Degree Project in Systems, Control and Robotics
Second cycle, 30 credits

Outlier Robustness in Server-Assisted Collaborative SLAM
Evaluating Outlier Impact and Improving Robustness

JOSÉ MIGUEL DE ALMEIDA PEDRO
Outlier Robustness in Server-Assisted Collaborative SLAM

Evaluating Outlier Impact and Improving Robustness

JOSÉ MIGUEL DE ALMEIDA PEDRO

Master’s Programme, Systems, Control and Robotics, 120 credits
Date: June 27, 2023

Supervisors: David Umsonst, Roberto Castro Sundin, Patric Jensfelt
Examiner: Mårten Björkman

School of Electrical Engineering and Computer Science
Host company: Ericsson AB

Swedish title: Robusthet mot outliers i serverassisterad, samarbetande SLAM
Swedish subtitle: En utvärdering utav outliers påverkan och hur robustheten kan ökas
Abstract

In order to be able to perform many tasks, autonomous devices need to understand their environment and know where they are in this environment. Simultaneous Localisation and Mapping (SLAM) is a solution to this problem. When several devices attempt to jointly solve this problem they use Collaborative SLAM (C-SLAM), but this is a very resource-demanding process. In order to enable resource-constrained devices, like small mobile robots or eXtended Reality (XR) devices, to run C-SLAM we look towards a Server-Assisted C-SLAM architecture to lift the computational burden from these devices.

In a real-world scenario, sensors might fail, the devices might process sensor data wrongly or a malicious actor might inject wrong data into the system. In order for these solutions to be reliable, they must be able to deal with these outliers.

This thesis looks into the impact of outliers in Server-Assisted C-SLAM algorithms and presents two novel solutions for a robust algorithm, based on robust estimation of the initial device poses. We show the novel solutions outperform the state of the art both in estimation accuracy, yielding better estimates of the real device trajectories, and computational performance, making it suitable for device-constrained devices.

Keywords

SLAM, Robust Estimation, Multi-Device Algorithms
**Sammanfattning**

För att kunna utföra flertalet uppgifter måste autonoma enheter förstå sin miljö och veta var de befinner sig i den här miljön. Simultaneous Localization and Mapping (SLAM) är en lösning på detta problem. När flera enheter försöker lösa detta problem tillsammans använder de Samarbetande SLAM (C-SLAM), men detta är en mycket resurskrävande process. För att möjliggöra att resursbegränsade enheter, så som exempelvis små mobila robotar eller eXtended Reality (XR)-enheter, ska kunna köra C-SLAM föreslås en serverassisterar C-SLAM-arkitektur beräkningsbördan kan lyftas från dessa enheter till servern.

I ett verkligt scenario kan sensorer vara felaktiga, enheter behandla sensordata felaktigt eller illvilliga aktörer injicera felaktig data i systemet. Därför undersöker detta arbete effekten av outliers i Serverassisterade C-SLAM-algoritmer och presenterar två nya lösningar för en robust algoritm, baserad på robusta uppskattningar av enhetens initiala positioner. Denna lösning visar sig överträffa likartade lösningar i litteraturen både vad gäller uppskattningsnoggrannhet, vilket ger bättre uppskattningar av den verkliga enhetsbanor och beräkningsprestanda, vilket gör den lämplig för enheter med begränsade resurser.

**Nyckelord**

SLAM, Robust uppskattning, Algoritmer för flera enheter
Acknowledgments

I would like to begin by thanking my supervisors, David, Roberto and Patric, who made the process of researching and writing this thesis always feel engaging and fun. Your unwavering support and availability to discuss ideas and provide amazing feedback were a vital part of the last 5 months and without them, this thesis would certainly look much different. I feel lucky to have such great supervisors.

I would also like to thank everyone in our team at Ericsson for being so welcoming and making the time I’ve spent here so fun. Thank you to my colleagues in our thesis supervision group for all the great feedback.

Finally, a special thanks to my family and friends, in particular to my parents. Thank you for the unending and unrelenting support, not just during the thesis but every and anytime I needed it.

Obrigado!

Stockholm, June 2023

Zé
Contents

1 Introduction 1
  1.1 Background ............................................. 2
  1.2 Problem ............................................. 4
    1.2.1 Scientific and engineering issues ................. 4
  1.3 Purpose & Goals ...................................... 4
  1.4 Delimitations ........................................ 6
  1.5 Structure of the thesis ............................... 6

2 Background and Related Work 7
  2.1 Simultaneous Localisation and Mapping ................. 7
  2.2 Collaborative SLAM .................................... 11
  2.3 Server-Assisted SLAM .................................. 15
  2.4 Robust Estimation for SLAM ......................... 17
  2.5 Summary ............................................. 22

3 Baseline & Evaluation Framework 23
  3.1 Baseline ............................................. 23
  3.2 Evaluation Framework .................................. 27
  3.3 Data Generation ....................................... 31
  3.4 Summary ............................................. 36

4 Outlier Impact on Server-Assisted C-SLAM 37
  4.1 Robust C-SLAM Implementation ....................... 37
  4.2 Robust SLAM vs Robust C-SLAM ....................... 40
  4.3 Experimental Design & Results ....................... 41
  4.4 Summary ............................................. 47

5 Robust Frame Estimation 49
  5.1 Robust Frame Estimator ............................... 50
  5.2 Further Relaxation of the Robust Frame Estimator .... 53
### 6 Conclusions and Future work

6.1 Conclusions .......................... 67
  6.1.1 Impact & Reflection ............... 68

6.2 Limitations & Future Work .......... 68

References .......................... 71

A Outlier-Free Dataset Generation ..... 77

B 3-Dimensional Special Euclidean Group 81
  B.1 Noisy Measurements ................. 82
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>SLAM Pipeline</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>SLAM functional diagram</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Example factor graph for SLAM</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Server-Assisted SLAM architecture</td>
<td>15</td>
</tr>
<tr>
<td>2.4</td>
<td>Reduced pose graph generation</td>
<td>16</td>
</tr>
<tr>
<td>3.1</td>
<td>Server-Assisted C-SLAM multi-device architecture</td>
<td>24</td>
</tr>
<tr>
<td>3.2</td>
<td>Example reference and estimated trajectories</td>
<td>27</td>
</tr>
<tr>
<td>3.3</td>
<td>Alignment for the Individual Error</td>
<td>29</td>
</tr>
<tr>
<td>3.4</td>
<td>Alignment for the Collaborative Error</td>
<td>30</td>
</tr>
<tr>
<td>3.5</td>
<td>Ground truth and noisy trajectory</td>
<td>32</td>
</tr>
<tr>
<td>3.6</td>
<td>Example trajectory for two devices</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>Individual Error with Random outliers</td>
<td>42</td>
</tr>
<tr>
<td>4.2</td>
<td>Collaborative Error with Random outliers</td>
<td>43</td>
</tr>
<tr>
<td>4.3</td>
<td>Mean Iteration Times with Random outliers</td>
<td>43</td>
</tr>
<tr>
<td>4.4</td>
<td>Mean Iteration Times with Random outliers, excluding GNC</td>
<td>44</td>
</tr>
<tr>
<td>4.5</td>
<td>Individual Error with Local Grouped outliers</td>
<td>45</td>
</tr>
<tr>
<td>4.6</td>
<td>Collaborative Error with Local Grouped outliers</td>
<td>45</td>
</tr>
<tr>
<td>4.7</td>
<td>Mean Iteration Times with Local Grouped outliers</td>
<td>46</td>
</tr>
<tr>
<td>4.8</td>
<td>Collaborative Error with Random outliers and priors</td>
<td>46</td>
</tr>
<tr>
<td>5.1</td>
<td>Proposed robust C-SLAM architecture</td>
<td>49</td>
</tr>
<tr>
<td>5.2</td>
<td>Example two-device pose graph</td>
<td>51</td>
</tr>
<tr>
<td>5.3</td>
<td>Relaxed pose graph by removing intra-agent loop closures</td>
<td>52</td>
</tr>
<tr>
<td>5.4</td>
<td>Simplification of relaxed pose graph by condensing odometry</td>
<td>52</td>
</tr>
<tr>
<td>5.5</td>
<td>Pose graph with independent loop-closures</td>
<td>54</td>
</tr>
<tr>
<td>5.6</td>
<td>More relaxed pose graph</td>
<td>54</td>
</tr>
<tr>
<td>5.7</td>
<td>Putting RFE/RFE:) together with the rest of the system</td>
<td>57</td>
</tr>
</tbody>
</table>
5.8 Collaborative Error for different RFE tuning parameters . . . 58
5.9 Collaborative Error for different RFE:) tuning parameters . . . 58
5.10 Collaborative Error for the incremental approaches . . . . . 59
5.11 Computational performance for the incremental approaches . 60
5.12 Computational performance for the incremental approaches,
RFE excluded . . . . . . . . . . . . . . . . . . . . . . . . . .60
5.13 Collaborative Error comparing against the batch approach . . 61
5.14 Collaborative Error scaling with the number of devices . . . 62
5.15 Computational performance scaling with the number of devices 63
5.16 Ablation study of the accuracy of RFE . . . . . . . . . . . 64
5.17 Ablation study of the accuracy of RFE, with filtering . . . . 64
5.18 Ablation study of the accuracy of RFE:) . . . . . . . . . . 65
5.19 Ablation study of the accuracy of RFE:), with filtering . . . 65
A.1 Ground truth trajectory on a grid world . . . . . . . . . . . 78
B.1 Chain of odometry measurements . . . . . . . . . . . . . . 83
List of Tables

