An Experimental Evaluation of Learning-Based Methods for Loop Closure Detection in Simultaneous Localization and Mapping

by

Luis Fernando Herrera Arias

S.B. Computer Science and Engineering
Massachusetts Institute of Technology, 2020

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2021

Author . . . .
Department of Electrical Engineering and Computer Science
July 21, 2021

Certified by . . . .
Luca Carlone
Leonardo Career Development Associate Professor
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

An Experimental Evaluation of Learning-Based Methods for Loop Closure Detection in Simultaneous Localization and Mapping

by

Luis Fernando Herrera Arias

Submitted to the Department of Electrical Engineering and Computer Science on July 21, 2021, in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

Simultaneous Localization and Mapping (SLAM) is the capability to estimate a robot's trajectory in an initially unknown environment while reconstructing the geometry of the environment. In order to bound the accumulation of localization error in SLAM, it is crucial to recognize previously seen locations, a process called "loop closure." This allows the robot to make corrections to its localization and map estimates. This project evaluates ORB feature extraction and matching, a state-of-the-art technique to detect loop closures, against recently developed learning-based approaches. In particular, our first contribution is to benchmark established techniques based on hand-crafted descriptor matching against novel learning-based approaches based on neural networks (i.e., SuperPoint and SuperGlue). As a second contribution, we integrate a learning-based loop closure detection method as part of Kimera, a SLAM system, and demonstrate its performance in both simulated and real benchmarking datasets. Finally, we collect data on long trajectories using a Jackal robot to compare the different approaches on real-world situations beyond available datasets. Our evaluation shows that, while learning-based approaches detect many more loop closures across wider baselines, when integrated in a SLAM system, they do not lead to substantial performance improvements compared to standard ORB feature matching.

Thesis Supervisor: Luca Carlone
Title: Leonardo Career Development Associate Professor

Acknowledgments

For –

my loving and supporting partner, Josh, who was always there;

my grandma, Maria, who believes in everything I do;

my friends, Vibha, Asia, Max, and Nikhil.

Para -

mi tita, Maria, que siempre cree que puedo lograr cualquier cosa.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mapping and 3D reconstruction are foundational problems and have attracted interest across multiple research communities, including computer vision, robotics, and machine learning. Geometric reconstruction is critical for a robot's understanding and navigation of its environment. A simultaneous localization and mapping (SLAM) system allows a robot to create a global map of an initially unknown environment while maintaining an estimate of its position within it. When we combine maps of an environment that are built by two or more robots that are simultaneously exploring and communicating with each other, we obtain a multi-robot SLAM system.

Over time, any SLAM system mapping an unknown environment inevitably accumulates error on its estimated trajectory. Loop closures provide the ability to recognize a previously seen location in the internal map, then use the robot's current position and its position at the time the location was previously seen to make corrections to the accumulated error in the estimated trajectory. For this reason, loop closure detection constitutes one of the key components to ensure that the localization and mapping errors remain bounded. One common method to reduce the error accumulation is to have a robust place recognition module that detects loop closures, and a pose graph optimization method that corrects the entire trajectory using loop closure inliers [4]. In the absence of loop closures, the localization and mapping results would diverge over time. This is why improvements to the loop closure capabilities could realize significant improvements to the entire SLAM system.

Place recognition has its own challenges, such as occlusions, differences in illumination, and perceptual aliasing between similar looking locations. Progress has made it possible to develop the state-of-the-art algorithms that we currently rely on, such as ORB for feature extraction and feature matching [5], and Bag-of-Words to quickly consider candidates for place recognition [6]. But recent developments in machine learning approaches have produced bold new methods that promise to outperform the classical ones. In this thesis, we evaluate the use of learning-based approaches in loop closure detection. In particular, we benchmark the current state-of-the-art and novel learning-based approaches including SuperPoint [1] and SuperGlue [2]. We integrate such approaches in a SLAM system and demonstrate its performance in simulated and benchmarking datasets. Finally, we evaluate the performance of both approaches on new data collected on a Jackal robot. The data collected on the Jackal was also used to test a multi-robot architecture, called Kimera-Multi [7].

This thesis begins with a literature review of all the systems we integrated and worked with, including Kimera, Kimera-Multi, SuperPoint, SuperGlue, ORB, and Graduated Non-Convexity [8] for outlier rejection. The rest of the thesis contains 3 experimental chapters describing the series of experiments we did, as well as evaluations and limitations of the tested approaches. The experimental evaluation is organized as follows:

1. Chapter 3 presents an overview of the learning-based approach, expanding on the literature review in Chapter 2. We describe the difference between the classical methods, such as ORB, and the learning-based approach. We then evaluate the results from both methods on benchmarking and simulated data. To be specific, we focus on measuring the *diversity* of the loop closures these methods can detect, as well as the *error* of each loop on the trajectories when comparing against the group truth. Our results in this chapter show improvements in the overall number of loop closures that the learning-based approaches

are able to find.

2. Chapter 4 expands on the previous results by comparing the same metrics post outlier rejection. The overall SLAM system must be robust to outliers while being accurate. We must accept correct loop closures that will decrease the drift while rejecting the ones that would increase our absolute trajectory error. To that end, we tested the effects of the Graduated Non-Convexity approach (GNC) [8] for robust fitting, which has proven to be robust to high ratios of outliers. In this chapter, we also compare the absolute trajectory errors for our simulated and benchmarking datasets while using both ORB and learning-based approaches for loop closure detection. Surprisingly, these results do not suggest that replacing the classical methods with the learning-based approaches produces significantly better results.

3. Chapter 5 is about real robot data collected on a Jackal at MIT and at the Medfield Hospital in Massachusetts. We did 6 different long runs on 2 locations (3 runs on each) and returned the robot to its original position. To measure accumulated error, in the absence of ground truth, we calculate the end-to-end drift. Furthermore, we visualize the trajectories overlaid on a satellite map as well as the corresponding mesh reconstructions. This chapter focuses on how the approaches perform on real data. While these results do not change the conclusions from Chapter 4, the experiments did provide an opportunity to exercise a multi-robot system, Kimera-Multi [7]. These experiments led to a joint paper, whose preprint can be found at [9].

We conclude the thesis with a brief discussion on the current limitations of the learning-based approaches. We also describe future work that could improve the performance of such approaches in SLAM systems. Our evaluation shows that, while SuperPoint and SuperGlue detect many more loop closures with wider baselines, when integrated in a SLAM system, they do not lead to substantial performance improvements compared to ORB-based methods.

Chapter 2

Related Work

A major contribution of this thesis is the testing and evaluation of learning-based and classical approaches for feature extraction and feature matching for loop closure detection in a SLAM system. In particular, we test ORB [5] against SuperPoint [1] and SuperGlue [2] using Kimera [3] as a SLAM pipeline. We will go into detail on all these techniques in this chapter. We also mention the outlier rejection technique used for experiments in the later chapters. We will also give a brief overview of Kimera-Multi [7], the recently-developed multi-robot version of Kimera.

2.1 Loop Closure Detection

The core of this thesis lays on the experimental evaluation of learning-based methods for loop closure detection. We evaluate such methods against ORB, a hand-crafted, heuristic-based approach that is currently considered state-of-the-art. We review these two approaches in this section.

2.1.1 Classical Methods

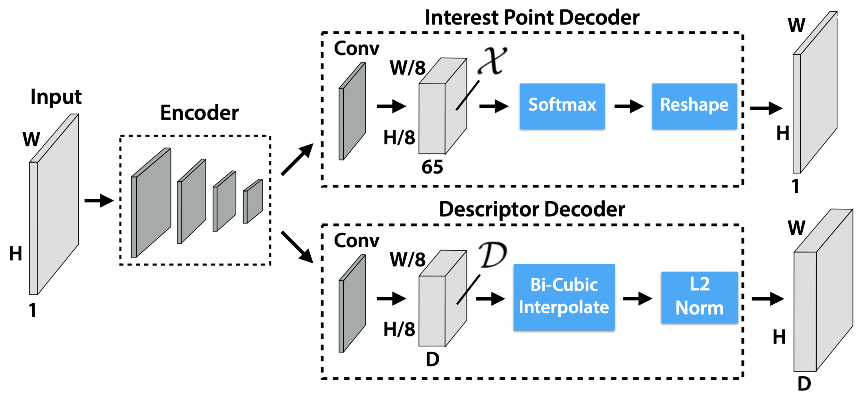
ORB was introduced in 2011 as an alternative to other commonly-used feature descriptors, such as SIFT [10] and SURF [11]. SIFT is considered very successful for feature detection with descriptors, and is still widely used in many computer vision problems. However, SIFT is costly in terms of computation and not an ideal solution for real-time applications like SLAM systems [5]. SURF is an alternative with lower computation cost. ORB, however, produces better results than SURF, and it performs as well as SIFT while being capable of running in real-time [5].

ORB uses FAST keypoint detector [12] and BRIEF for descriptors [13]. Hence the name ORB (Oriented FAST, Rotated BRIEF). It is worth noting that BRIEF lacks rotational invariance, but the addition of a nearest-neighbor technique and a learning step yields rotational invariance results for ORB [5]. ORB works by first detecting FAST keypoints in a image. The only input to FAST is a threshold between the central pixel and those in a ring around it [12]. ORB orders these keypoints by using Harris corner measures [14]. These two ideas work together as follows: ORB takes as input a target of N keypoints for a given image. The FAST threshold is then set to be low enough such that FAST produces more than N keypoints. Then ORB uses the Harris corner measures to select the top N points that ORB will return as the N desired keypoints [5]. As for the descriptors, ORB builds an additional step on BRIEF [13] in which ORB discretizes rotation angles every 12 degrees and uses a lookup table with BRIEF patterns that were previously computed. This table allows ORB to create a consistent descriptor for a given keypoint, invariant of rotation, as long as the computed patch orientation is consistent from different angles looking at the point [5].

2.1.2 SuperPoint

SuperPoint is a pretrained convolutional neural network for keypoint detection and their corresponding descriptors [1]. The model takes a full 2D image frame as an input and produces a set of keypoints and descriptors. As shown in Figure 2-1, it does this by compressing and encoding the image, and running the encoded version on two separate neural networks. One uses the pretrained weights to identify keypoints on the image, while ignoring extreme illumination limitations that other techniques usually have. The second neural network is responsible for generating vectors of descriptors

Figure 2-1: SuperPoint architecture (figure courtesy of [1])



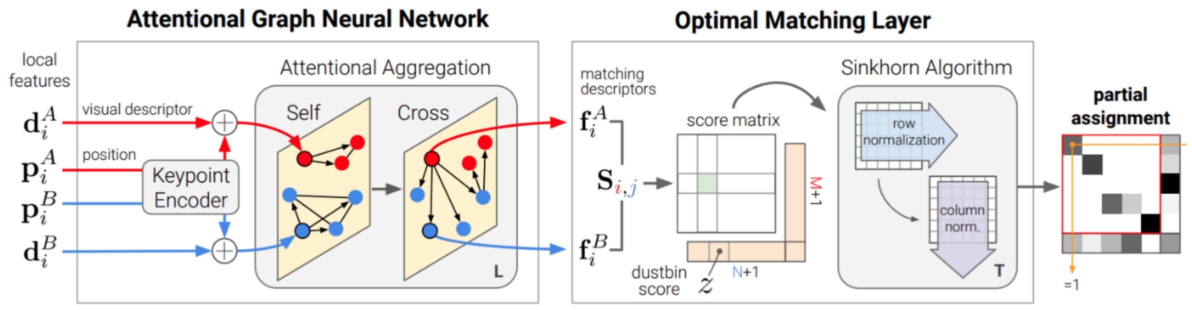
for interest points. SuperPoint claims to perform better that classical methods like SIFT or ORB, identifying more keypoints and providing richer descriptors.

2.1.3 SuperGlue

SuperGlue is a deep neural network that performs feature matching [2]. The classical pipeline used for place recognition, given two images as inputs, includes an initial detection of keypoints and descriptors, the feature-matching step, and an outlier filter followed by the pose estimation. The extraction of keypoints is often performed with a classical method such as SIFT or ORB. Then, the feature matching step is performed using a nearest neighbor approach between the descriptors. Finally, the filtering step takes into account heuristics such as a ratio test and mutual check. The pose is then estimated using a geometric verification method such as RANSAC [15]. In contrast, SuperGlue combines the description, feature matching, and outlier filtering. It does so by performing a context aggregation technique, followed by an optimization layer that produces a partial assignment. SuperGlue claims to obtain best results when used with SuperPoint.

Figure 2-2 shows the two main components of SuperGlue. It takes local keypoints between two images, as well as their corresponding descriptors, and performs attention aggregation on its pretrained graph neural network. This step allows SuperGlue to

Figure 2-2: SuperGlue Architecture (figure courtesy of [2])



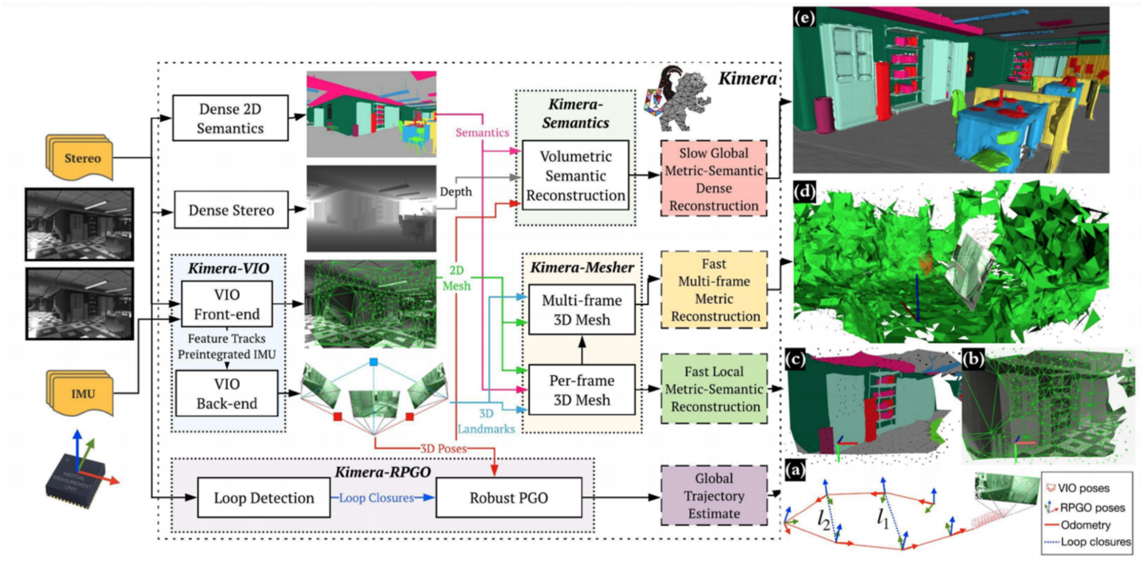
compare keypoints and their descriptors within an image (self-attention aggregation), which allows the neural net to take into consideration similarities and locality of the image. Then, it does a similar comparison between the two images (cross-attention aggregation) while keeping the knowledge of the self-attention step. The output of this first component is a score matrix of matching, which gets passed to an optimization layer. There, SuperGlue uses optimal transport and an outlier filtering algorithm that takes into consideration possible occlusion and visibility in order to produce a partial assignment that only matches each keypoint to at most one keypoint on the other image.

