 2018. Accessed 11-07-2018.

[13] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[14] Edmund Optics Inc. Understanding focal length and field of view, Nov 2018. `https://www.edmundoptics.com/resources/application-notes/imaging/understanding-focal-length-and-field-of-view/`.

[15] M. Kaneko, K. Iwami, T. Ogawa, T. Yamasaki, and K. Aizawa. Mask-slam: Robust feature-based monocular slam by masking using semantic segmentation. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[16] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: A general framework for graph optimization. *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011.

[17] Mat_Schroeder25. Onshape discussion forum: Is it possible to change the orientation of the default geometry, May 2017. `https://forum.onshape.com/discussion/6555/is-it-possible-to-change-the-orientation-of-the-default-geometry`.

[18] R. Mur-Artal and J.D. Tardos. Fast relocalisation and loop closing in keyframe-based slam. *2014 IEEE International Conference on Robotics and Automation*, 2014.

[19] R. Mur-Artal and J. D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017.

[20] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015.

[21] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem. Sim4cv: A photo-realistic simulator for computer vision applications. *International Journal of Computer Vision*, 2018. URL `http://dx.doi.org/10.1007/s11263-018-1073-7`.

[22] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. *2011 International Conference on Computer Vision*, pages 2320–2327, 2011.

[23] F. Pomerleau, P. Krüsi, F. Colas, P. Furgale, and R. Siegwart. Long-term 3d map maintenance in dynamic environments. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3712–3719, May 2014.

[24] S.R. Richter, Z. Hayder, and V. Koltun. Playing for benchmarks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2232–2241, 2018.

[25] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, 2:1508–1515 Vol. 2, 2005.

[26] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. *2011 International Conference on Computer Vision*, pages 2564–2571, Nov 2011.

[27] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *CoRR*, abs/1705.05065, 2017. URL `http://arxiv.org/abs/1705.05065`.

[28] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 2012.

[29] Chieh-Chih Wang, C. Thorpe, and S. Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas. *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, 1:842–849 vol.1, Sept 2003.

[30] R. Wang, M. Schwörer, and D. Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3923–3931, 2017.

[31] J. Zhang and S. Singh. Lidar odometry and mapping in real time. *Robotics: Science and Systems Conference (RSS)*, pages 1935–1942, 2014.

[32] J. Zhang and S. Singh. Visual-lidar odometry and mapping: Low-drift, robust and fast. *IEEE International Conference on Robotics and Automation*, pages 2174–2181, 2015.

[33] J. Zhang, M. Kaess, and S. Singh. Real-time depth enhanced monocular odometry. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4973–4980, Sept 2014.

[34] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334, December 2000. URL `https://www.microsoft.com/en-us/research/publication/a-flexible-new-technique-for-camera-calibration/`. MSR-TR-98-71, Updated March 25, 1999.

[35] Guoxiang Zhou, Berta Bescós, Marcin Dymczyk, Mark Pfeiffer, José Neira, and Roland Siegwart. Dynamic objects segmentation for visual localization in urban environments. *CoRR*, abs/1807.02996, 2018. URL `http://arxiv.org/abs/1807.02996`.