3.1 Baseline Parameters .............................................. 26
4.1 Non-outlier dataset parameters ............................... 41
4.2 Outlier generation parameters ................................. 44
xii | List of Tables
## List of acronyms and abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATE</td>
<td>Absolute Trajectory Error</td>
</tr>
<tr>
<td>C-SLAM</td>
<td>Collaborative SLAM</td>
</tr>
<tr>
<td>DCS</td>
<td>Dynamic Covariance Scaling</td>
</tr>
<tr>
<td>GNC</td>
<td>Graduated Non-Convexity</td>
</tr>
<tr>
<td>iCATE</td>
<td>Incremental Collaborative ATE</td>
</tr>
<tr>
<td>IRLS</td>
<td>Iteratively Reweighted Least Squares</td>
</tr>
<tr>
<td>iSAM2</td>
<td>Incremental Smoothing and Mapping 2</td>
</tr>
<tr>
<td>MAP</td>
<td>Maximum a Posteriori</td>
</tr>
<tr>
<td>MLE</td>
<td>Maximum Likelihood Estimation</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
</tr>
<tr>
<td>PCM</td>
<td>Pairwise Consistency Maximisation</td>
</tr>
<tr>
<td>PGO</td>
<td>Pose Graph Optimisation</td>
</tr>
<tr>
<td>RFE</td>
<td>Robust Frame Estimator</td>
</tr>
<tr>
<td>RFE:)</td>
<td>More Relaxed Robust Frame Estimator</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Squared Error</td>
</tr>
<tr>
<td>SLAM</td>
<td>Simultaneous Localisation and Mapping</td>
</tr>
<tr>
<td>TLS</td>
<td>Truncated Least Squares</td>
</tr>
<tr>
<td>VIO</td>
<td>Visual-Inertial Odometry</td>
</tr>
<tr>
<td>XR</td>
<td>eXtended Reality</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The Simultaneous Localisation and Mapping (SLAM) problem has its origin in robotics [1, 2] and consists of how to estimate a map and the trajectory followed by the robot on that map from the sequence of sensor readings over time. Although the problem was first defined in a robotics context, its significance has increased in recent years due to its applicability in the field of eXtended Reality (XR) [3].

In scenarios where there is more than one device, which can be a robot, an XR device, or something else, it is often beneficial to consider an augmented version of SLAM, Collaborative SLAM (C-SLAM). This way, instead of each device solving its own SLAM problem, all the devices collectively solve the C-SLAM problem by sharing some information. By doing this, they can benefit from the fact that some of the other devices might have visited the same places, which allows the different devices to establish measurements that relate their poses [4]. This contrasts with traditional, single-device SLAM, where these kinds of measurements can only be established between the device’s poses at different points in time.

SLAM and C-SLAM have been studied at length and a lot of progress has been made in developing solutions which can produce very good estimates. However, these solutions are very computationally demanding and therefore not suited to be run on small mobile devices, which are typically quite resource-constrained. In order to address this issue, research has been done looking into offloading computations to a server [5], reducing the computational effort of the agents, leading to Server-Assisted SLAM.

Typically, the sensor readings are noisy, i.e., they are a function of the true value of the property being measured that is corrupted by some noise. Statistical analysis allows us to get good estimates of that value from the
measurement. However, sometimes completely spurious measurements are obtained from which little to no information about the true value can be recovered. These spurious measurements are called outliers. Since we cannot obtain good estimates for the true value from an outlier, we want to detect them and remove them from the set of used measurements.

The topic of this thesis is the study of the impact that outliers have on Server-Assisted C-SLAM and the development of a novel solution for the outlier robust Server-Assisted C-SLAM problem that is suitable for resource-constrained devices. In order to address this, first the results concerning the impact of outliers on the baseline method and in naive robust implementations are presented in Chapter 4. Then, motivated by these results, two new methods for robust C-SLAM are presented and evaluated in Chapter 5.

1.1 Background

The goal of SLAM is to estimate a map and the trajectory of a device in that map. A trajectory is a collection of poses, indexed in time. A device’s pose consists of their position and orientation, which can be in either 2 or 3-dimensional space. Without loss of generality, only 3-dimensional poses will be considered in this work.

Modern SLAM algorithms are functionally separated into two parts, the front-end and the back-end. The front-end is responsible for taking the sensor readings and processing them into spatial measurements.

The back-end’s function is to take the measurements generated by the front-end, which constrain the device’s poses at different time instants, and estimate the device’s trajectory. Figure 1.1 shows a diagram illustrating the functional separation of a usual SLAM algorithm.

![Figure 1.1: Pipeline of a SLAM algorithm.](image)

Depending on the type of sensor, the front-end can do different things. For
example, if the sensor is a camera, the front-end might try to find identifiable features in the image and associate them with previous images. If the sensor is a LIDAR, then the front-end might attempt to find what is the transformation between point clouds to associate different sensor readings. Through these associations, measurements can be obtained which relate the position of the device at different times. This association of sensor readings at different times is normally dependent on estimates of the device’s poses at those times in order to determine the likelihood that the association is valid. Thus, there is a very tight coupling between the front-end, which processes the sensor information, and the back-end, which estimates the device’s trajectory from the processed measurements.

In the back-end, measurements are classified into two groups. The first group is *odometry*, which are measurements that relate the device’s pose in two consecutive time instants. The second group are *loop closures*, which are the remaining measurements, relating the device’s poses between longer intervals of time. Loop closures are very important for SLAM since they are the key element in allowing the estimates to not accumulate odometry measurement noise indefinitely. To understand this, think of a scenario where only odometry is available, informing the device that it moved a certain amount in a certain direction since the last pose. Since this information has some uncertainty and it only relates consecutive poses, after some odometry measurements the uncertainty has accumulated and the device is quite unsure of where it is. However, when a loop closure informs the device that its current pose is related to an older one, it constrains the current pose with the uncertainty of that measurement and the older pose, which is typically much lower than what has been accumulated through odometry. Additionally, this reduced uncertainty can also be propagated backwards, by accumulating the odometry uncertainty backwards, while it is lower than the forward accumulated odometry uncertainty. Therefore, loop closures are vital in keeping the pose uncertainties bounded in a SLAM problem.

When talking about C-SLAM, measurements relating two different devices’ poses are also called loop closures. In this case, we distinguish them by calling them *intra-agent* loop closures and *inter-agent* loop closures, respectively.

A lot of research has been dedicated to making front-ends more robust to outliers, which in the context of the front-end are usually wrong associations, like in [6]. However, less attention has been given to robustifying the back-end, which is also very important given the tight coupling already mentioned [7]. When talking about the back-end, outliers are wrong loop closures and
odometry is assumed to be outlier-free. This assumption is discussed in further detail in Section 2.1.

In order to make the back-end robust, we look towards the field of Robust Estimation for methods which can deal with outliers. Given that algorithms for robust estimation are very computationally demanding [8], coupled with the target application being resource-constrained devices, we look towards Server-Assisted SLAM [5] to alleviate the device’s computational resources.

The background for this problem is explored in further depth in Chapter 2.

1.2 Problem

This thesis concerns itself with two problems. The first is investigating the impact that outliers have in the back-end of a Server-Assisted C-SLAM algorithm. The second is designing a novel and outlier-robust solution to the Server-Assisted C-SLAM back-end problem, under the requirement that the developed solution is suitable for online applications.

1.2.1 Scientific and engineering issues

The main scientific issue lies in the development of a Robust Server-Assisted C-SLAM solution, given that existing robust solutions are either not Server-Assisted [9] or not truly C-SLAM [10, 11]. The main engineering issue is the real-time constraint, given that the purpose of the developed solution is online applications, but existing robust methods are either too computationally demanding [12] or show unsatisfactory robustness in the collaborative setting [13].

1.3 Purpose & Goals

The thesis presented and defended in this project benefits robotics and XR manufacturers and developers, and subsequently the users of those devices. They benefit since the devices can be made smaller, lower power and less resource hungry, making them cheaper and more comfortable to use. This comes at the cost of requiring servers, but the social and practical complexities of this cost are much simpler to address.