2.2 SLAM Systems

Advances in robotics and computer vision have made it possible to develop robust SLAM systems [16, 17]. Similarly, there have been drastic progress in methods to generate semantic labeling of 3D reconstructions of entire environments [18, 19, 20, 21, 22, 23, 24]. These two subfields, however, have seen much of their respective improvements develop in isolation. Kimera is one of the first open-source libraries to offer real-time Metric-Semantic SLAM on CPU [3]. This combination allows for reconstruction of the geometry of the environment, simultaneous localization, and the construction of semantically annotated meshes that accurately label objects for easy human understanding of the environment. We use the SLAM modules in Kimera as the baseline system for this thesis work, and conduct experiments testing the integration of new algorithms to Kimera.

2.2.1 Kimera

Figure 2-3: Kimera Modules and Architecture (figure courtesy of [3])



As shown in Figure 2-3, there are 4 modules that are part of Kimera: visual-inertial odometry (VIO) for a fast and accurate state estimation; a robust pose graph optimizer (RPGO) for global trajectory estimation; a lightweight 3D mesher for fast mesh reconstruction; and a dense 3D metric-semantic mapper for a global metric-semantic mesh reconstruction. This modularity makes it possible to run Kimera as a state-of-the-art VIO or a full SLAM system with or without semantic mesh reconstruction. Taking a closer look at the role of loop closure detection in Kimera-RPGO loop closure detection, we can better understand how place recognition works and how it affects the overall system. Kimera-RPGO detects loop closures by comparing the current frame against past keyframes for which we stored keypoints and descriptors using ORB. Then, it computes a pose estimation (possibly up to scale) between the two images. Finally it executes pose graph optimization to obtain an optimal trajectory estimate. In order to detect loop closures, Kimera relies on Bag of Words (DBoW2) [6]. DBoW2 can quickly extract candidate keyframes for loop closures.

Then, the feature matching algorithm is performed, and a geometric verification is used to reject outliers. In Chapter 3, we will give a more detailed overview of Kimera-VIO. Then, in Chapter 4, we will go over Kimera-RPGO, as well as how the loop closure components integrate into the system.

2.2.2 Kimera-Multi

Recent work expanding the capabilities of Kimera to a multi-agent system led to Kimera-Multi, the first fully distributed system for multi-robot metric-semantic SLAM [7]. Kimera-Multi combines the previous Kimera modules with a multi-robot, distributed architecture for simultaneous reconstructions of an environment from multiple vehicles. Kimera-Multi depends on the loop closures obtained across the system, and more specifically, across different robots in the environment. To achieve this, robots exchange their DBoW2 vectors in order to find loop closure candidates across all vectors from all robots, and in doing so, Kimera-Multi can accept loop closures between different robots. Those are then sent to a distributed pose graph solver, which also takes the trajectories and optimizes the global trajectories of the robots on the global map. This pipeline can then inform the robots of the resulting trajectory estimate so they can perform local mesh optimizations, which deforms the local meshes in order to achieve global consistency. Because Kimera-Multi global consistency relies on loop closure detection, improvements to Kimera loop closure detection algorithms would benefit not only the single-robot system, but the entire multi-robot stack.

2.3 Outlier Rejection

A full SLAM system with a loop closure module requires the capability of optimizing the trajectory given the detected loop closures, while also rejecting incorrect and inconsistent loops. To that end, the final section of this chapter is dedicated to outlier rejection, with a particular emphasis on the methods used by Kimera. The pose graph optimization problem has also seen a number of approaches and advances to achieve the robustness and accuracy we have today. Early methods were based on a standard least square formulation, RANSAC [15], branch and bound [25], and M-estimation [26]. Since then, many methods have been developed to improve the capabilities of outlier rejection and improve on the PGO problem [27, 28, 28, 29, 30, 31, 32, 33, 34, 35].

Kimera-RPGO is implemented with significant use of GTSAM [36]. On the first release of Kimera, RPGO relied on the Incremental Consistent Measurement Set Maximization approach (PCM) [37] for outlier rejection. However, with the recent development of Graduated Non-Convexity (GNC) for robust spatial perception [8], our team also integrated GNC into Kimera and already saw improvements in performance. For the work presented in this thesis, we decided to show results using GNC as outlier rejection method.

Chapter 3

Loop Closure Improvements

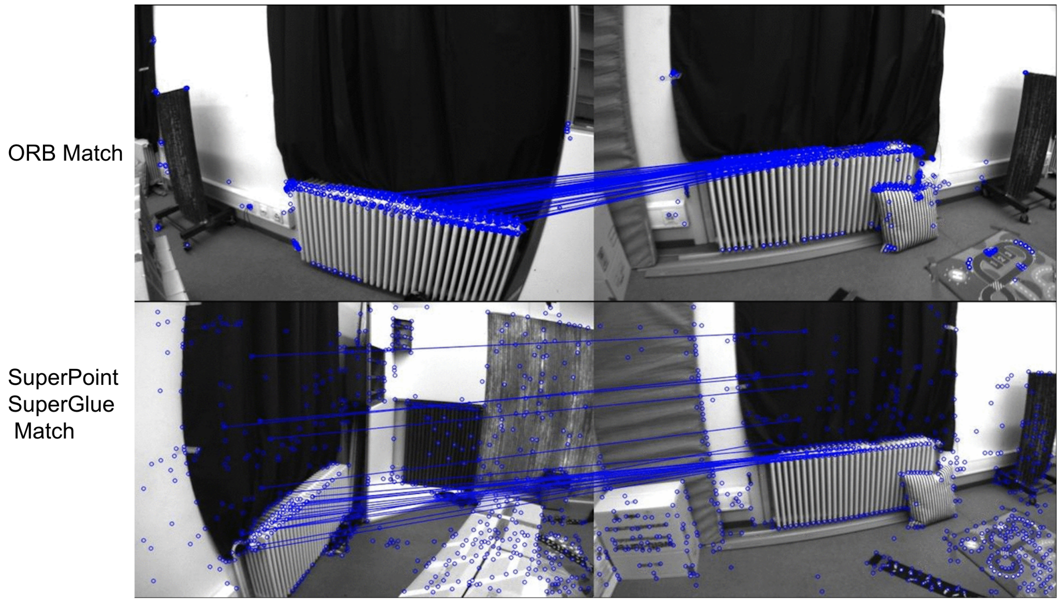
3.1 Introduction

In this chapter, we present our initial results from the experimental work done for this thesis. Our work centers around one initial question about our loop closure techniques: Can learning-based methods improve loop closure detection? We begin by comparing ORB [5] against SuperPoint [1] and SuperGlue [2] reviewing results in the literature, and presenting our own experimental results. We describe our integration with Kimera [3], the system used for our experimental evaluations. Finally, we show extensive results from our experiments on benchmarking and simulated datasets in which we see promising improvements for loop closures using the learning-based approaches. For this chapter, we only focus on the quality of the relative pose of each loop closure in isolation.

3.2 Background

Loop closures are essential to reduce accumulated drift on SLAM systems. Kimera is a real-time, modular SLAM system [3], and as such, we desire to have real-time feature extraction, place recognition, and feature matching. These techniques, as already stressed, are needed for an efficient loop closure module. The Kimera Robust Post Graph Optimization module currently relies on ORB, a state-of-the-art feature

Figure 3-1: ORB and SuperPoint/SuperGlue visual comparison



extraction and feature matching technique that has proven to be rotation invariant and resistant to noise. ORB is two orders of magnitude faster than other feature matching techniques like SIFT, providing a real-time solution for a SLAM system [5].

SuperPoint and SuperGlue are recently-developed, learning-based approaches for feature extraction and feature matching, respectively. SuperPoint offers a pretrained neural network that overcomes some of the difficult challenges of feature extraction, like extreme illumination and darkness [1]. SuperGlue is trained on extensive real-world situations, which allows the neural network to correctly match complicated geometric transformations and, in some situations, overcome occlusion [2].

These promising aspects from the learning-based approaches incentivize us to compare them against the classical method we currently use for Kimera. Figure 4-1 shows a visual comparison of a feature match between ORB and SuperPoint/SuperGlue. These images were taken from Vicon Room 2, one of the sequences in EuRoc [38] – our benchmarking dataset. SuperPoint/SuperGlue produce correct matches with wider baselines, as the one shown in 4-1. ORB cannot replicate this match. On this rosbag from the dataset, SuperPoint/SuperGlue detects approximately three times as many feature matches. We will provide a more extensive, quantitative comparison in our evaluation section.

3.3 System Overview and Experimental Setup

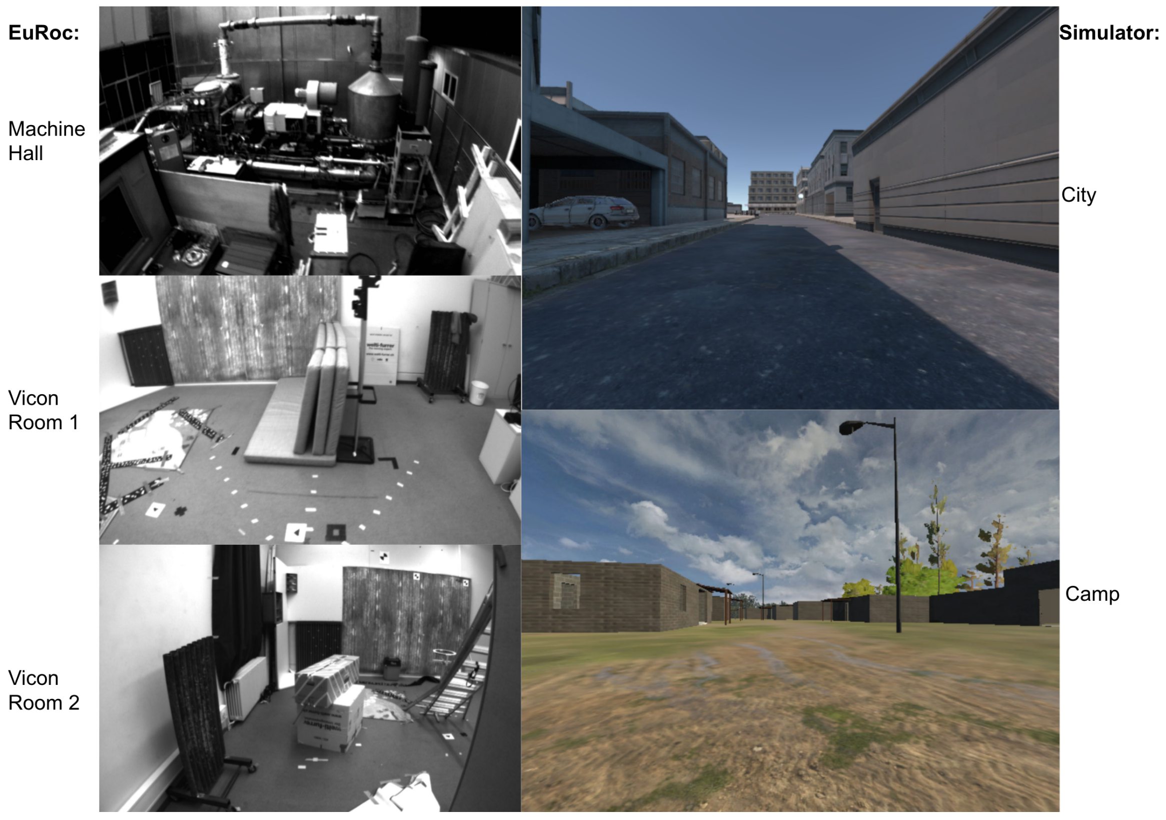
As previously stated, Kimera [3] is the base system for the work done for this thesis. Although Kimera has four different modules, our work focuses on only two: Kimera Visual-Inertial Odometry (Kimera-VIO) and Kimera Robust Post Graph Optimization (Kimera-RPGO). Kimera-VIO uses monocular and stereo keyframes, and the maximuma-posteriori visual-inertial estimator introduced in [39]. The front-end of Kimera-VIO processes the raw sensor data from the cameras and IMU. The back-end fuses the results to produce the estimate output. The loop closure detection is part of Kimera-RPGO, which compares the current keyframe with past keyframes. This module also computes global consistent keyframe poses. For the work presented in this thesis, we will mostly focus on the loop closure detection of Kimera-RPGO. In the next chapter, we go into further detail about outlier rejection.

For this chapter, we will focus on how ORB and SuperPoint/SuperGlue detect loop closures in Kimera. We isolate metrics of the loop closures by themselves to evaluate the improvements and errors that the learning-based approach would introduce. The four main components of our loop closure pipeline are feature extraction, place recognition, feature matching, and a geometrical verification. For feature extraction, we use either ORB or SuperPoint to extract features from each keyframe. These features are used for visual feature tracking in Kimera-VIO and for place recognition. We rely on bag-of-word representations of the keypoints and use the DBoW2 library [6] to quickly compare our current keyframe with past keyframes and find potential matches. Then we perform feature matching using either ORB or SuperPoint – this produces a result similar to Figure 4-1. Finally, we enforce a geometrical verification using both a 5-point monocular RANSAC [40] and a 3-point stereo RANSAC [41]. Only loop closures that pass the geometric verification are then sent to Kimera-

RPGO. For the evaluation in this section, we will look at all of these loop closures.

We use multiple datasets for our evaluation. The EuRoc dataset [38] contains 11 different sets of data from 3 different rooms: Machine Hall; Vicon Room 1; Vicon Room 2. We used these benchmarking datasets in addition to our own simulated dataset, taken from a photo-realistic Unity simulator. We include data from two different simulated environments: City and Camp. With the exception of Machine Hall, which contains 5 different sequences, every other location in this dataset has 3 different sequences. That is, every location contains 3 different rosbags we run Kimera on, as well as ground truth trajectories. Figure 3-2 show images of the difference between the scenarios.

Figure 3-2: Snapshots of the different environments



3.4 Evaluation

For our evaluation of loop closure improvements, we collected the rotation and translation of loop closures that passed the geometric verification, using either ORB or SuperPoint/SuperGlue. We plotted the histograms of the magnitude of the rotations and translations across loop closures. Larger rotations and translations indicate the capability of performing loop closures across distant (wide baseline) images. Furthermore, we computed the rotational and translation error of each loop closure with respect to the ground truth, and plotted those on histograms as well. Each set of data was run with Kimera a total of 10 times. The Kimera parameter used for all the experiments can be found at the appendix. The immediate conclusion from these results is that the learning-based approach produces significantly many more wide-baseline loop closures, but the distribution of the errors indicates the presence of more outliers. As for computational time, both approaches run on real-time but SuperGlue requires GPU.

Machine Hall

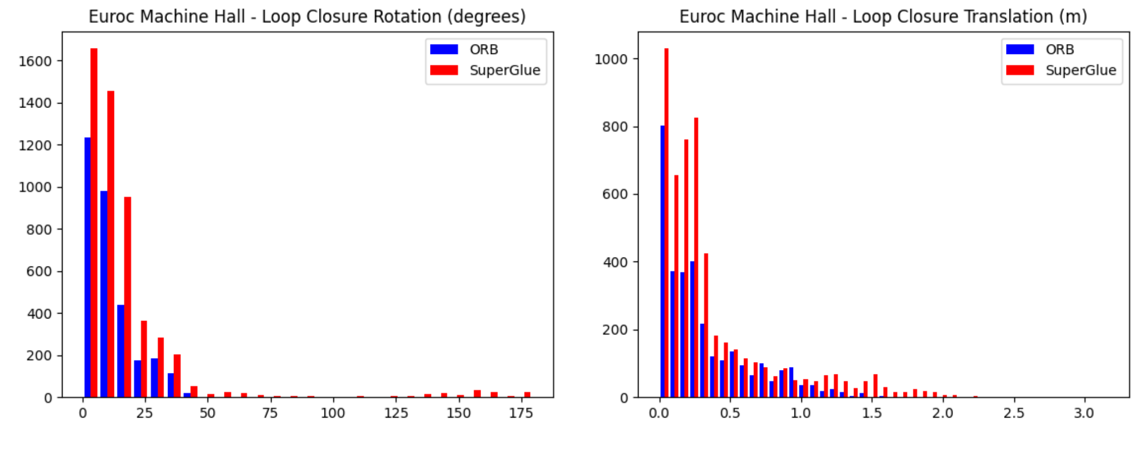


Figure 3-3: Machine Hall - Rotation and translation between loop closure poses

Machine Hall shows a small increase of loops detected by SuperGlue, as shown in Figure 3-3. As for the error when comparing against the ground truth, Figure 3-

4 shows that, while most loops from both methods have very small error overall, SuperGlue does have a number of loops with larger errors. This gives us our first indication that SuperGlue produces more loop closures, with the potential of many more outliers.

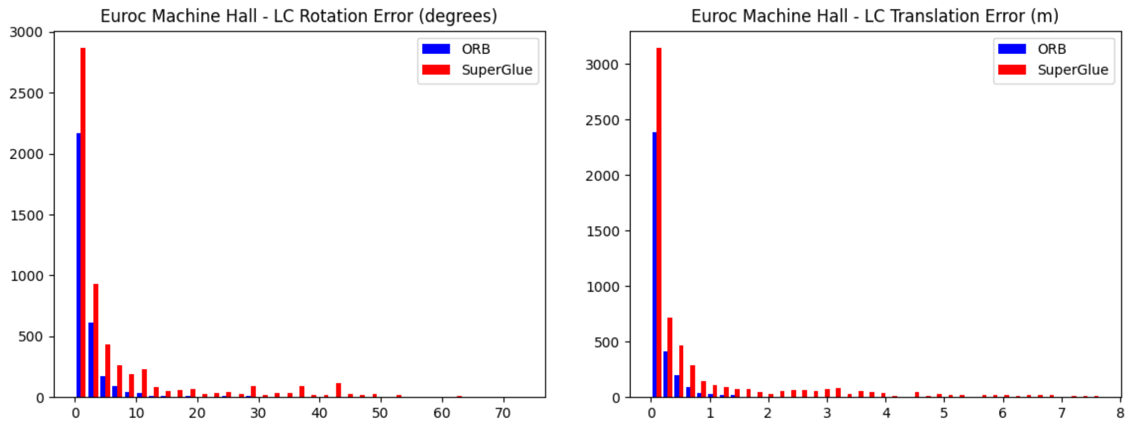


Figure 3-4: Machine Hall - loop closure error against the ground truth

Vicon Room 1

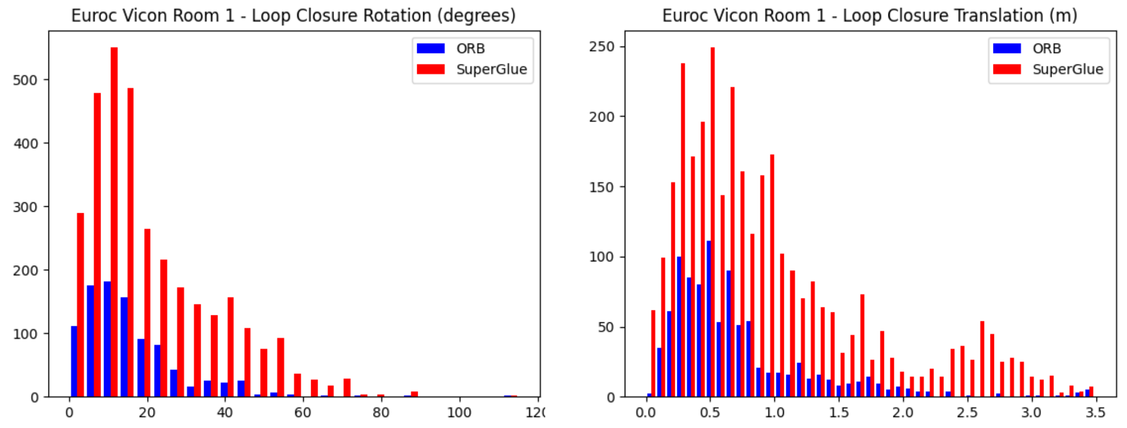


Figure 3-5: Vicon Room 1 - Rotation and translation between loop closure poses

In contrast with Machine Hall, Vicon Room 1 offers a new environment in which the distinction between ORB and SuperGlue is clear. Figure 3-5 shows a significant increase in loop closures detected by SuperGlue. In Figure 3-6 we see very small errors from the loop closures when comparing against the ground truth. These results indicate the presence of many more loop closures with little error. In this dataset, learning-based approaches lead to improved quality for the loop closures.

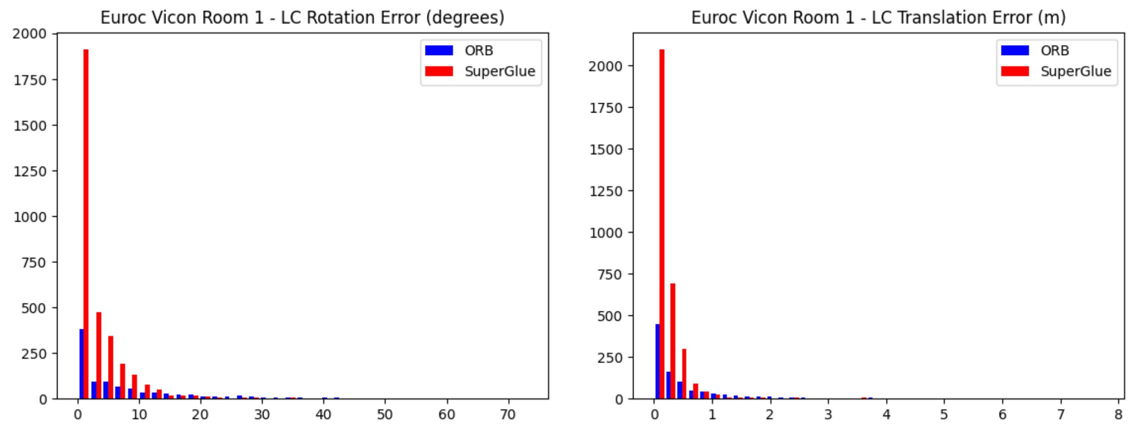


Figure 3-6: Vicon Room 1 - loop closure error against the ground truth

Vicon Room 2

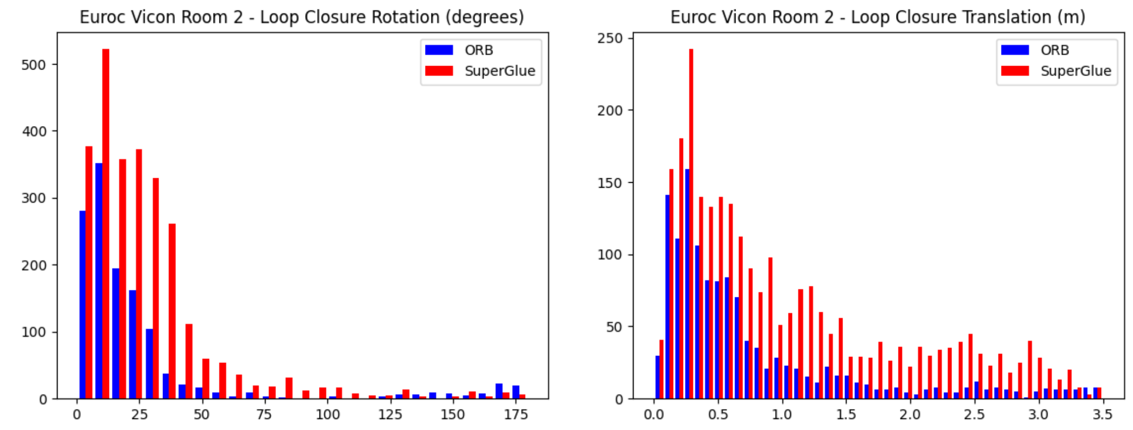


Figure 3-7: Vicon Room 2 - Rotation and translation between loop closure poses

Vicon Room 2 preserves the promising results we saw on Vicon Room 1. Just like Figure 3-5, Figure 3-7 also shows the presence of many more loop closures detected by SuperGlue. In contrast with 3-6, 3-8 shows slightly greater error on our loop closures, but it is worth noticing that ORB also have a similar distribution on the error. Thus, we are still seeing many additional loop closures, with large rotation and translation, and still showing similar error as ORB.

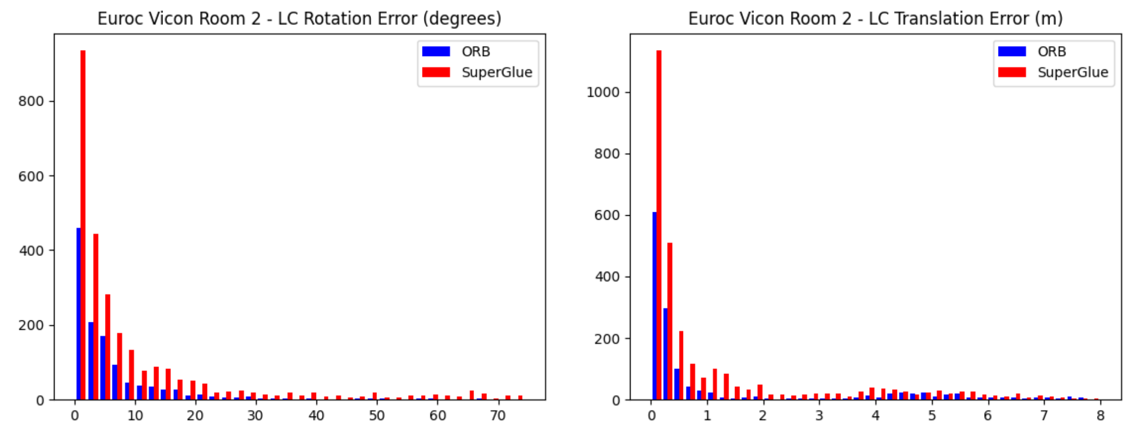


Figure 3-8: Vicon Room 2 - loop closure error against the ground truth

City Simulator

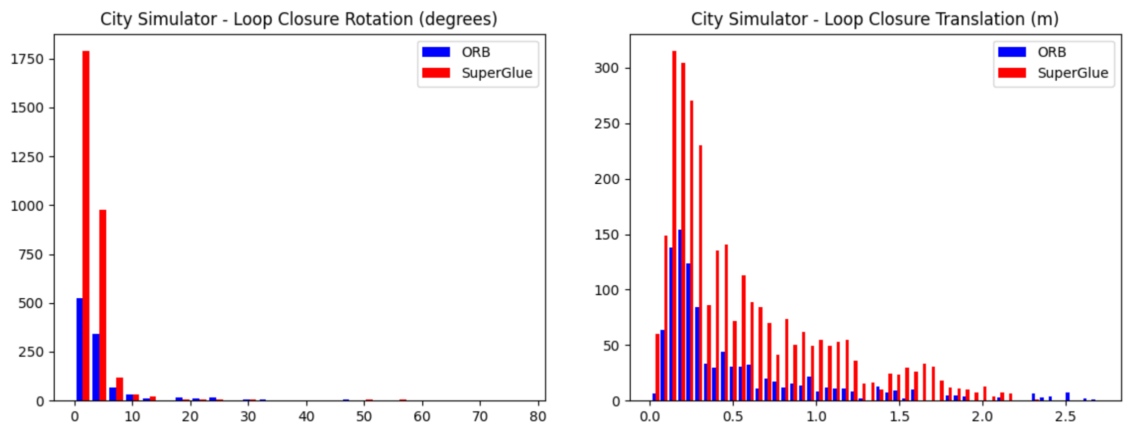


Figure 3-9: City - Rotation and translation between loop closure poses

The simulator is a completely different type of dataset, that pictures large outdoor environments. One major difference we can see from the plots immediately is that all loops detected here have very small rotation angles. We still see many more loop closures from SuperGlue when looking at the translation, with the errors being small.