The adoptability of mobile robots and XR devices depends on several factors. The first of them is their correct and reliable functioning under all circumstances. For this to be achieved, robust estimation is very important,
since sensor failures and malicious actors, among other scenarios, will interfere with the functioning of the devices due to outliers. The second factor is comfort, for XR devices, and autonomy, and a very big part of this is size and power. In order to be able to further miniaturise devices and make them low-power, solutions have to be found that allow running demanding algorithms like C-SLAM on resource-constrained devices. The hypothesis in this work is that one way of achieving this is through Server-Assisted C-SLAM. The goals of this project can be divided into the following:

1. Quantitatively characterise the impact of outliers in existing non-robust Server-Assisted C-SLAM solutions, as well as naive robust solutions.

2. Develop a solution which surpasses the state of the art in Robust Server-Assisted C-SLAM.

The goals this project aims to achieve have the possibility of having a great social impact. By helping transform low-powered, resource-constrained devices into C-SLAM capable devices, access to XR can get democratised and give everyone a chance to be involved and benefit from the richness of information available in the Metaverse. On the mobile robotics side, resource-efficient C-SLAM enables the use of, e.g., small drones in tight, GPS-denied environments, like disaster sites where existing larger mobile robots cannot reach.

While the perspectives of societal benefit are exciting, the environmental and ethical implications must also be considered. Regarding environmental sustainability, it can be expected for the project to make a positive impact, since the reduction in necessary power in mobile devices means smaller batteries can be used. Batteries play an important part in the environmental impact of electronic devices, in part due to the difficulty of recycling them. Meanwhile, the computational power required is transferred to servers, which do not require batteries and are typically more energy-efficient, for the same computations.

However, in order to deploy a C-SLAM system in the real world, measures must be taken in order to guarantee that is done ethically. By making SLAM capable devices ubiquitous, a lot of data can be collected and this has a big ethical and social risk which must be taken into consideration. For example, a data leak in a C-SLAM system might let malicious actors know a device’s position which it should not or know how an area of the world looks like when it should be private. Therefore, precautions must be taken to ensure the safety of data collection and storage, as well as make sure that the data is not being used for nefarious purposes.
1.4 Delimitations

As a baseline for Server-Assisted C-SLAM algorithms, the algorithm developed at Ericsson is going to be used as representative of general Server-Assisted C-SLAM solutions. This algorithm is an extension of [5] to C-SLAM and is introduced in Section 2.3.

This thesis focuses on exploiting the structure of the C-SLAM problem to design relaxed problems, based on the full C-SLAM problem. These relaxed problems, by being easier, allow the utilisation of more robust, but computationally more expensive, numerical optimisation methods. However, our work does not look into the development of more robust or more efficient numerical optimisation methods.

1.5 Structure of the thesis

Chapter 2 introduces the foundational theory behind SLAM solutions and presents the state of the art in related work.

Chapter 3 introduces the baseline, discusses existing result evaluation metrics for SLAM and how they can be extended to incremental C-SLAM and presents the data generation method.

Chapter 4 deals with evaluating the impact of outliers in Server-Assisted C-SLAM solutions. The importance of priors for the devices’ initial relative poses is identified and discussed.

Chapter 5 introduces the methods developed for tackling the issue of the absence of priors in the robust C-SLAM problem. These methods are described in detail and results are shown demonstrating their effectiveness.

Chapter 6 summarises the conclusions of this work, discusses their implication and discusses what can be expanded upon for future work.
Chapter 2

Background and Related Work

This chapter provides background information about the SLAM problem, its extension to C-SLAM, Server-Assisted SLAM and the field of robust estimation. The theory behind these topics is summarised and the state of the art is presented.

2.1 Simultaneous Localisation and Mapping

SLAM is typically formulated as the Maximum a Posteriori (MAP) estimation [4] of the device trajectory, $\mathcal{X}$, and the map, $\mathcal{M}$, given the sensor data, $\tilde{Z}$,

$$(\mathcal{X}^*, \mathcal{M}^*) = \arg\max_{\mathcal{X}, \mathcal{M}} p(\mathcal{X}, \mathcal{M} | \tilde{Z}).$$  \hspace{1cm} (2.1)

The trajectory is a sequence of poses, $\mathcal{X} = \{X_i\}_{i=0}^T$, consisting of the device’s position and orientation, represented by, e.g., the cartesian coordinates, $(x, y, z)$, and the Euler angles, $(\alpha, \beta, \gamma)$, respectively.

Maps can have very different representations. These can vary between dense representations, like occupancy grid and voxel maps [14], sparse representations, like feature maps [15], or hybrid representations, like hierarchical maps [16].

However, SLAM is never solved directly due to its computational intractability and is instead broken up into components which are easier to solve. The first major division, as mentioned in Chapter 1, is between front-end and back-end.

The front-end is tasked with processing the raw sensor data into spatial measurements, i.e., measurements which describe spatial constraints between poses or between poses and the map. These measurements, $Z = \{z_i\}_{i=1}^M$, can
be, e.g., the bearing and range to another pose or some part of the map. Thus, the front-end transforms the sensor data, $\tilde{Z}$, into measurements, $Z$, which can then be passed to the back-end.

The measurements generated by the front-end are separated into three disjoint sets:

- Odometry, $Z^O$, which are measurements relating two consecutive poses of the same device. These measurements are typically assumed to be outlier-free. This is due to the fact that they are derived, at least partially, from sensors which do not require data associations, like wheel odometry or an IMU (Inertial Measurement Unit). Thus, they are much less susceptible to outliers, compared to the other measurements.

- Intra-agent loop closures, $Z^{IA}$, which are measurements relating two non-consecutive poses of the same device.

- Inter-agent loop closures, $Z^{IE}$, which are measurements relating two poses of different devices.

The back-end is tasked with solving an optimisation problem very similar to (2.1), but it is the measurements coming from the front-end that are used as input, instead of the sensor data,

$$ (\mathcal{X}^*, \mathcal{M}^*) = \arg\max_{\mathcal{X}, \mathcal{M}} p(\mathcal{X}, \mathcal{M}|Z) . $$

(2.2)

It is not necessary, and in fact, it is not common, that the pipeline is purely sequential. One way this happens is the back-end can supply the current estimate to the front-end, enabling better measurement processing. A typical functional diagram of the SLAM pipeline is shown in Figure 2.1.

---

**Figure 2.1**: Functional diagram of a SLAM pipeline.
SLAM Back-end

The focus in our work is on the back-end of the SLAM problem. Several methods have been developed for solving the back-end optimisation problem in (2.2) and they can be divided into two categories: filtering and smoothing.

Filtering approaches to SLAM attempt to estimate only the most recent device pose, as well as the map. For real-time localisation applications, this might appear to be an efficient approach since it only cares about the last pose, which is why these approaches were initially the most popular for SLAM. However, it was shown that the matrices involved become necessarily dense [17] and it is not possible to completely correct errors in previous bad estimates. This leads to reduced efficiency over time and poor accuracy, which resulted in filtering gradually becoming less popular.

Smoothing approaches solve the problem in (2.2) efficiently, by exploiting the sparsity of the matrices involved when the full posterior is considered [17]. The main advantage of smoothing approaches over filtering is that by optimising over the full posterior, future measurements can help correct past errors, which is not possible to the same extent in filtering. Due to this efficiency, as well as the fact that they provide full trajectory estimates, most modern approaches use smoothing instead of filtering.

Smoothing exploits an alternate formulation of (2.2),

\[(X^*, M^*) = \arg \max_{X, M} p(X, M | Z) = \arg \max_{X, M} p(Z | X, M) p(X, M). \tag{2.3}\]

Since the measurements are independent, given the trajectory and the map, and typically there is only a prior on the first device pose, the formulation further simplifies to

\[(X^*, M^*) = \arg \max_{X, M} p(X_0) \prod_{z_i \in Z} p(z_i | X_i, M_i), \tag{2.4}\]

where \(X_i \subset X\) and \(M_i \subset M\) are the subsets of the trajectory and map involved in the \(i\)-th measurement, \(z_i\), and \(X_0\) is the first device pose. The measurements, \(z_i\), are assumed to be a function of the trajectory and the map, corrupted by Gaussian noise,

\[z_i = h_i(X_i, M_i) + w_i, \quad w_i \sim N(0, \Sigma_{z_i}). \tag{2.5}\]
Furthermore, the prior is assumed to also be Gaussian. Thus, disregarding constants, the probability density functions can be made explicit

$$(X^*, M^*) = \arg\max_{X, M} \exp\left(-\frac{1}{2} ||X_0 - \overline{X}_0||^2_{\Sigma X_0}\right) \prod_{z_i \in Z} \exp\left(-\frac{1}{2} ||h_i(X_i, M_i) - z_i||^2_{\Sigma z_i}\right),$$

(2.6)

where $||.||^2_{\Sigma}$ is the squared Mahalanobis distance and $\overline{X}_0$ is the expected value of the prior on the initial pose. Finally, by taking the negative log-likelihood of the cost function, the exponentials cancel out,

$$(X^*, M^*) = \arg\min_{X, M} ||X_0 - \overline{X}_0||^2_{\Sigma X_0} + \sum_{z_i \in Z} ||h_i(X_i, M_i) - z_i||_2^2.$$ 

(2.7)

Thus, under the smoothing approach SLAM becomes a Non-Linear Least Squares problem.

Smoothing can also be seen as the optimisation over a factor graph [18], where the variable nodes represent the poses or map being estimated and the factors nodes represent the measurements relating those variables. An example of such a factor graph is shown in Figure 2.2, where the large circles are the variable nodes of the graph while the smaller circles are the factors nodes.

![Figure 2.2: Example of a factor graph in a SLAM problem. Black factor nodes are odometry and the prior, while the orange factor node is an intra-agent loop closure edge.](image-url)

This representation highlights the inherent sparsity of the full SLAM posterior, since it is clear that the degree of each node, the number of edges connected to it, is very small. This motivates why smoothing can be made efficient, since (2.7) can be solved with, e.g., gradient-based sparse solvers.
However, although smoothing exploits the sparsity of the SLAM problem, depending on the map representation and the corresponding number of nodes in the factor graph, the complexity might still be intractable.

In Pose Graph SLAM, the back-end relies on the front-end being able to generate measurements which only relate poses to each other, such that
\[ z_i = h_i(X_i) + w_i, \quad w_i \sim \mathcal{N}(0, \Sigma_{z_i}). \] (2.8)

Thus, it can avoid building an explicit map while optimising the poses and then use the optimised poses to build the map \textit{a posteriori}, at the cost of a denser pose graph [18]. This is motivated by noticing that (2.2) can be factorised as
\[ p(\mathcal{X}, \mathcal{M}|Z) = p(\mathcal{M}|\mathcal{X}, Z)p(\mathcal{X}|Z). \] (2.9)

If the trajectory is known, the problem of estimating the map is known to become simpler when compared to the joint optimisation problem,
\[ \mathcal{M}^* = \arg\max_{\mathcal{M}} p(\mathcal{M}|\mathcal{X}, Z). \] (2.10)

Thus, Pose Graph SLAM consists of the following approximation of (2.2)
\[ \mathcal{X}^* = \arg\max_{\mathcal{X}} p(\mathcal{X}|Z) \quad \text{ (2.11)} \]
\[ \mathcal{M}^* = \arg\max_{\mathcal{M}} p(\mathcal{M}|\mathcal{X}^*, Z), \quad \text{ (2.12)} \]

where the trajectory optimisation problem can still be formulated as the smoothing problem in (2.7). In this work, we are not concerned with the optimisation of the map, so the output of our solution is only the trajectory of the devices. Thus, the back-end optimisation problem in this work is formulated as the Pose Graph SLAM problem and the equivalent factor graph consists only of pose nodes. The smoothing problem in Pose Graph SLAM is also called Pose Graph Optimisation (PGO).

### 2.2 Collaborative SLAM

Collaborative SLAM is taken into consideration when several devices are in the same environment. In such a scenario, if all of the devices need to localize and map their environment, there are two options:

- Every device runs SLAM by itself, obtaining an estimate of its trajectory
and its own independent estimate of the environment map.

• The devices collaboratively solve the SLAM problem, obtaining an estimate of their trajectory and a shared estimate of the environment map.

In case the first option is chosen, collaboration is still possible after the algorithms are run if the individually obtained maps are merged, which is called Collaborative Mapping [19]. However, it is the second option which is called C-SLAM.

In C-SLAM, the trajectory set, $\mathcal{X}$, contains each of the $N$ devices’ trajectories, $\mathcal{X}^j = \{X_i^j\}_{i=0}^P$, i.e., $\mathcal{X} = \bigcup_{j=1}^N \mathcal{X}^j$. Extending (2.3) to the collaborative scenario, under the same assumption of measurement independence and priors only applying to the first pose of each device,

$$\left(\mathcal{X}^*, \mathcal{M}^*\right) = \arg\max_{\mathcal{X}, \mathcal{M}} p(X_0^1) \cdots p(X_0^N) \prod_{z_i \in Z} p(z_i | \mathcal{X}_i, \mathcal{M}_i), \quad (2.13)$$

where $\mathcal{X}_i$ is the subset of device poses involved in measurement $i$, which may or may not come from multiple devices, i.e., $\mathcal{X}_i \subset \mathcal{X}$.

This formulation makes apparent the problem with the prior on the initial device poses. In a single-device scenario, even if there is no absolute prior for the first device pose, e.g., an initial GPS measurement, an arbitrary one can be chosen to define a reference frame for the estimated trajectory and map.

In the collaborative case, providing a prior on the devices’ initial poses, even if it is not absolute, requires knowing their initial relative poses. When these relative poses are unknown, the only available prior for the initial poses is a uniform prior on all but one of the devices, anchoring the map to a reference frame. Thus, the C-SLAM back-end problem is often reduced to

$$\left(\mathcal{X}^*, \mathcal{M}^*\right) = \arg\max_{\mathcal{X}, \mathcal{M}} p(X_0^1) \prod_{z_i \in Z} p(z_i | \mathcal{X}_i, \mathcal{M}_i). \quad (2.14)$$

Given that the C-SLAM problem typically involves several physical devices, the architectures of its solutions can be of different natures. Existing solutions have commonly been divided into two classes, centralized and decentralized, with both allowing distributed variants [4]. It is useful to consider these distinctions since each architecture has its own set of advantages and drawbacks.

Centralized approaches benefit from being able to always use all available information, achieving better estimates but demanding very large
communication bandwidth and processing power, thus scaling poorly with the number of agents [4]. Non-distributed implementations of these solutions have the increased benefit of allowing agents to have limited computing power [5] since much of the expensive computations can be offloaded to a centralized unit, like a server.

Distributed decentralized approaches show better scaling with the number of agents and have more economic communication schemes, with the drawback of not having all the information available, which can lead to worse results, requiring agents with high computational power and facing synchronization issues [4, 9].

Although nothing prohibits the development of distributed centralized solutions or non-distributed decentralized solutions, these are not common and, to the best of the author’s knowledge, none of the state-of-the-art frameworks is implemented in such a way. This is likely due to the fact that these kinds of solutions would have possibly contradicting guiding principles.

For the distributed centralized case, since centralized means the problem gets solved in a central computing unit and distributed means the problem gets separated into individual parts that can be solved in separate units, there is, as of now, seemingly not much interest. However, it could be the case it changes in the future since the necessity for scalability in the centralized approach might push these solutions to parallelisation over several computers, while still conceptually functioning as one centralized unit. It should be noted that this is already done in almost every computer, given that most processors are multi-core. The difference lies in the fact that in this case, the distribution of the computations is completely agnostic to the problem and so it is not considered a distributed C-SLAM implementation.

For the non-distributed decentralized case, an implementation of this would mean every computing unit would solve the same global problem, creating a lot of redundancy and wasting computational resources, which is probably why it has not seen interest.

Given the increased susceptibility of distributed algorithms to malicious actors [20], the increasing capability of mobile communication networks, like 5G and 6G, and the social push towards device miniaturisation, reducing the available onboard computational resources, there are several incentives to investigate centralized C-SLAM solutions.
State of the Art

The state-of-the-art in distributed decentralized C-SLAM is Kimera-Multi [9], where Tian et al. present a robust, fully distributed and decentralized approach to C-SLAM. They improve over previous solutions in this domain by introducing a robust initialization of inter-agent frame transforms, based on [12], and a fully distributed pose graph optimiser. They show that their novel robust frame initialization outperforms existing robust solutions and that their distributed optimiser has comparable performance to non-distributed solvers, meanwhile operating under realistic communication constraints. However, the method implemented relies on batch PGO, making it unsuited for real-time operation in resource-constrained devices, where incremental approaches are necessary.

In centralised C-SLAM, COVINS [11] proposes a system architecture for Visual-Inertial based C-SLAM in which agents perform local SLAM through a Visual-Inertial Odometry (VIO) solution. At the same time, this information is communicated to a server, which performs inter-agent loop closures and global map optimisation. In contrast to previous approaches [19], communication is also established from the server to the agents, since after global optimisation the server informs the agents of the most recent global pose estimate, allowing them to correct their odometry drift. Furthermore, the option to deploy the server to a cloud platform is analysed, validating the scalability of the computing power necessary for a centralized approach. However, although claiming to employ a robust cost function, the optimisation method for this cost function is not discussed. Given the computational hardness of robust estimation [7], it is crucial to describe which heuristic is being used to solve the problem and that missing information makes the results impossible to reproduce.

LAMP2.0 [10] proposes a robust, centralised Collaborative Mapping solution. While not strictly C-SLAM, since the central computing unit does not give information back to the devices which prevent them from localising in the global map, it proposes a complete robust architecture for the front-end and back-end. For the back-end it uses Pairwise Consistency Maximisation (PCM) to pre-filter loop closure measurements in an attempt to reduce the number of outliers that get passed to the main optimiser. However, since it uses Graduated Non-Convexity (GNC) for the main optimiser like Kimera-Multi, it is unsuitable for real-time operation.