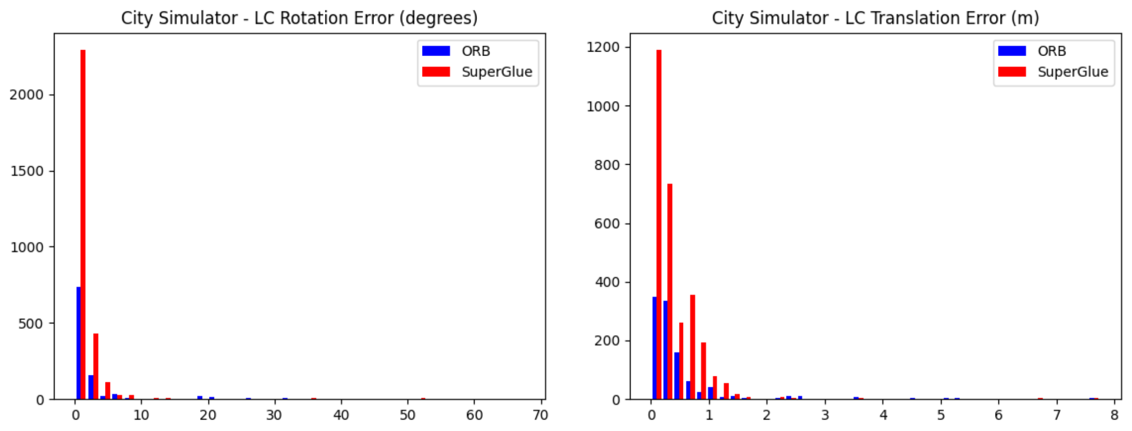


Figure 3-10: City - loop closure error against the ground truth

Camp Simulator

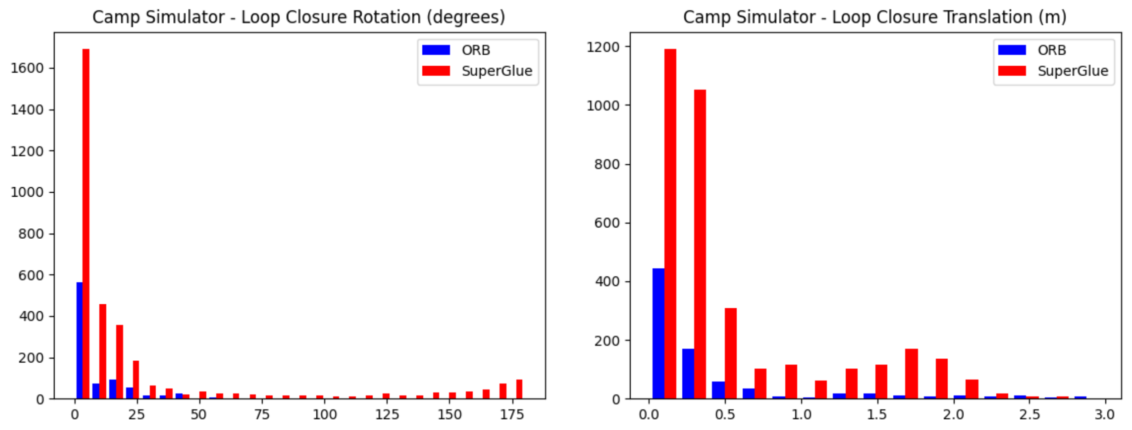
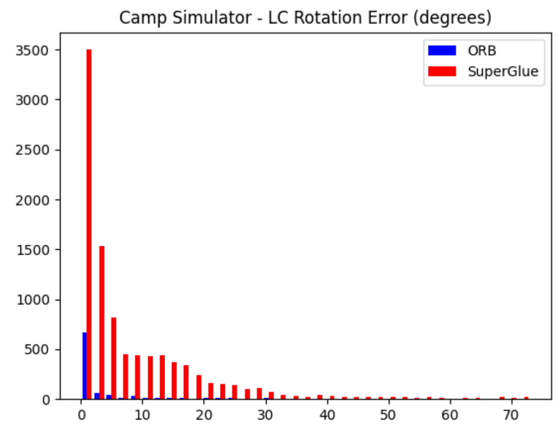


Figure 3-11: Camp - Rotation and translation between loop closure poses

Camp shows us yet another very different set of results. Notice on Figure 3-11 that the angle of rotation of the loops with SuperGlue are extremely wide, potentially suggesting incorrect loops. The errors in this particular location are also clearly, much higher for SuperGlue alone, from 3-12



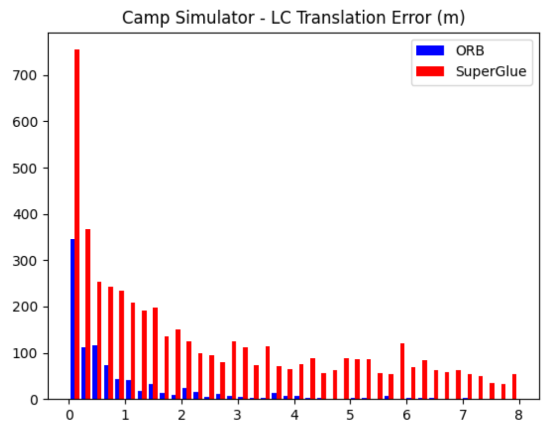


Figure 3-12: Camp - loop closure error against the ground truth

3.5 Closing Remarks

There are three observations we can see from the results on these dataset:

1. Learning-based feature extraction and feature matching always produce more loop closures with similar or greater rotation and translation baselines.

2. The error distribution of the loop closures detected using SuperGlue is similar to that of ORB. While some cases did show slightly greater errors for SuperGlue, most of the errors clusters around 0.

3. The performance of SuperGlue seems to be dataset-dependent, with more promising results in Vicon Rooms and City, and slightly worse performance in Machine Hall and Camp.

From these results, we conclude that the learning-based approach is indeed capable of producing many more loop closures. However, these results do not tell how these loop closures affect a SLAM system, as we are only looking at loop closures in isolation, each loop individually. In Chapter 4, we will look at the impact of the loop closures on the trajectory estimate of a SLAM system. We will also look at the loop closure again after implementing our outlier rejection step, thus removing all loops that we ultimately do not use for the final, optimized output from our system.

In fairness to ORB, we must acknowledge that more loop closures could be possible to obtain by relaxing ORB parameters. Although we do not include histograms showing these results, we did run experiments with such settings. We ultimately decided against using those as the absolute trajectory errors increased dramatically for ORB. We will discuss the absolute trajectory errors in the next chapter.

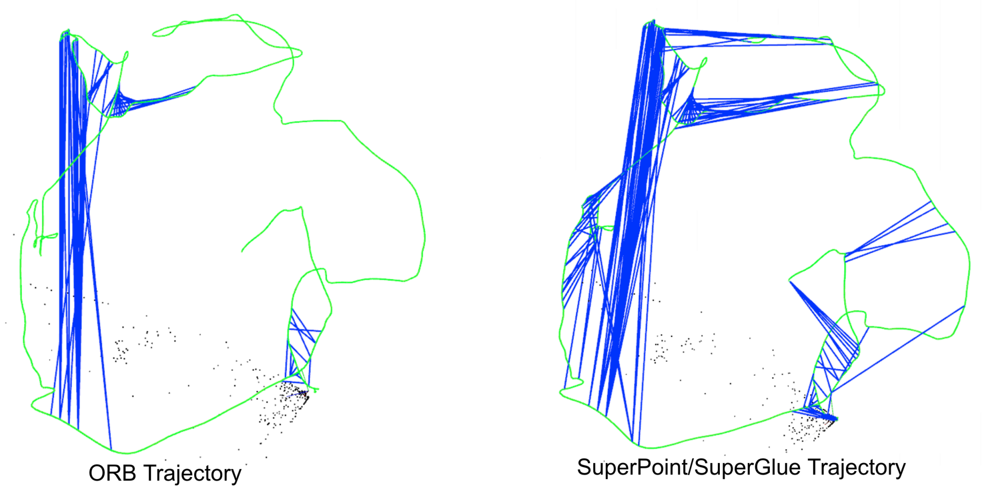
Chapter 4

SLAM Evaluation

4.1 Introduction

To give a better sense of what a trajectory looks like and how loop closures can connect different parts of the trajectory and reduce error accumulation, Figure 4-1 provides a visualization using both loop closure detection methods. In this figure, the green line represents the optimized trajectory output from Kimera. The blue lines are loop closures detected on this trajectory. As we will discuss in this chapter, many of those loop closures are ultimately rejected by Kimera-RPGO. The accepted loop closures are used to make correction on the trajectory and reduce the accumulated drift. This visualization reaffirms our conclusion from the previous chapter: that SuperGlue is able to detect many more loop closures with significantly greater angles between the two poses. In the previous chapter, we talked about how loop closures detected using SuperPoint and SuperGlue compare to ORB. That chapter isolates the metrics discussed therein: loop closure rotations, translations, and errors. In this chapter, we look beyond the isolated loop closure detection. We discuss how Kimera-RPGO rejects outlier, and plot the same metrics from Chapter 3 after we remove outlier. Furthermore, we use the ground truth trajectories to obtain the absolute trajectory error of our optimized, estimated trajectories using Kimera with both loop closure detection methods. This gives us a full overview of how the methods perform on the entire pipeline of a SLAM system.

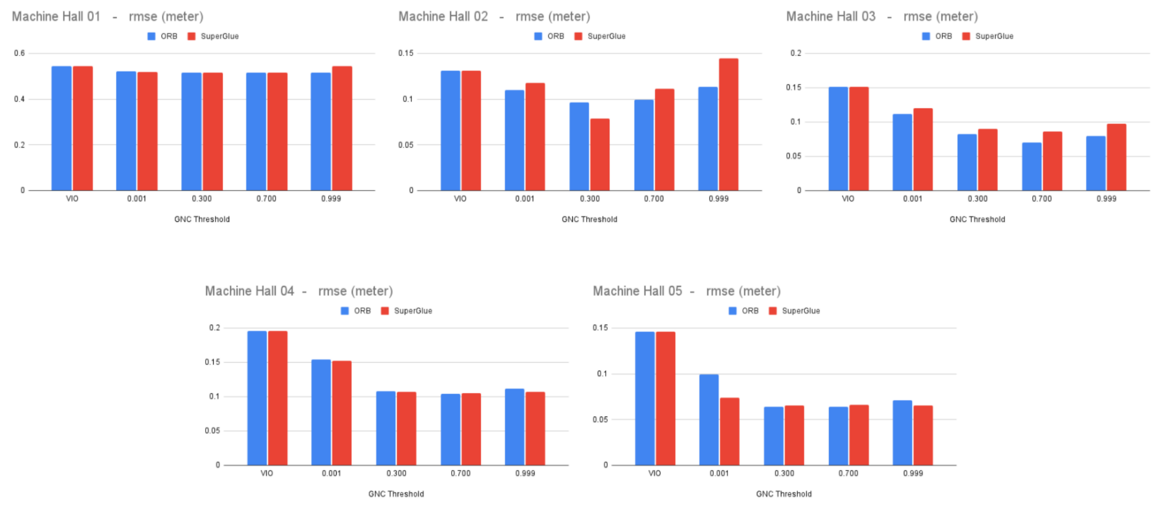
Figure 4-1: ORB and SuperPoint/SuperGlue trajectory comparison



4.2 Outlier Rejection in Pose Graph Optimization

The previous chapter already explained the system used for our experimental evaluations. The main addition we will discuss in this chapter is outlier rejection. Kimera-RPGO is in charge of rejecting outliers from our detected loop closures, as well as keeping global consistency for all poses on the graph. The module is implemented with significant use of GTSAM [36]. On the first introduction on Kimera, RPGO relied on the Incremental Consistent Measurement Set Maximization (PCM) [37] for outlier rejection. However, with the recent development of the Graduated Non-Convexity (GNC) for robust spatial perception [8], our team also integrated GNC into Kimera and saw performance improvements. In the integration with Kimera, GNC takes a probability threshold for filtering outliers and a covariance for each residual. In the rest of this sections, we will look at the results from an evaluation of different probabilistic threshold on the EuRoc dataset [38]. We use a fixed isotropic covariance with a standard deviation of roughly 0.1 degrees for rotation and 0.01 meters for translation. The exact parameters use can be found at the appendix, with the GNC parameters being at the end of table A.3. To evaluate the performance of the system, we will rely on the absolute trajectory error (ATE) measured from the RMSE of 10 iterations for every trajectory. The ATE is the most popular accuracy metric

Figure 4-2: GNC Probability threshold on Machine Hall



for SLAM. It measures the average deviation from ground truth trajectory per frame [42]. A small ATE would indicate a better performance.

From these evaluations, we can see that accepting more loop closures reduces the ATE. However, if the probabilistic threshold for GNC becomes too large, the pipeline starts accepting a number of incorrect loop closures as inliers, causing the ATE to increase again. We can see from the plots on figures 4-2, 4-3, and 4-4 that this point is actually on the high end, so we can use a conservative threshold of around 0.70 for the rest of the experiments presented in this thesis.

Figure 4-3: GNC Probability threshold on Vicon Room 1

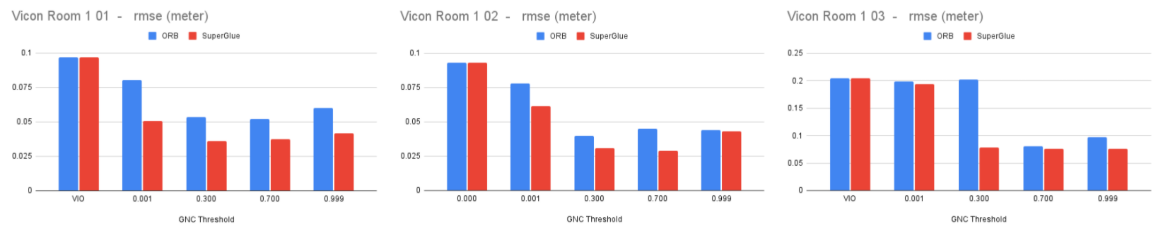
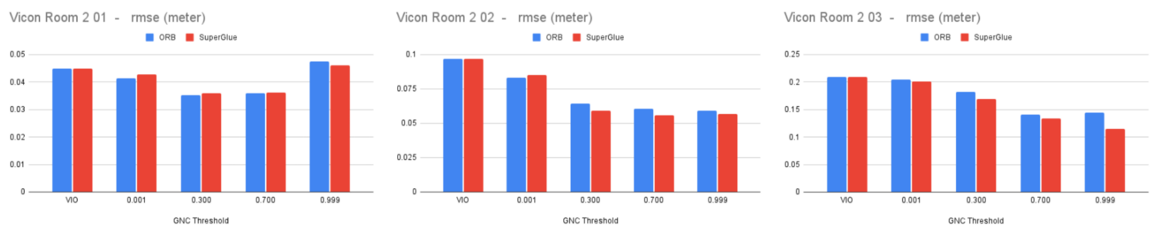


Figure 4-4: GNC Probability threshold on Vicon Room 2



4.3 Loop Closures Post Outlier Rejection

We will revisit the individual loop closure metrics from the previous chapter, but after filtering using GNC. For this section, we will also define two types of loop closures. Those are:

1. **Diverse Loop:** A loop closure is diverse (i.e., it has a wide baseline) if its rotation angle is greater than 20 degrees and its translation is greater than 0.5 meters. Diverse loops are desirable since they indicate that the loop closure detection module is able to match images taken at very different viewpoints.

2. **Small Error Loop:** A loop closure has small error if its rotation error is less than 2 degrees and its translation error is less than 0.05 meters.

Given these definitions, we will now analyze the number of diverse and small error loops we see in these datasets before and after outlier rejection.

	ORB	SuperGlue
Machine Hall	168	**282**
Vicon Room 1	235	**1176**
Vicon Room 2	411	**1302**
City	**96**	24
Camp	150	**288**

Table 4.1: Diverse Loops Detected

	ORB	SuperGlue
Machine Hall	89	**196**
Vicon Room 1	25	**683**
Vicon Room 2	60	**349**
City	**1**	0
Camp	0	**14**

Table 4.2: Diverse Loops Accepted

	ORB	SuperGlue
Machine Hall	1567	**1959**
Vicon Room 1	274	**1036**
Vicon Room 2	292	**592**
City	65	**446**
Camp	346	**358**

Table 4.3: Small Error Loops Detected

	ORB	SuperGlue
Machine Hall	1565	**1938**
Vicon Room 1	271	**1035**
Vicon Room 2	289	**589**
City	35	**316**
Camp	346	**353**

Table 4.4: Small Error Loops Accepted

From these tables, there are two important conclusions we can make:

1. We do not reject almost any of the small error loop closures. This reinforces the robustness and accuracy of GNC as outlier rejection method for Kimera-RPGO.

2. There are more diverse loop closures that remain after our outlier rejection step when we use SuperGlue. This is the case for both absolute and relative numbers. Thus, we know that SuperGlue is providing loop closures with significantly larger rotation angles to our optimized trajectory.

In the remaining of this section, we will revisit the histograms from Chapter 3 after we filter out the rejected loop closures. This allows us to look only at the accepted loop closures by our SLAM system and make better conclusions about these two methods.

Machine Hall

After outlier rejection, Machine Hall shows a smaller increase of loops detected for SuperGlue, as shown in Figure 4-5. The differences between these plots are visually negligible. As for the error when comparing against the ground truth, Figure 4-6 show that SuperGlue now has a greater number of larger errors for some loops. This gives us a new indication that SuperGlue and ORB produces similar numbers of loop closures, with SuperGlue having a few additional loops but higher errors as well.

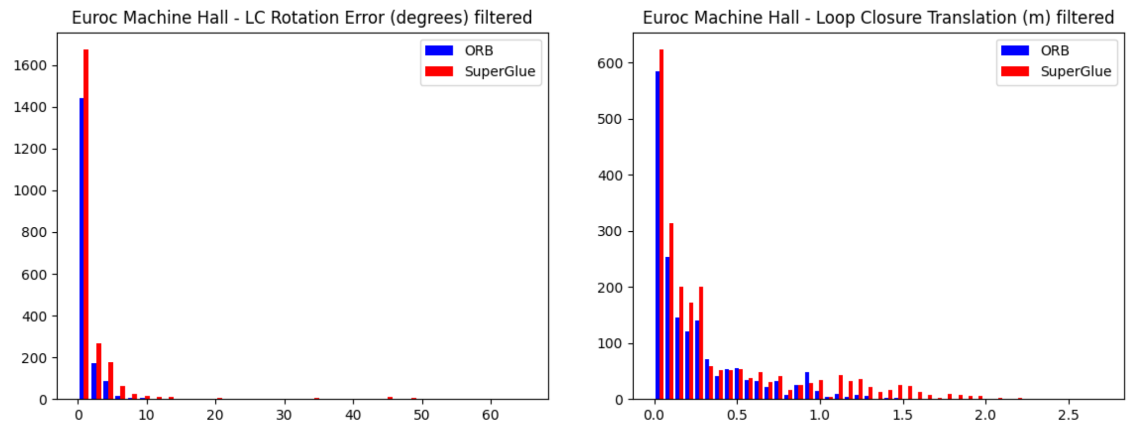


Figure 4-5: Machine Hall - Rotation and translation between loop closure poses restricted to the GNC inliers

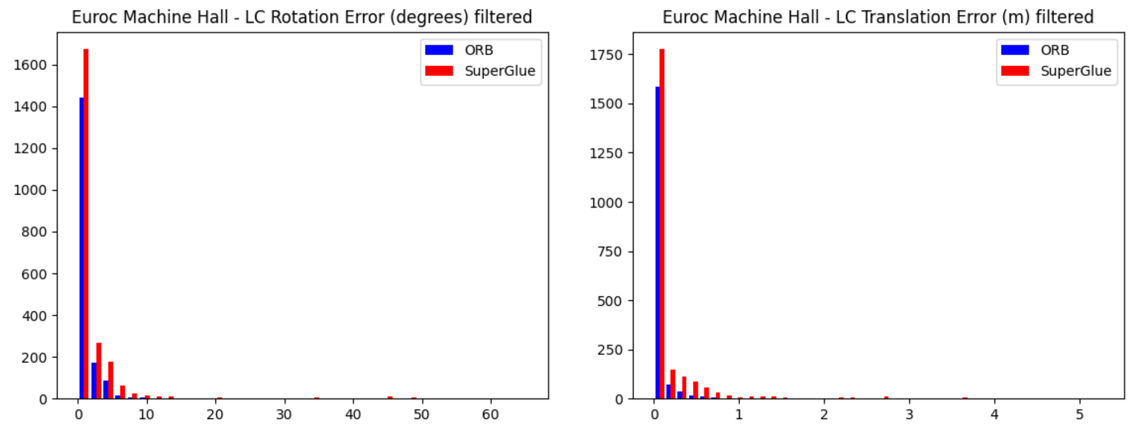
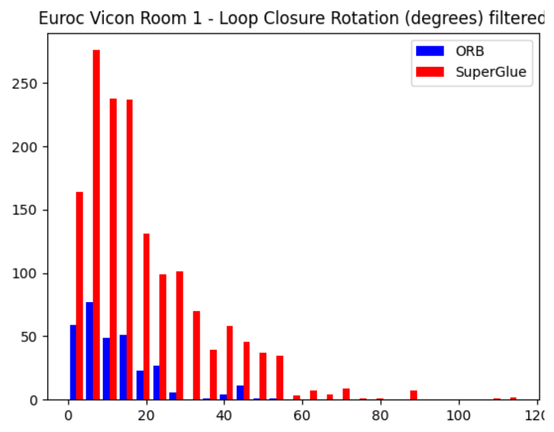


Figure 4-6: Machine Hall - Loop closure error against the ground truth restricted to the GNC inliers

Vicon Room 1



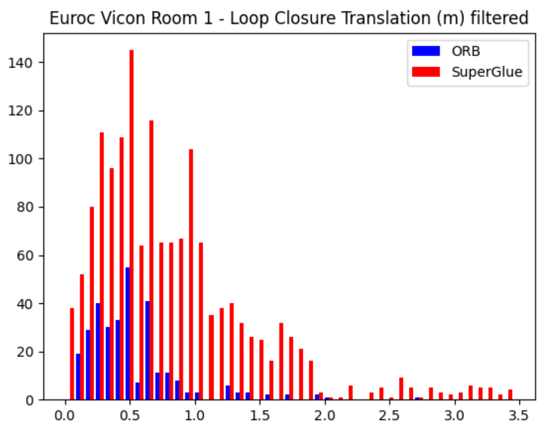


Figure 4-7: Vicon Room 1 - Rotation and translation between loop closure poses restricted to the GNC inliers

In contrast with Machine Hall, Vicon Room 1 still offers a different environment where the distinction between ORB and SuperGlue is easier to see. Figure 4-7 shows a very significant increase in loop closures detected by SuperGlue. In contrast to Chapter 3, Figure 4-8 shows that the error of the loops produced by SuperGlue is visually greater than those of ORB. These results indicate the presence of many more loop closures with larger error. Notice the presence of error of greater than 10 degrees post-filtering only for SuperGlue.

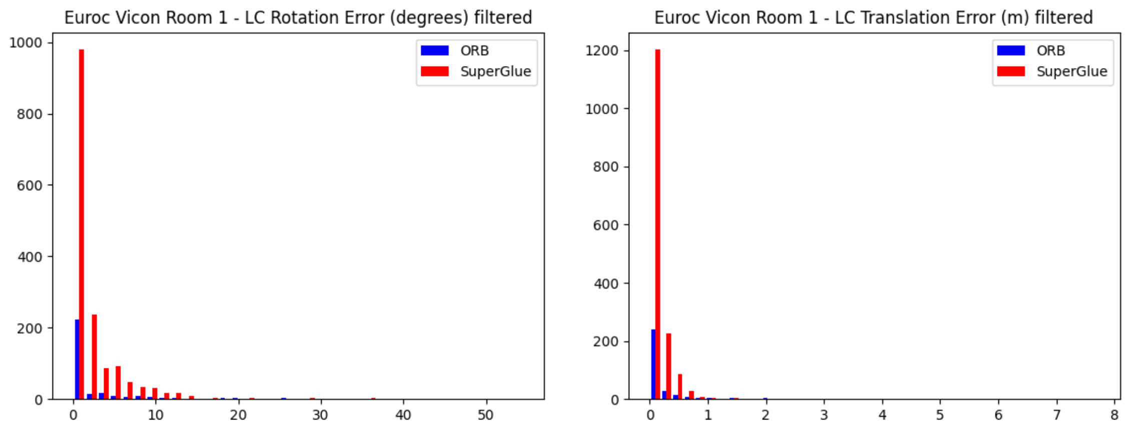


Figure 4-8: Vicon Room 1 - Loop closure error against the ground truth restricted to the GNC inliers

Vicon Room 2

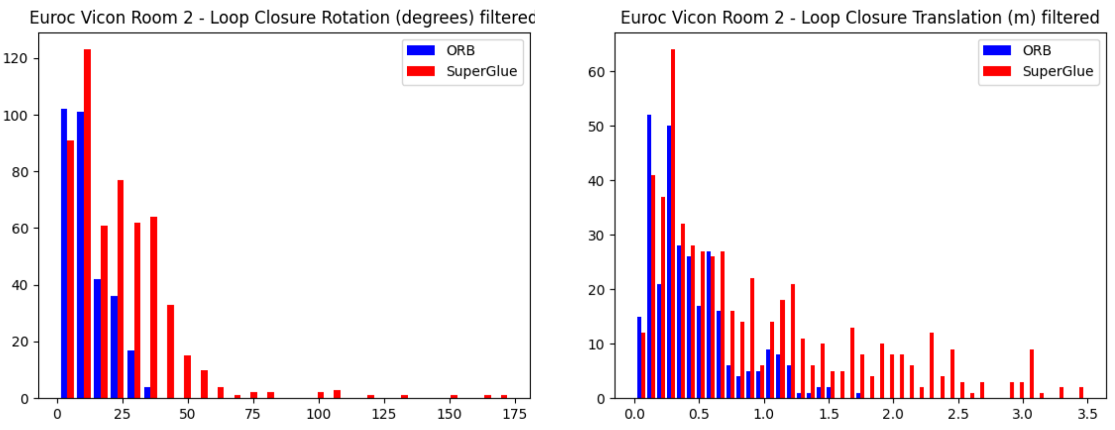


Figure 4-9: Vicon Room 2 - Rotation and translation between loop closure poses restricted to the GNC inliers

Vicon Room 2, once again, shows similar results as Vicon Room 1. Figure 4-9 shows the presence of more loop closures detected by SuperGlue, but not as many more as before outlier rejection. Figure 4-10 shows slightly greater error on our loop closures, with most of the large error produced by ORB removed post filtering. Thus, we still see additional loop closures from SuperGlue, but only SuperGlue loops still have greater error.