Both PCM and GNC, which are very popular in robust C-SLAM solutions, are further introduced in Section 2.4.
2.3 Server-Assisted SLAM

SLAM’s computational performance has been the focus of a lot of the research on the topic. However, in order to obtain acceptable estimation accuracy, SLAM solutions are still computationally demanding. Thus, in order to enable resource-constrained devices to locate themselves and map their environment, some approaches propose delegating most of the computations to a server, while having the device solve a simplified version of the problem.

The approach proposed in [5] introduces an architecture like the one shown in Figure 2.3.

According to this architecture, the server is collecting all measurements from the agent and solving the problem in (2.11). Then, according to some method, a subset of the pose graph’s nodes is chosen, \( X_d \subset X \). A smaller pose graph is then created containing only those nodes, \( X_d \), the measurements which relate them and some summarized information from the rest of the pose graph. This reduced pose graph is obtained by marginalising the nodes which are not meant to be sent to the device. An illustration of the process can be seen in Figure 2.4.

The important part of this method is that, through marginalisation, the reduced graph has fewer nodes than the complete server graph but is still a correct approximation of that graph. This is why the marginalised graph has prior-like edges, which contain the information of the eliminated nodes obtained through marginalisation.

Thus, two important parameters which characterise the functioning of this Server-Assisted SLAM solution are the size of the graph which is sent to the device, \( N_d \), and at what interval the server updates the device’s graph, \( T_u \). Two methods are proposed to choose the nodes which are sent to the device:

- The temporal heuristic, where the \( N_d \) most recent nodes are sent to the
device.

- The spatial heuristic, where the $N_d$ nodes which are estimated to be closest to the estimate of the most recent pose are sent to the device.

Given that the server does not update the agent graph at every step, the most recent agent estimate is always included in the reduced graph. This guarantees that the agent can continue to localise itself at all times.

The interested reader is referred to the original source [5], where some further details on how and when loop-closure edges are generated in this framework are explored, but these mechanisms are not relevant in the context of this work.

State of the Art

As addressed previously, the interest in a complete, centralized, C-SLAM solution is somewhat recent, with a lot of previous research on centralized multi-agent SLAM focusing on Collaborative Mapping [19]. Some work has also been done in Server-Assisted Single-Agent SLAM, which is of interest for the development of Centralized C-SLAM solutions.

Zhang et al. [5] introduce a marginalisation-based approach to generate local maps the server can send back to the agent for single-agent SLAM. The approach proposed has the server keep the agent’s measurements and,
periodically, do global optimisation. From this globally optimised pose graph, a reduced one is generated for the agent by marginalising unwanted variables. The reduced pose graph’s size is determined by the agent’s settings and the variables that are marginalised can be chosen according to different heuristics. They show that their approach allows a resource-constrained device to run a local SLAM solution, achieving good quality real-time localisation, with the help of the server to be able to do loop-closures and keep the entire map history, all this while maintaining acceptable communication bandwidth. Their approach fails to address robustness to outliers and they do not look into extending this framework to the C-SLAM problem.

A graph-theoretical approach for centralized C-SLAM is presented by Bernreiter et al. [21], where the server provides feedback to the agents by sending a summarised version of the global graph back to the agents. Then, using graph spectral analysis, the agents are able to detect discrepancies between their local graphs and the global graph, provided by the server. To fix these discrepancies, the agents add edges to their pose graphs to constrain them to be similar to the global graph. They show that their approach achieves significant performance improvements in the agents’ localisation, with very limited bandwidth. They also show that the increase in the number of edges of each agent’s pose graph is small enough that it doesn’t compromise the sparsity which is key for the efficiency of smoothing-based SLAM algorithms. However, their approach fails to consider robustness to outliers and there is never a reduction or pruning of the agents’ graphs, making this approach unsuitable for long-term operation or resource-constrained devices.

### 2.4 Robust Estimation for SLAM

The problem of estimation consists of estimating a set of parameters, $\theta$, from observed data, $Y = \{y_i\}_{i=1}^M$, where $M$ is the number of observations.

A common estimation framework is Maximum Likelihood Estimation (MLE), under which it is assumed that the data is sampled from a distribution family parameterized by the set of parameters to be estimated. The goal of MLE is determining the $\theta_{\text{MLE}}$ which maximise the probability of the data having been observed,

$$\theta_{\text{MLE}} = \arg\max_{\theta} p(Y|\theta),$$

(2.15)

where $p(Y|\theta)$ is the probability density function of the distribution which
is assumed to have generated the observed data. Under the assumption that measurements are independent, the likelihood function can be written as

\[ p(Y|\theta) = \prod_{i=1}^{M} p(y_i|\theta). \quad (2.16) \]

By taking the negative log-likelihood function instead, and minimizing instead of maximizing, the estimate can be determined by

\[ \theta_{\text{MLE}} = \underset{\theta}{\text{argmin}} \sum_{i=1}^{M} \ln p(y_i|\theta). \quad (2.17) \]

The issue with MLE is that, if some of the observations are not generated by the assumed distribution, i.e., if they’re outliers, it fails to converge to the true parameters [22]. The field of statistics which addresses the issue of estimation when some of the observations are outliers is known as robust estimation.

In one of the seminal works on robust estimation [22], a generalisation of MLE called M-Estimation, for MLE-like, is proposed. In this framework, the robust estimate is given by

\[ \theta^* = \underset{\theta}{\text{argmin}} \sum_{i=1}^{M} \rho(y_i, \theta), \quad (2.18) \]

where the function \( \rho \) obeys some properties (See [22] for details).

For this work, it is particularly relevant to consider the Gaussian case. Because of the log term in (2.17), considering that the observations are generated by a Gaussian distribution with the expected value given by \( h_i(\theta) \) and with covariance \( \Sigma_i \), the estimate is

\[ \theta_{\text{MLE}} = \underset{\theta}{\text{argmin}} \sum_{i=1}^{M} ||h_i(\theta) - y_i||^2_{\Sigma_i}, \quad (2.19) \]

Defining the residuals as \( r_i(y_i, \theta) = h_i(\theta) - y_i \), (2.19) can be written clearly as an M-Estimation problem,

\[ \theta_{\text{MLE}} = \sum_{i=1}^{M} \rho(r_i(y_i, \theta)), \quad \rho(r_i) = ||r_i||^2_{\Sigma_i}, \quad (2.20) \]
By formulating the problem as in (2.20), it can be seen that the function $\rho$ in M-Estimation can be thought of as the weighting function on residuals. For MLE, since this weighting function is quadratic on the norm of the residuals, larger residuals have disproportionately more impact on the estimate than smaller ones, promoting the influence of outliers.

Common robust M-Estimators [23] have weighting functions like Truncated Least Squares (TLS),

$$\rho_{TLS}(r_i) = \begin{cases} ||r_i||_2^2, & ||r_i||_2^2 < \epsilon \\ \epsilon, & \text{otherwise} \end{cases}, \quad (2.21)$$

or the $L1$-norm,

$$\rho_{L1}(r_i) = ||r_i||_1. \quad (2.22)$$

For the TLS kernel, since the threshold is on the Mahalanobis distance, it is often defined as a certain quantile of the appropriate dimension $\chi^2$ distribution. Letting $F_{\chi^2}^{-1}$ be the inverse cumulative distribution function for the $\chi^2$ distribution and $\gamma$ the desired quantile, the threshold is determined by

$$\epsilon = F_{\chi^2}^{-1}(\gamma). \quad (2.23)$$

Thus, from now on we refer to using a certain $\chi^2$ quantile as the threshold for TLS, since the actual threshold, $\epsilon$, is uniquely defined by it. We also interchangeably talk about $\chi^2$ percentiles and quantiles, where $\gamma$ is the quantile and $\gamma \times 100$ is the percentile.

There are several other robust weighting functions, which are also called robust kernels. They all have lower weights for larger residuals with regards to the MLE, which has a quadratic weighting of the residuals. Through this sub-quadratic weighting, robust kernels reduce the impact of outliers in the estimation problem.

Although the smoothing problem in (2.7) is not strictly an MLE formulation, M-Estimation is still applicable. Thus, to perform robust PGO a robust kernel is used. The robust PGO problem is formulated as

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \sum_{i=0}^{M} \rho(r_i(z_i, \mathcal{X})). \quad (2.24)$$

A common approach to robust PGO takes advantage of the assumption that odometry is outlier-free to simplify the problem. That way the problem can
be relaxed into
\[
X^* = \arg\min_X \sum_{z_i \in Z^O} ||r_i(z_i, X)||^2_{Z_i} + \sum_{z_j \in Z^{LC}} \rho(r_j(z_j, X)),
\]
(2.25)

where \(Z^{LC} = Z^{IA} \cup Z^{IE}\). This way the robust kernel is only applied to loop closures, given that the odometry edges are known to be inliers.