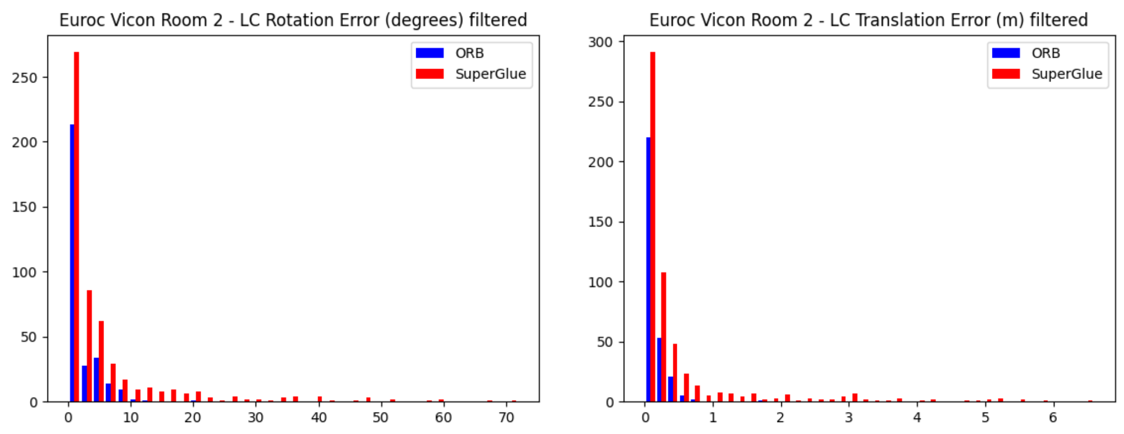


Figure 4-10: Vicon Room 2 - Loop closure error against the ground truth restricted to the GNC inliers

City Simulator

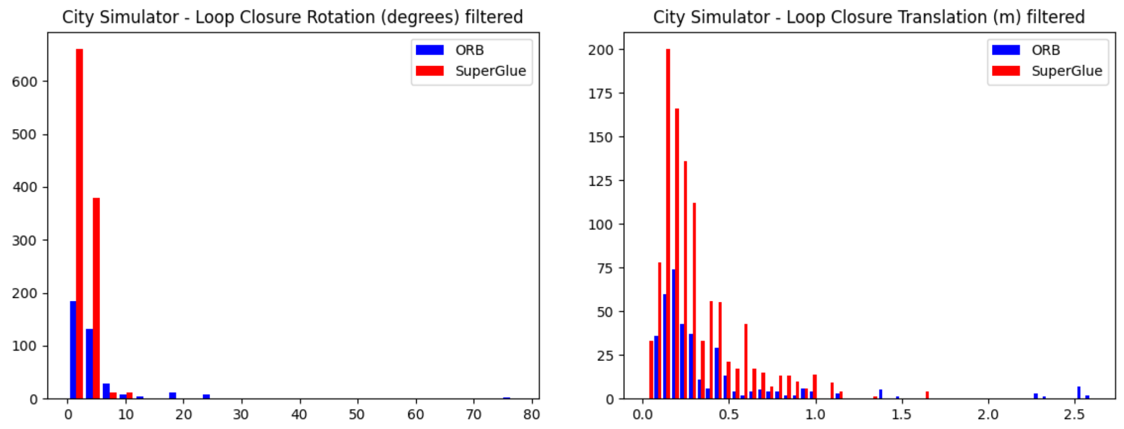


Figure 4-11: City - Rotation and translation between loop closure poses restricted to the GNC inliers

For City, 4-11 shows a drastic decrease of loops post filtering. The rotation and translation are small in general, as is the error in 4-12.

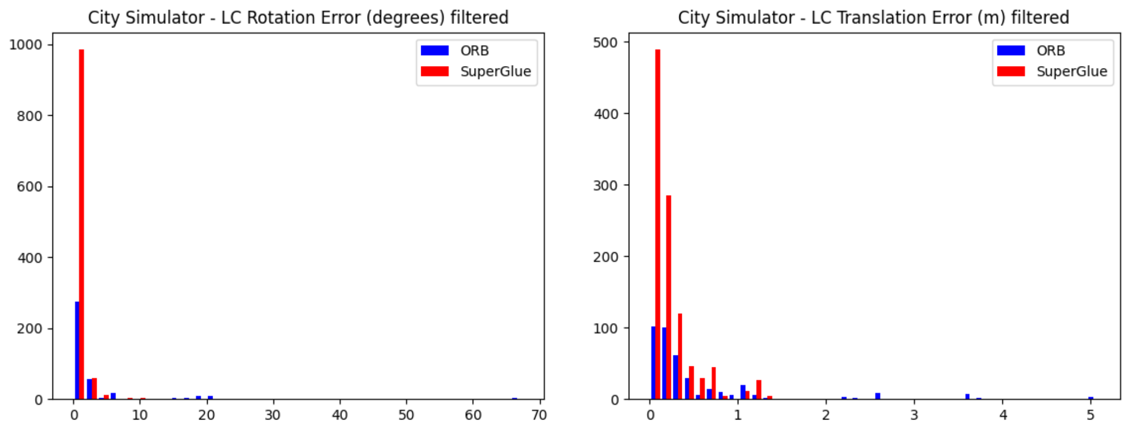


Figure 4-12: City - Loop closure error against the ground truth restricted to the GNC inliers

Camp Simulator

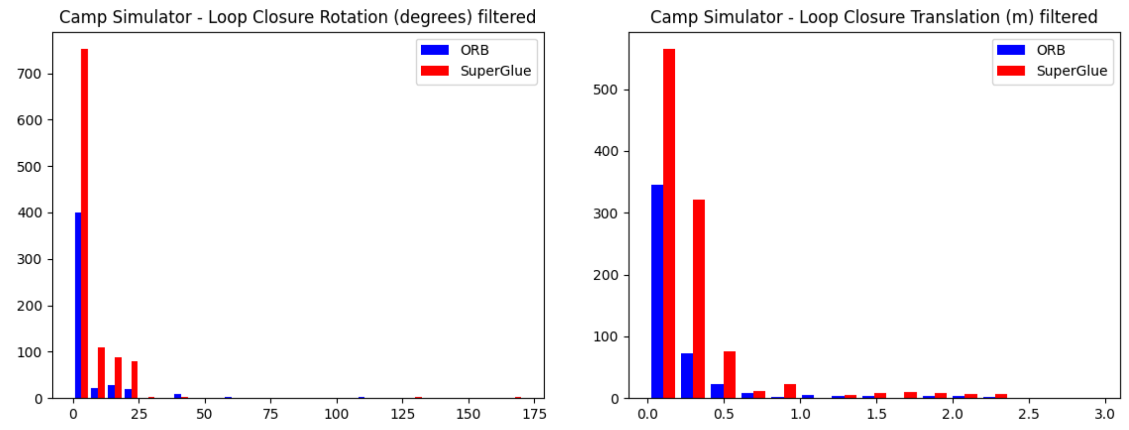


Figure 4-13: Camp - Rotation and translation between loop closure poses restricted to the GNC inliers

Similarly to City, Camp shows a drastic decrease of loops detected in 4-13, while SuperGlue clearly has some high error loops that survived the filtering step in 4-14.

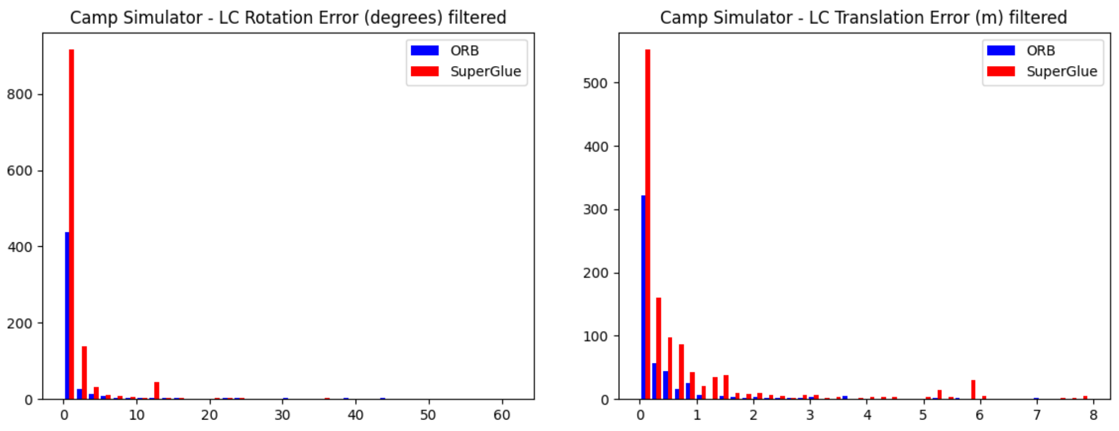


Figure 4-14: Camp - Loop closure error against the ground truth restricted to the GNC inliers

4.4 Evaluation on SLAM System

We introduce the ATE in section 4.2 as a baseline to evaluate the overall performance of Kimera-VIO and Kimera-RPGO. We used this metric in that section to evaluate the probability threshold we used for GNC in our experiments. In this section, we do a close inspection of the ATEs across every single rosbag in our datasets. The goal of this section is to provide a final, complete evaluation of the performance of ORB and SuperGlue in a full SLAM system. We also include the results from Kimera-VIO (without loop closures enabled) as a reference point.

Table 4.5 shows all the ATEs across our datasets. Besides the results from Camp (where the increase in error when loop closures are enable is likely from aliasing or from accepting an incorrect loop closure), the results from the other 4 environments are fairly consistent. Loop closures improve the performance of Kimera-VIO by reducing its absolute trajectory error. When we compare ORB against SuperGlue in this table, it is important to first notice that the differences between these results of the two methods are in the order of a few centimeters. In other words, the performance is very similar, with almost negligible difference in the improvement we see.

	VIO only	ORB	SuperGlue
Machine Hall 01	0.54	**0.51**	**0.51**
Machine Hall 02	0.13	**0.10**	0.11
Machine Hall 03	0.15	**0.06**	0.08
Machine Hall 04	0.19	**0.10**	**0.10**
Machine Hall 05	0.14	**0.06**	**0.06**
Vicon Room 1 01	0.09	0.05	**0.03**
Vicon Room 1 02	0.09	0.04	**0.03**
Vicon Room 1 03	0.20	0.15	**0.07**
Vicon Room 2 01	0.04	**0.03**	**0.03**
Vicon Room 2 02	0.09	**0.06**	**0.06**
Vicon Room 2 03	0.20	**0.13**	**0.13**
City 01	2.99	2.29	**2.19**
City 02	5.23	3.46	**3.29**
City 03	0.86	0.62	**0.60**
Camp 01	**0.32**	0.48	1.07
Camp 02	0.33	**0.18**	1.49
Camp 03	**0.42**	2.36	13.91

Table 4.5: Absolute Trajectory Error (ATE) - rmse (meter)

It is also important to notice that the performance of Kimera-VIO on its own is already pretty good with very low ATE across these datasets. City 02 actually shows the worst performance from VIO, and the one set in which the improvement from introducing loop closures is significant (about 2 meters). On the other hand, wrong loop closures on Camp 03 are very likely responsible for a significant increase in what was a low ATE from Kimera-VIO.

4.5 Closing Remarks

The promising loop closing improvements we first saw from isolating them in Chapter 3 seem to be reduced when we introduce the outlier rejection step. Looking at the number of diverse and small-error loop closures, it was clear the SuperGlue detects many more diverse loops that survive filtering, and that the small error loops do not get removed by robust PGO. When looking at ATE results, the conclusion is that SuperGlue, while performing better in a small number of runs, does not perform as well in many other runs. Overall, the difference in performance is negligible. One consistency with Chapter 3: the performance of SuperGlue seems to be dataset-dependent, with more promising results in Vicon Rooms and City, and slightly worse performance in Machine Hall and Camp.

Despite SuperGlue detecting more loop closures with wide baseline, it does not outperform ORB. One possible explanation could be that, because more of these diverse loop closures survived the filtering step, they introduce a greater risk of outliers with greater error. That is, consider the outliers with an incorrect wide baseline. SuperGlue will detect more of those as well. Thus, the risk of accepting an incorrect loop closure with wide baseline is higher. This incorrect loop closure would introduce greater error into the optimized trajectory. Future work can be done to analyze the large error loops that survive outlier rejection from both methods.