The main problem with M-Estimation is that it has been shown to be NP-hard [8]. Thus, for real-world applications where the number of observations and dimension of the parameters can exceed hundreds, exact solutions are intractable. Furthermore, obtaining approximately exact solutions, in the sense that guarantees are given about the approximate solution being in some bounded neighbourhood of the exact one, is also NP-hard. This means that robust estimation requires the use of heuristics, that in order to be fast cannot provide any guarantees on the solutions obtained.

**State of the Art**

A lot of work has been done on the robustness of the SLAM front-end. The state of the art in Visual SLAM, ORB-SLAM3 [6], significantly improves over previous solutions when it comes to the robustness of the data association problem, which is where outliers that are passed to the back-end usually come from. However, given the interleaving of the SLAM front-end and back-end the necessity for improving the back-end’s robustness becomes clear, given the codependence between both parts of the SLAM pipeline.

With the convergence of the research community towards the back-end being formulated as a MAP estimation problem [7], its robust version is the same problem as that of robust estimation. Several algorithms have been presented to attempt to solve the robust estimation in a computationally tractable way, mostly centred around different robust kernels.

Chebrolu *et al.* [24] expand on a parameterised robust kernel, which for different parameter values assumes several well-known robust kernels, and simultaneously optimises over the pose variables and the kernel parameter. By doing this, they achieve better performance than the use of a single kernel since this adaptation means the kernel choice is adequate for the data, which previously inhibited robust kernels from showing reliable performance. Ramezani *et al.* improve on this idea for SLAM with AEROS [25], by choosing an optimisation scheme which better suits the GraphSLAM problem.

Sünderhauf and Protzel formulate the robust PGO by associating a switching variable with each edge in the graph [26], simultaneously optimising
over the poses and these switching variables in order to disable outlier edges. Agarwal et al. expand upon this work with Dynamic Covariance Scaling (DCS) [13], by showing that in some cases the problem can be restated as an Iteratively Reweighted Least Squares (IRLS) problem over only the pose variables. They significantly improve computational performance by reducing the number of optimisation variables, while getting nearly identical estimation performance.

These previously discussed algorithms are all local, in the sense that they require an initial estimate for their optimisation process. Additionally, the solution to the previous iteration is usually used as the initial guess for the next one. This means that when new data is made available that changes the optimal solution, these local solvers can stay trapped in the previous minima, which is now only local.

To address this issue, Yang et al. present GNC [12], an algorithm which attempts to find a global solution for the usual robust optimisation problem. This is done by defining a sequence of kernels which smoothly transition from convex to the desired kernel. The convex problems are solved first since they are easier and have a single minimum. The solution to these problems is then used as an initial guess to the next problem in the sequence, eventually reaching the original problem. It is important to remark that global here means that an initial guess is not required, in contrast with local methods, and not that the global minimum of the problem is guaranteed to be found, since that is not possible with a polynomial-time algorithm [8]. The main disadvantage of GNC is that it is a batch algorithm, and thus not applicable to real-time SLAM.

McGann et al. [27] expand GNC into an incremental version, suitable for integration with existing incremental PGO solvers. Their results show that the incremental version of GNC does not significantly degrade performance while making it real-time compatible, which it previously was not. They do not look into the extension of this robust incremental algorithm to a C-SLAM context.

PCM [28] attempts to address the robust SLAM problem by pre-filtering loop closure measurements before giving them to the PGO solver. This is done by evaluating consistency between pairs of loop closure measurements, together with the odometry measurements which link the involved poses. A graph is constructed where nodes are loop closure measurements and edges connecting two nodes signal that those measurements are consistent with each other. The set of inlier loop closures is determined by finding the maximum clique in this graph, that is, the largest set of nodes which are all connected with each other or, equivalently, the largest set of loop closures which are all mutually consistent. Due to the computational hardness of finding the
maximum clique of a graph, which is NP-hard, PCM also proposes a heuristic version for finding the maximum clique. While the method proposed by the original authors is a batch method, an extension that allows the method to be run in an incremental fashion was proposed in Kimera-Multi [9].

Work has also been done on trying to tackle and tame the computational complexity of robust SLAM. Antonante et al. [8] discuss the computational hardness of the robust estimation problem, emphasizing the computational infeasibility of obtaining exact, but also approximately exact, solutions to the problem. The necessity for tuning in existing algorithms is discussed and novel algorithms are presented, being described as minimally tuned. They show the tuning required for these algorithms is only dependent on the application subject and not on the specific datasets, as previous algorithms [12]. However, the applicability of these robust estimation algorithms to real-time SLAM is compromised due to the necessary computational time.

In [29], Yang and Carlone present a novel framework for optimality guarantees in robust estimation. They side-step the computational hardness of the problem by relying on either heuristics, like GNC [12], or convex relaxations of the exact problem to get an approximate solution. Then, they compute the suboptimality bound to be able to certify whether the solution obtained is optimal or not. While the demonstrated implementations are not suitable for real-time use, since they are still computationally very demanding, they open the door for new approaches to robust estimation. Being able to certify obtained solutions pushes the boundary on the guarantees of safety that autonomous devices can provide, which are highly dependent on the robustness of algorithms like SLAM.

2.5 Summary

A lot of research has been done on improving SLAM estimation quality, the issues associated with augmenting SLAM into C-SLAM, and also robust estimation, achieving impressive results. However, there is a gap in the literature concerning robust, centralized C-SLAM, as well as centralized C-SLAM, catered for resource-constrained agents.

This work bridges that gap by studying the impact that outliers have on the presented robust methods when applied to Server-Assisted C-SLAM and presenting a novel solution, based on state of the art methods.
Chapter 3

Baseline & Evaluation Framework

In this chapter, we introduce the baseline Server-Assisted C-SLAM algorithm, which is an extension of [5]. Then, we discuss the existing metrics for evaluating the accuracy of Single-Device SLAM and their extension to incremental C-SLAM. Finally, we describe the data generation process, with special emphasis on how outliers are generated in synthetic data.

3.1 Baseline

The baseline is an extension of the Server-Assisted SLAM algorithm described in Section 2.3. The extension is based on the observation that there is nothing stopping the server from having nodes from more than one device in its graph.

The main contribution is the extension of the existing methods for choosing which nodes to send to the device. Besides deciding whether to select nodes which are temporally or spatially closest, it is now also important to choose whether the reduced graph should contain only nodes from the device the graph is being sent to or from all devices. Thus, there are now four heuristics to choose which nodes to incorporate in the graph to send to the device:

- The temporal heuristic with only the device’s nodes, where the $N_d$ most recent device nodes are sent.
- The temporal heuristic with all devices’ nodes, where the $N_d$ most recent nodes, from all devices, are sent.
- The spatial heuristic with only the device’s nodes, where the $N_d$ device
nodes estimated to be closest to the device’s most recent pose estimate are sent.

- The spatial heuristic with all devices’ nodes, where the $N_d$ nodes, from all devices, estimated to be closest to the device’s most recent pose estimate are sent.

This algorithm expects a multi-device architecture such as the one depicted in Figure 3.1. The implementations used for this work merely simulate the existence of multiple devices through functional separation, even though all software is running on the same computer. The implementation and testing of these solutions on a true multi-device setup is left as future work.

![Figure 3.1: Multi-device architecture for the Server-Assisted C-SLAM algorithm.](image)

In our work, only the back-end is analysed. The source for the Server-Assisted SLAM algorithm presented in Section 2.3 [5] tackles some issues concerning also how the front-ends are connected. For simplicity and to keep the focus on the back-end, the functioning of the front-ends is assumed to be such that:

- Each device has access, at every time step, to its own odometry.

- Whether each device has access to loop closures which affect them, both intra- and inter-agent, as they happen is a controllable parameter of the algorithm. If they do not have access to the loop closures in the time step they happen, they still receive them from the server when it updates.

- The server has access, at every time step, to every device’s odometry, as well as all intra- and inter-agent loop closures.

As depicted in Figure 3.1, every device and the server have a separate back-end. Every back-end implementation in the baseline uses the same PGO solver, Incremental Smoothing and Mapping 2 (iSAM2) [30]. The implementation
of the solver is provided by GTSAM [31]. However, the way the solver is used differs between the server and the devices. In Algorithm 1 the baseline implementation for the devices’ back-ends is shown.

Algorithm 1 Baseline Device Back-end.

\[
\begin{align*}
\text{PoseGraph} & \leftarrow \emptyset \\
\hat{\mathcal{X}} & \leftarrow \emptyset \\
\textbf{while} & \text{ device is running do} \\
& Z_{\text{new}} \leftarrow \text{GetNewMeasurements}() \\
& \text{CommunicateMeasurementsToServer}(Z_{\text{new}}) \\
& \textbf{if} \text{ NewServerPoseGraph()} \textbf{ then} \\
& \quad \text{PoseGraph} \leftarrow \text{GetServerPoseGraph()} \\
& \textbf{else} \\
& \quad \text{PoseGraph} \leftarrow \text{UpdatePoseGraph}(\text{PoseGraph}, Z_{\text{new}}) \\
& \hat{\mathcal{X}} \leftarrow \text{PGOSolver}(\text{PoseGraph})
\end{align*}
\]

The server back-end is intended to run at a lower frequency compared to the devices, but always using all the information available. Then, after performing optimisation with all the information it generates reduced pose graphs for each device. This enables the devices to keep localising in real-time, with high accuracy. The baseline implementation for the server’s back-end is shown in Algorithm 2.

The server can be configured to generate reduced graphs with all devices’ nodes, not just the nodes of the device the graph is being sent to. In this case, only nodes from devices which are actively collaborating with the target device can have their nodes included. Active collaboration means that the devices’ pose graphs are connected through inter-agent loop closures. Until two devices begin actively collaborating, in practice they are not solving the same C-SLAM problem.

For two devices to be actively collaborating, they do not need to have a direct inter-agent loop closure connecting them. Considering three example devices: A, B and C. If A is connected to B and B to C, then A and C are actively collaborating, even if they are not directly connected.

iSAM2 was chosen as the PGO solver since it is incremental, making it computationally efficient during the estimation process, and also efficient for performing marginalisation, which is necessary on the server back-end.

The parameters of this, non-robust, baseline implementation are fixed, having been determined in previous tests. The values used throughout our work are given in Table 3.1.
Algorithm 2 Baseline Server Back-end.

GlobalPoseGraph ← ∅
for each device d do
    Ḫ^d ← ∅
    Ḫ ← { Ḫ^1, . . . , Ḫ^N }  
    t ← 0
while devices are running do
    for each device d do
        Z^d_{new} ← GetMeasurementsFromDevice(d)
        Z_{new} ← Z^1_{new} ∪ . . . ∪ Z^N_{new}  ▷ Get device measurements

GlobalPoseGraph ← UpdatePoseGraph(GlobalPoseGraph, Z_{new})
if IsServerUpdateTime(t) then
    Ḫ ← PGOSolver(GlobalPoseGraph)
    for each device d do
        if ShouldGeneratePoseGraph(d, t) then
            PoseGraph^d ← GeneratePoseGraph(d, GlobalPoseGraph, Ḫ)
            SendPoseGraph(d, PoseGraph^d)
        t ← t + 1

Table 3.1: Configuration parameters for the baseline solution

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Size of the reduced graph (N_d)</td>
<td>300</td>
</tr>
<tr>
<td>Period of the server update window (T_u)</td>
<td>50</td>
</tr>
<tr>
<td>Reduced graph generation heuristic</td>
<td>Temporal</td>
</tr>
<tr>
<td>Only device’s own nodes</td>
<td>Yes</td>
</tr>
<tr>
<td>Devices can do loop closures</td>
<td>Yes</td>
</tr>
</tbody>
</table>

A consequence of this choice of parameters is that since each device’s graph only has its own nodes, devices cannot discover inter-agent loop closures by themselves. Additionally, even though devices can perform intra-agent loop closures with these settings, these are also limited to the nodes available on the device. Therefore, only intra-agent loop closures with the most recent 300 poses can be done in real-time on the device. The remaining loop closures can only be done on the server and are then sent to the device during the server update.
3.2 Evaluation Framework

Several methods exist for evaluating the results of SLAM solutions. However, existing methods either rely on the existence of an explicit map, making them unsuitable for evaluating a Pose Graph SLAM solution, or are not suitable for a collaborative setting. Additionally, most of the works concerned with evaluating localisation accuracy are not using incremental solvers [9, 25].

In this section, a method for evaluating localisation and mapping accuracy in an incremental Collaborative Pose Graph SLAM framework is proposed. This method is based on the Absolute Trajectory Error (ATE) [32], which quantifies the error between an estimated trajectory, $\hat{X}$, and a reference trajectory, $\bar{X}$. In Figure 3.2 an example pair of reference and estimated trajectories for a two-device scenario are shown.

![Reference and estimated trajectories](image)

**Figure 3.2: Reference and estimated trajectories for a two-device scenario. The estimated trajectories’ poses are in lighter shades of the same colors, in order to distinguish them from the reference trajectories.**

**Absolute Trajectory Error (ATE)**

In order to define our evaluation metrics, we begin by introducing how the ATE is calculated.

As discussed in Section 2.1, when there is no absolute initial prior for single-device SLAM, an arbitrary one can be used to anchor the estimate to a single frame. The consequence is that the reference and estimated trajectories are not necessarily using the same frame.
To address this, the first step to calculate the ATE is to align both trajectories. This means that a transform, $\mathcal{T} \in SE(3)$, is found that minimises the Mean Squared Error (MSE) between the position components of both trajectories,

$$\mathcal{T} = \arg\min_{F \in SE(3)} \sum_{i=0}^{T} ||\bar{X}_{i,p} - F \oplus \hat{X}_{i,p}||^2,$$

(3.1)

where $\bar{X}_{i,p}$ is the position component of the $i$-th pose of the reference trajectory and $\hat{X}_{i,p}$ the same for the estimated trajectory. In this expression, $F \oplus \hat{X}_{i,p}$ is the application of the rigid transform $F$ to the position $\hat{X}_{i,p}$, rotating and translating it to another frame. The details of how this operation is performed, as well as other $SE(3)$ operations, are described in Appendix B.

After alignment, the ATE is given by the Root Mean Squared Error (RMSE) between the position components of the reference trajectory and the aligned estimated trajectory,

$$\text{ATE} = \sqrt{\frac{1}{T} \sum_{i=0}^{T} ||X_{i,p} - \mathcal{T} \oplus \hat{X}_{i,p}||^2}.$$

(3.2)

**Incremental Collaborative ATE**

We now introduce a set of metrics, based on the ATE, suitable for evaluating the results of an incremental Collaborative Pose Graph SLAM solution. We call this set of metrics the Incremental Collaborative ATE (iCATE).

The extension of the ATE to a collaborative setting requires deciding which sets of devices to use for the alignment of the estimated trajectories to the reference trajectories, as well as which sets of trajectories to evaluate the errors on. The motivation behind these choices is based on the systems which might use the trajectory estimate. For control, it is important for the estimate to be correct in the local frame, even if it might be wrong in the global frame. For planning, it is important that the estimate be correct in the global frame.

To address this, two metrics are defined: the *Individual Error*, which aims to measure the individual trajectory error in the local frame, and the *Collaborative Error*, which aims to measure the collaborative trajectory error in the global frame.

The Individual Error for device $d$ at time $t \leq T$, $\text{IE}_t^d$, is given by aligning
the estimated trajectory, $\hat{X}^d$, with the reference trajectory, $\bar{X}^d$, up to time $t$,

$$T_t^d = \arg\min_{F \in SE(3)} \sum_{i=0}^{t} ||\bar{X}_i^d - F \oplus \hat{X}_i^d||^2, \quad (3.3)$$

and then using that alignment transform to compute the Individual Error,

$$IE_t^d = \frac{1}{t} \sum_{i=0}^{t} ||\bar{X}_i^d - T_t^d \oplus \hat{X}_i^d||^2. \quad (3.4)$$

Thus, the Individual Error is no more than the standard ATE for each device. However, an important difference in the treatment of the Individual Error compared to the standard ATE is that the Individual Error is a time signal, being evaluated for every time instant the device is running. In contrast, the ATE is typically only considered for the final estimate of a device, which is equal to the Individual Error for that device at the final time, $IE_T^d$. In Figure 3.3, an example of the alignment for the Individual Error is shown for the two-device scenario from Figure 3.2.

![Figure 3.3: Trajectory alignment for the Individual Error in a two-device example. Even though the estimated global frame is wrong, since trajectories are aligned individually the Individual Error is still low.](image)

The Collaborative Error, $CE_t^{A_c}$, is defined for each set of actively collaborating devices, as defined in Section 3.1, $A_c$, at each time $t \leq T$. In order to calculate the Collaborative Error, the trajectories in the set of collaborating devices are aligned as a whole,

$$T_t^{A_c} = \arg\min_{F \in SE(3)} \sum_{d \in A_c} \sum_{i=0}^{t} ||\bar{X}_i^d - F \oplus \hat{X}_i^d||^2. \quad (3.5)$$
Then, the Collaborative Error is the RMSE between all the aligned estimated trajectories and their respective reference trajectories,

\[
CE_t^{Ac} = \sqrt{\frac{1}{\#A_c} \sum_{d \in A_c} \frac{1}{t} \sum_{i=0}^{t} \| \bar{X}_d^i - T^d_i \oplus \hat{X}_d^i \|^2},
\]

(3.6)

where \(\#A_c\) is the number of elements in the set \(A_c\).

Thus, the Collaborative Error is similar to the ATE, if all collaborating devices’ trajectories were considered as a single trajectory. Since all the trajectories are taken as one, the Collaborative Error gives information about the accuracy of the estimate of the relative positions between devices and, equivalently, their positions in the global frame. In Figure 3.4, an example of the alignment for the Collaborative Error is shown for the two-device scenario from Figure 3.2.