Chapter 5

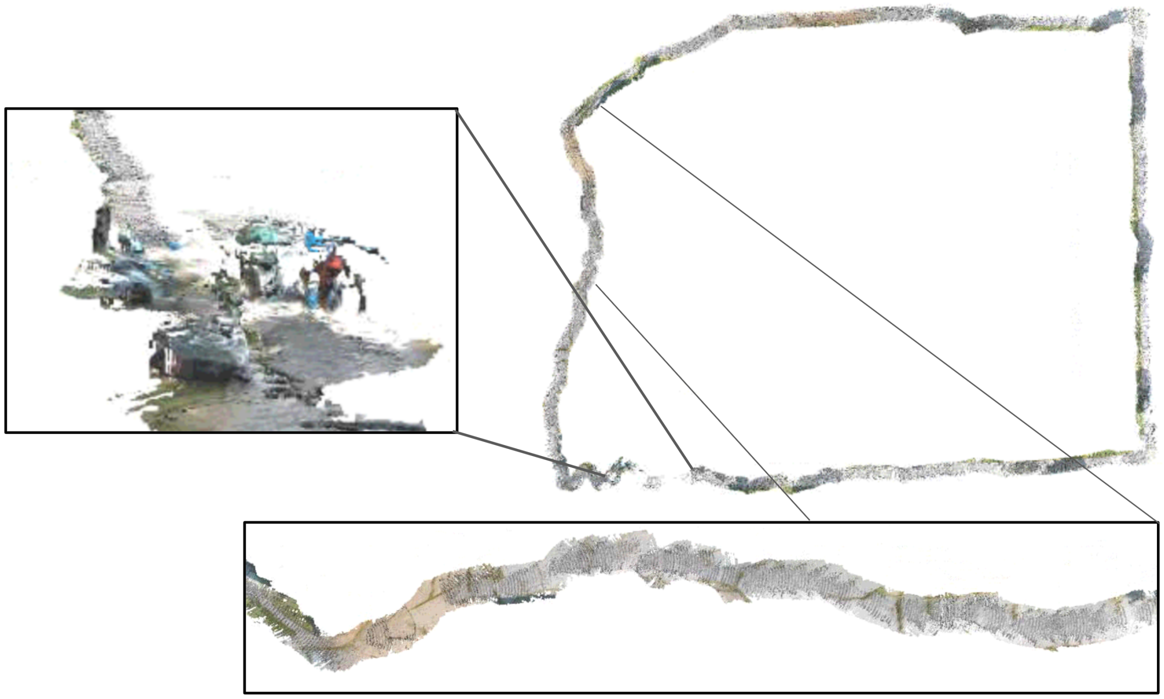
Real Robot Experiments

5.1 Experiments

The goal of these real experiments is to demonstrate the performance of Kimera on two challenging outdoor datasets, collected using a Clearpath Jackal UGV equipped with a forward-facing RealSense D435i RGBD Camera and IMU. These experiments were done outdoors, controlling the robot using teleoperation, while exploring large-scale areas. The first dataset was collected at the Medfield State Hospital, Massachusetts, USA. Three sets of trajectories were recorded, with the longest trajectory being 860 meters in length. The second dataset was collected around the Ray and Maria Stata Center at MIT, and also includes three different trajectories, each trajectory over 500 meters in length. Both sets of experiments are challenging and include many similar-looking scenes that induce spurious loop closures.

In the absence of ground truth on this experiments, we employed the end-to-end error metric, as in [39], by approximately returning the robot to its original position and computing the distance between the first and last estimated positions. This metric is less reliable than the absolute trajectory error, as the estimated trajectory could have wrong sections and still give a small end-to-end error particularly if it is able to find a loop closure between and start and end positions. To account for such cases, we also visualize the estimated trajectories overlaid on a satellite map of the

Figure 5-1: Mesh reconstruction at Medfield Hospital



scenarios. The end-to-end error, however, still provides useful information about the final estimation drift on each trajectory. We also took the opportunity to test the full Kimera stack on real data. Figure 5-1 shows the mesh reconstruction from one of the runs at Medfield.

5.2 Results

On the Medfield dataset Kimera-VIO accumulates a drift of approximately 15-25 m on each trajectory sequence. We note that the drift is mostly in the vertical direction, hence only partially visible in Figure 5-2. Through loop closures and Kimera-RPGO, Kimera significantly reduces the error. In comparison, the Stata dataset, shown in Figure 5-3, is more challenging, partially due to the lack of enough loop closure opportunities.

	Length(m)	VIO	ORB	SuperGlue
Stata Blue	610	29.35	**10.10**	10.19
Stata Orange	570	24.19	**15.14**	15.47
Stata Red	515	49.02	15.42	**0.13**
Medfield Blue	728	24.55	**0.09**	0.15
Medfield Orange	860	14.84	**1.77**	7.90
Medfield Red	600	**18.74**	19.82	19.61

Table 5.1: End-to-End Error (meter)

For five of these six trajectories, Kimera-VIO accumulates higher drifts than the optimized trajectories with loop closures, as shown in table 5.1. When comparing ORB and SuperGlue, we get similar performances, with two significant exceptions on the Medfield Orange set, in which ORB does significantly better than SuperGlue, and on the Stata Red set, in which SuperGlue does significantly better than ORB. Trajectory estimates are qualitatively correct: the optimized trajectories show better results than Kimera-VIO alone. There is no significant difference between the two methods.

5.3 Closing Remarks

These real experiments provided a great opportunity to also evaluate our loop closure modules on single-robot Kimera. Although these results show very impressive results for our single robot systems, both SuperGlue and ORB provide very similar results. This does not suggest a benefit in replacing ORB with SuperGlue, confirming the results shown in Chapter 4.

Figure 5-2: Medfield Trajectories

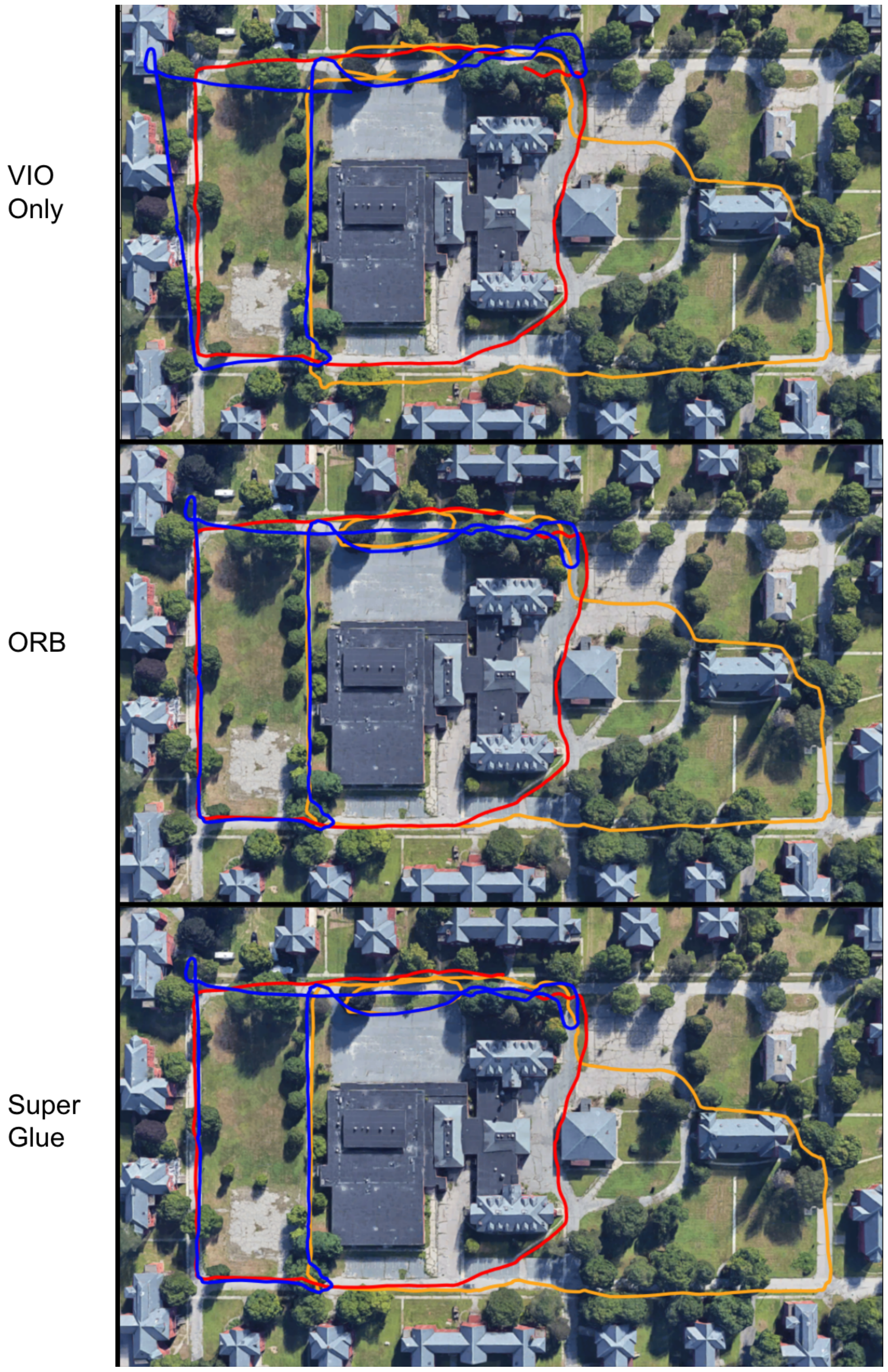
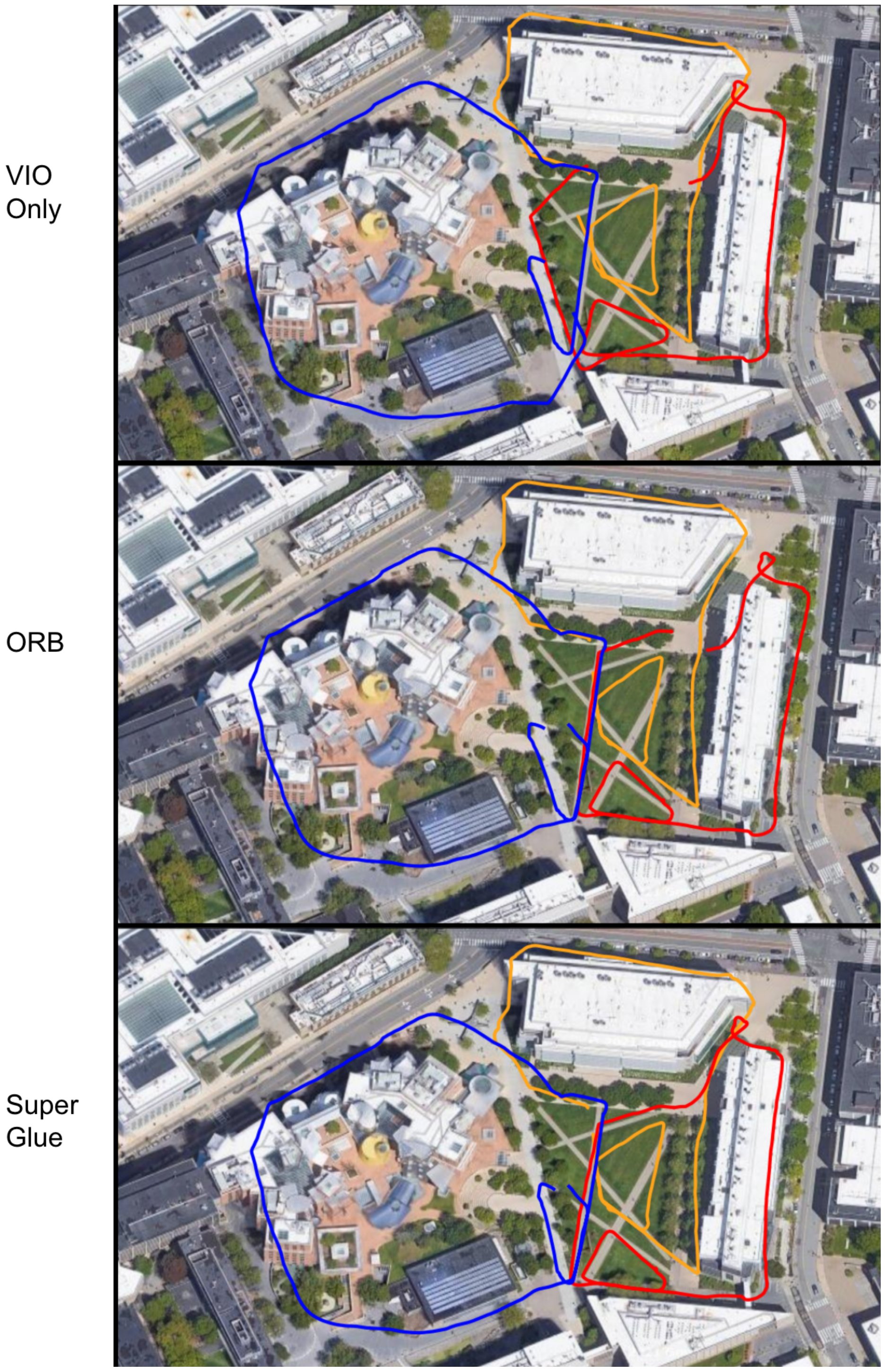


Figure 5-3: Stata Trajectories



Chapter 6

Conclusion

In this last chapter, we present possible directions for future work in learning-based loop closure detection. We end with a summary of our work, our results, and the general conclusions we can learn from them.

6.1 Future Work

The work done in this thesis is an initial evaluation of learning-based approaches for loop closure detection in a SLAM system. But the applications go beyond SLAM. Kimera itself is a modular system that could create metric-semantic meshes. These meshes also benefit from correction on the trajectory estimate and correct the meshes as well. Another example is Kimera-Multi, which presents a multi robot setup that would allow us to combine information from multiple agents for multi-robot SLAM and mesh reconstruction. A multi-robot system like Kimera-Multi requires a robust, reliable, and accurate inter-robot loop closure module. Any improvements we can achieve on loop closure detection will not only improve our single-robot trajectory estimation, but it would also improve the meshes and the entire multi-robot system too. The potential to improve loop closure detection, and in effect all the systems that rely on it, will continue to push us and other researchers to try new innovative methods and ideas. SuperPoint and SuperGlue did show us the ability to detect loops with wider baseline and small errors. However, our full evaluation in a SLAM system did not show the overwhelming results we would have liked to see. One possible, though speculative, explanation is that the greater amount of these *diverse* loop closures presents the possibility of a greater error with just few outliers that survive the outlier rejection step. Indeed, we see improvements when using very extreme (small) probabilistic thresholds for GNC.

Future work with SuperPoint and SuperGlue could focus on reducing some of the matches these methods produce while improving their quality. For instance, SuperGlue gives a score to every individual match (keypoint to keypoint). We currently select the medium score of all matches within two keyframe as the threshold to consider the match valid (thus, always selecting the top half of scores as valid SuperGlue matches). Relaxing this threshold would lead to many more matches, which we expected GNC to filter. Testing stricter criteria to reject incorrect matches could thus be a productive line of future research. Additionally, testing SuperPoint and SuperGlue in other SLAM systems, or testing new learning-based methods on Kimera could be promising paths to see improvements using learning-based approaches on SLAM systems.

6.2 Closing Remarks

In this thesis, we presented an evaluation of the loop closures capabilities of both ORB, a state-of-the-art method for feature extraction and feature matching, and SuperPoint with SuperGlue, novel pretrained learning-based approaches for feature extraction and feature matching respectively. In our related work section, we described the history of these algorithms and how we arrived at ORB as a state-of-the-art approach that performs well and runs in real-time on CPU. Recently, SuperPoint and SuperGlue have shown very promising capabilities that can outperform ORB. Our results from analyzing loop closures in isolation between these two methods confirm that SuperPoint indeed detects a set of richer key-features while SuperGlue matches keyframes with wider baselines. However, when integrating these methods in Kimera, a full SLAM system, we found no statistically significant evidence that SuperPoint and SuperGlue can drastically improve the system. Indeed, they perform similarly to ORB in benchmarking, simulated, and real datasets. In some cases, one method would do slightly better than the other. However, it is also important to remember that while ORB provides real-time performance on CPU for Kimera, which also runs in real-time on CPU, SuperPoint and SuperGlue are neural networks that require GPU in order to maintain real-time performance. The additional computational power required for these methods adds to the cost of using them, while the performance when looking at the absolute trajectory errors on EuRoc and simulated data, and end-to-end errors on the real data, do not show any benefits of doing so. Our evaluation suggests that, while the SuperPoint and SuperGlue detect many more loop closures with wider baselines, when integrated in a SLAM system, their use does not lead to substantial performance improvements compared to classical methods, such as ORB feature matching.

Appendix A

Kimera Parameters

	EuRoc	**Simulation**	**Jackal**
backend_modality	0	0	0
autoInitialize	1	0	1
roundOnAutoInitialize	0	0	0
initialPositionSigma	1e-05	1e-05	1e-05
initialRollPitchSigma	0.174533	0.174533	0.174533
initialYawSigma	0.0017453	0.0017453	0.0017453
initialVelocitySigma	0.001	0.001	0.001
initialAccBiasSigma	0.1	0.1	0.0001
initialGyroBiasSigma	0.01	0.01	0.001
linearizationMode	0	0	0
degeneracyMode	1	1	1
rankTolerance	1	1	1
landmarkDistanceThreshold	10	100	20
outlierRejection	3	3	10
retriangulationThreshold	0.001	0.001	0.001
smartNoiseSigma	3.0	a	3.0
monoNoiseSigma	1.8	1.8	1.8
monoNormType	2	2	2
monoNormParam	4.6851	4.6851	4.6851
stereoNoiseSigma	1.8	1.8	1.8
stereoNormType	2	2	2
stereoNormParam	4.6851	4.6851	4.6851
regularityNoiseSigma	0.03	0.03	0.3
regularityNormType	1	1	1
regularityNormParam	0.04	0.04	0.04
addBetweenStereoFactors	1	1	0
betweenRotationPrecision	0	0	0
betweenTranslationPrecision	100	100	100

Table A.1: Back-End Parameters

	EuRoc	**Simulation**	**Jackal**
klt_win_size	24	24	24
klt_max_iter	30	30	30
klt_max_level	4	4	4
klt_eps	0.1	0.01	0.1
maxFeatureAge	25	60	25
maxFeaturesPerFrame	300	400	400
quality_level	0.001	0.001	0.001
min_distance	20	8	20
block_size	3	3	3
use_harris_detector	0	0	0
k	0.04	0.04	0.4
fast_thresh	10	10	10
equalizeImage	0	0	1
nominalBaseline	0.11	0.6	0.05
toleranceTemplateMatching	0.15	0.15	0.15
templ_cols	101	101	101
templ_rows	11	11	11
stripe_extra_rows	0	0	0
minPointDist	0.5	0.5	0.3
maxPointDist	10	100	10
bidirectionalMatching	0	0	0
enable_non_max_suppression	0	1	1
non_max_suppression_type	4	4	4
enable_subpixel_corner_finder	1	1	1
max_iters	40	40	40
epsilon_error	0.001	0.001	0.001
window_size	10	10	10
zero_zone	-1	-1	-1
subpixelRefinementStereo	0	0	0
useSuccessProbabilities	1	1	1
useRANSAC	1	1	1
minNrMonoInliers	10	10	10
minNrStereoInliers	5	5	5
ransac_threshold_mono	1e-06	0.1	1e-06
ransac_threshold_stereo	1	1	0.8
ransac_use_1point_stereo	1	1	0
ransac_use_2point_mono	1	1	0
ransac_max_iterations	100	200	500
ransac_probability	0.995	0.995	0.990
ransac_randomize	0	0	0
intra_keyframe_time	0.2	0.2	0.1
minNumberFeatures	0	0	0
useStereoTracking	1	1	1
disparityThreshold	0.5	0.0	0.5

Table A.2: Front-End Parameters

	EuRoc	**Simulation**	**Jackal**
use_nss	1	0.1	0.1
alpha	0.1	0.1	0.3
min_temporal_matches	3	3	3
recent_frames_window	60	60	90
max_db_results	50	50	50
min_nss_factor	0.05	0.05	0.05
min_matches_per_island	1	1	1
max_intraisland_gap	3	3	3
max_nrFrames_between_islands	3	3	3
max_nrFrames_between_queries	2	2	2
geom_check_id	0	0	0
min_correspondences	12	12	6
max_ransac_iterations_mono	500	500	500
ransac_probability_mono	0.995	0.995	0.995
ransac_threshold_mono	1e-5	1e-5	1e-5
ransac_randomize_mono	1	0	1
ransac_inlier_threshold$_mono$	0.01	0.01	0.01
pose_recovery_option_id	0	0	0
max_ransac_iterations_stereo	500	500	500
ransac_probability_stereo	0.995	0.995	0.995
ransac_threshold_stereo	0.3	0.3	0.3
ransac_randomize_stereo	1	0	1
ransac_inlier_threshold_stereo	0.3	0.3	0.3
use_mono_rot	0	0	0
refine_pose	1	1	1
nfeatures	1000	1000	1000
scale_factor	1.2	1.2	1.2
nlevels	8	8	8
edge_threshold	31	31	31
first_level	0	0	0
WTA_K	2	2	2
score_type_id	0	0	0
patch_sze	31	31	31
fast_threshold	20	20	20
betweenRotationPrecision	100000	100000	100000
betweenTranslationPrecision	10000	10000	10000
gnc_alpha	0.7	0.7	0.7
max_lc_cached_before_optimize	10	10	10

Table A.3: Loop Closure Parameters

	EuRoc	**Simulation**	**Jackal**
rate_hz	200	200	400
gyroscope_noise_density	1.6968e-04	1.6968e-04	0.00488
gyroscope_random_walk	1.9393e-05	1.9393e-05	4.88e-04
accelerometer_noise_density	2.0000e-3	2.0000e-3	0.1
accelerometer_random_walk	3.0000e-2	3.0000e-3	0.0147
do_imu_rate_time_alignment	1	1	1
time_alignment_window_size_s	10.0	10.0	10.0
time_alignment_variance_threshold_scaling	30.0	0.0	0.0
imu_integration_sigma	1.0e-8	1.0e-8	1.0e-8
imu_time_shift	0.0	0.0	0.0
n_gravity (z-direction)	-9.81	-9.81	-9.81

Table A.4: IMU Parameters

Bibliography

[1] T. Malisiewicz D. DeTone and A. Rabinovich. Superpoint: Self-supervised interest point detection and description. *CVPR Workshop on Deep Learning for Visual SLAM*, 2018.

[2] T. Malisiewicz P. Sarlin, D. DeTone and A. Rabinovich. Superglue: Learning feature matching with graph neural networks. *IEEE Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[3] A. Rosinol, M. Abate, Y. Chang, and L. Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. *International Conference on Robotics and Automation (ICRA)*, 2020.

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[5] K. Konolige E. Rublee, V. Rabaud and G. Bradski. “orb: an efficient alternative to sift or surf. *EEE International Conference on Computer Vision (ICCV)*, page 2564–2571, 2011.

[6] D. Galvez-Lopez and J. D. Tard. “bags of binary words for fast place recognition in image sequence. *IEEE Transactions on Robotic*, page 1188–119, 2012.

[7] J. How Y. Chang, Y. Tian and L. Carlone. Kimera-multi: a system for distributed multi-robot metric-semantic simultaneous localization and mapping. *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2021.

[8] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone. Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection. 5(2):1127–1134. ICRA Best paper award in Robot Vision.

[9] Y. Tian, Y. Chang, F. Herrera Arias, C. Nieto-Granda, J. P. How, and L. Carlone. Kimera-multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems. *CoRR*, arxiv.org/abs/2106.14386, 2021.

[10] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, page 91–110, 2004.

[11] L. Van Gool H. Bay, T. Tuytelaars. Surf: Speeded up robust features. *European Conference on Computer Vision*, 2006.

[12] E. Rosten and T. Drummond. Machine learning for highspeed corner detection. *European Conference on Computer Vision*, 1, 2006.

[13] C. Strecha M. Calonder, V. Lepetit and P. Fua. Brief: Binary robust independent elementary features. *European Conference on Computer Vision*, 2010.

[14] C. Harris and M. Stephens. A combined corner and edge detector. *Alvey Vision Conference*, page 147–151, 1988.

[15] M. Fischler and R. Bollesrd. “random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. ACM*, 24:381–395, 1981.

[16] C. Olsson O. Enqvist, F. Kahl. Non-sequential structure from motion. *Intl. Conf. on Computer Vision (ICCV)*, page 264– 271, 2011.

[17] S. Galliani T. Sattler K. Schindler M. Pollefeys A. Geiger T. Schops, J. L. Schonberger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. *IEEE Conference on Computer Vision and Patter Recognition (CVPR)*, 2017.

[18] S. Oprea V. Villena-Martinez Garcia-Garcia, S. Orts-Escolano and J. Garcıa-Rodrıguez. A review on deep learning techniques applied to semantic segmentation. *ArXiv Preprint: 1704.06857*, 2017.

[19] I. Sutskever Krizhevsky and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NIPS)*, page 1097– 1105, 2012.

[20] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, page 6517–6525, 2017.

[21] R. Girshick S. Ren, K. He and J. Sun. Faster r-cnn: Towards realtime object detection with region proposal networks. *Advances in Neural Information Processing Systems (NIPS)*, page 91–99, 2015.

[22] P. Dollar K. He, G. Gkioxari and R. Girshick. Mask r-cnn. *Intl. Conf. on Computer Vision (ICCV)*, page 2980–2988, 2017.

[23] P. Dollar R. Hu and K. He. Learning to segment every thing. *Intl. Conf. on Computer Vision (ICCV)*, page 4233–4241, 2017.

[24] A. Kendall V. Badrinarayanan and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 2017.

[25] J. Neira and J. Tardo. "data association in stochastic mapping using the joint compatibility test. *IEEE Trans. Robot. Automat.*, 17(6):890–897, 2001.

[26] G. Agamennoni M. Bosse and I. Gilitschenski. "robust estimation and applications in robotics. *Foundations and Trends in Robotics*, 4(4):225–269, 2016.

[27] N. Sünderhauf and P. Protzel. Switchable constraints for robust pose graph slam. *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[28] L. Spinello C. Stachniss P. Agarwal, G. D. Tipaldi and W. Burgard. Robust map optimization using dynamic covariance scaling. *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2013.

[29] L. Paz J. Casafranca and P. Piniés. A back-end l_1 norm based solution for factor graph slam. *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, page 17–23, 2013.

[30] A. Chatterjee and V. M. Govindu. Efficient and robust large-scale rotation averaging. *Intl. Conf. on Computer Vision (ICCV)*, page 521–528, 2013.

[31] ——. Robust relative rotation averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):958–972, 2018.

[32] K. Khosoussi G. Hu and S. Huang. Towards a reliable slam back-end. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 37–43, 2013.

[33] E. Olson and P. Agarwal. Inference on networks of mixtures for robust robot mapping. *Robotics: Science and Systems (RSS)*, 2012.

[34] M. Pfingsthorn and A. Birk. Simultaneous localization and mapping with multimodal probability distributions. *Intl. J. of Robotics Research*, 32(2):143–171, 2013.

[35] ——. Generalized graph slam: Solving local and global ambiguities through multimodal and hyperedge constraints. *Intl. J. of Robotics Research*, 35(6):601–630, 2016.

[36] F. Dellaert et al. "georgia tech smoothing and mapping (gtsam). *https://gtsam.org/*, 2019.

[37] R. M. Eustice J. G. Mangelson, D. Dominic and R. Vasudevan. "pairwise consistent measurement set maxi- mization for robust multi-robot map merging. *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, page 2916–2923, 2018.

[38] P. Gohl T. Schneider J. Rehder S. Omari M. Achtelik M. Burri, J. Nikolic and R. Siegwart. "the euroc micro aerial vehicle datasets. *Intl. J. of Robotics Research*, 2016.

[39] F. Dellaert C. Forster, L. Carlone and D. Scaramuzza. "on-manifold preintegration theory for fast and accurate visual-inertial navigation. *IEEE Trans. Robotics*, 33(1):1–21, 2017.

[40] D. Nister. "an efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Machine Intell.*, 26(6):756–770, 2004.

[41] B. Horn. "closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Amer.*, 4(4):629–642, 1987.

[42] O. Barinova K. Anton D. Prokhorov, D. Zhukov and A. Vorontsova. Measuring robustness of visual slam. *2019 16th International Conference on Machine Vision Applications (MVA)*, pages 1–6, 2019.