![Figure 3.4: Trajectory alignment for the Collaborative Error in a two-device example. Since the estimated global frame is wrong, the Collaborative Error is high, even if the Individual Error is low.](image)

Since these errors are time signals, calculated for every device or set of actively collaborating devices, they are summarised in order to make result analysis simpler. The first step to this is to consider the Individual Error at time \(t\), \(IE_t\), as the average of each device’s error. For the Collaborative Error, this average is across sets. To summarise the error information over time, the mean and median are chosen since they allow a good characterisation of the
overall error distribution. Thus, the iCATE is given by the four metrics

\[
\text{mean IE} = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{N} \sum_{i=1}^{N} IE_i^t \quad \text{median IE} = \text{median} \left\{ \frac{1}{N} \sum_{i=1}^{N} IE_i^t \right\} \quad (3.7)
\]

\[
\text{mean CE} = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{N'} \sum_{i=1}^{N'} CE_i^t \quad \text{median CE} = \text{median} \left\{ \frac{1}{N'} \sum_{i=1}^{N'} CE_i^t \right\} \quad (3.8)
\]

where \( N' \) is the number of actively collaborating sets of devices.

Since the goal of this Server-Assisted solution is to allow real-time localisation in resource-constrained devices, computational performance must also be evaluated. For this, the iteration times are measured, for the devices and the server.

### 3.3 Data Generation

The data generation is based on Kollagen [33], a multi-device pose graph generator inspired by the M3500 [34] dataset. Kollagen generates pose graphs by first generating ground truth trajectories in a planar grid world and subsequently perturbing these trajectories by contaminating the odometry and loop closure measurements with Gaussian noise. A detailed description of the functioning of Kollagen is given in Appendix A.

In Figure 3.5, an example ground truth and noisy trajectory are given for a single example device. The noisy trajectory is obtained as the best estimate for the trajectory given the noisy odometry and intra-agent loop closures generated from the ground truth.

After generating every device’s trajectory, the trajectories are all centred around the origin and inter-agent loop closures are generated. This centring is done to promote encounters between devices and consequently promote inter-agent loop closures. In Figure 3.6, the ground truth trajectories for two devices are shown.

After the devices’ trajectories and inter-agent loop closures have been generated, an outlier-free pose graph is available. However, Kollagen doesn’t deal with the generation of outliers. Given that the focus of our work is the robustness of Server-Assisted C-SLAM to outliers, we now look into the generation of synthetic outliers. Since there has already been extensive study of robust Single-Device SLAM [12, 13, 25, 26] and the impact of outlier intra-agent loop closures in this scenario, we focus only on the generation of outlier
In order to describe the implementation of the generation of outlier inter-agent loop closures, some concepts are first introduced. The set of devices is represented by $A = \{1, \ldots, N\}$, where $N$ is the number of devices. The set of device trajectories, $X = \{X^1, \ldots, X^N\}$, where $X^i$ is the trajectory of the $i$-th device, coincides with the sets of variable nodes in the pose graph.

To the best of the author’s knowledge, no work exists which investigates how outliers generated by real front-ends in realistic conditions are distributed. Instead, work into robust PGO focuses on synthetically generated outliers\cite{12, 13, 26}. The commonly used methods to synthetically generate outliers are described in the work of Sünderhauf and Protzel\cite{26}. Different methods are proposed, corresponding to different heuristics on how the front-end might generate outlier measurements:

- **Random** – Randomly choose any two nodes in the pose graph and add a random measurement between them. This method heuristically corresponds to random failures, like sensor failures or communication errors.

- **Local Grouped** – Randomly choose a node. Next, randomly choose a node, but only from those that are within some specified distance of the
node that was initially chosen and add a random measurement between them. Then, for the nodes consecutive to each of the chosen nodes, add a measurement which is consistent with the first random measurement and the odometry of these nodes. Keep doing this until the desired group length is reached. In this scenario, outliers are generated in a way that is heuristically similar to perceptual aliasing, i.e., the front-end confuses one place with another and generates measurements accordingly, until it leaves that place.

Two other modes, Random Grouped and Local, are also proposed but they are not explored in our work.

The Random outliers are interesting since they should be the easiest case, given that different outliers are unlikely to be correlated. Outliers generated in the Local Grouped mode are similar to perceptual aliasing, making them important to study since this is an important mode of failure.

The generation of outlier inter-agent loop closures is done after the outlier-free pose graph has been generated. This allows for both to be saved in order to evaluate the impact of outliers. The outlier generation methods require the agent set, the pose graph nodes, the inlier loop closure measurements and also

Figure 3.6: Example of the ground truth trajectories for two devices. The ground truth trajectories are centred such that their centres of mass are at the origin, promoting encounters between devices. Observing the ground truth trajectory of device 1, which is clearer in Figure 3.5, an overlapping segment can be seen in the lower left area of the grid world.
the desired number of outliers, $M_{\text{outliers}}$.

In Algorithm 3 the implementation for the Random mode is shown. UniformPairDraw draws two elements from a set, without replacement, with uniform probability. UniformDraw draws a single element from a set with uniform probability. UniformDrawContinuous$(a, b)$ draws a real number from a uniform probability distribution with density $\mathcal{U}(a, b)$. LearnScale determines the maximum absolute value of the translational components of all measurements in the set that it receives as input.

Algorithm 3 Outlier generation in Random mode.

**Input:** $A, \mathcal{X}, Z_{\text{inliers}}^{IE}, M_{\text{outliers}}$

**Output:** $Z_{\text{outliers}}^{IE}$

$Z_{\text{outliers}}^{IE} \leftarrow \emptyset$

$x_{\text{max}}, y_{\text{max}} \leftarrow \text{LearnScale}(Z_{\text{inliers}}^{IE})$

**while** $\#Z_{\text{outliers}}^{IE} \neq M_{\text{outliers}}$ **do**

$D_1, D_2 \leftarrow \text{UniformDrawPair}(A)$ \hspace{1cm} \triangleright \text{Choose devices to draw the nodes}$

$N_1 \leftarrow \text{UniformDraw}(\mathcal{X}^{D_1})$

$N_2 \leftarrow \text{UniformDraw}(\mathcal{X}^{D_2})$ \hspace{1cm} \triangleright \text{Draw a node from each device}$

**if** $(D_1, N_1, D_2, N_2, \cdot) \in Z_{\text{outliers}}^{IE} \cup Z_{\text{inliers}}^{IE}$ **then**

**continue** \hspace{1cm} \triangleright \text{If it already exists, try to sample again}$

$x \leftarrow \text{UniformDrawContinuous}(-x_{\text{max}}, x_{\text{max}})$

$y \leftarrow \text{UniformDrawContinuous}(-y_{\text{max}}, y_{\text{max}})$

$\theta \leftarrow \text{UniformDrawContinuous}(0, 2\pi)$

$z^{lc} \leftarrow (x, y, \theta)$

$Z_{\text{outliers}}^{IE} \leftarrow Z_{\text{outliers}}^{IE} \cup \{ (D_1, N_1, D_2, N_2, z^{lc}) \}$

Since $SE(n)$ is infinite, given that the translational component of an element is in $\mathbb{R}^n$, uniform sampling is impossible. To address this issue, we learn the scale of the translational components of the inlier measurements. The translational component of the outlier measurements is then sampled from a uniform distribution with this learned scale, while the orientation component is sampled from $\mathcal{U}(0, 2\pi)$. It is important to have the scale of the outliers be similar to the inliers, otherwise they could just be quickly identified by being much larger or much smaller than the inliers. While these kinds of easier outliers also exist, it is reasonable to assume that they’ve already been dealt with by the front-end.
For the Local Grouped mode, besides the already mentioned required parameters, the maximum distance between nodes, $R$, and the desired group length, $G$, must also be given. In Algorithm 4, the method for generating outlier inter-agent loop closures using the Local Grouped method is shown.

**Algorithm 4** Outlier generation in Local Grouped mode.

**Input:** $A, \mathcal{X}, Z_{\text{inliers}}^{IE}, M_{\text{outliers}}, R, G$

**Output:** $Z_{\text{outliers}}^{IE}$

$Z_{\text{outliers}}^{IE} \leftarrow \emptyset$

$x_{\text{max}}, y_{\text{max}} \leftarrow \text{LearnScale}(Z_{\text{inliers}}^{IE})$

while $\#Z_{\text{outliers}}^{IE} \neq M_{\text{outliers}}$ do

$D_1, N_1 \leftarrow \text{UniformDraw}(\mathcal{X})$  \hspace{1cm} \triangleright \text{Draw a random node}

repeat

$D_2, N_2 \leftarrow \text{UniformDraw}(\mathcal{X} \setminus \mathcal{X}^{A_1})$

until $\text{distance}(N_1, N_2) \leq R$  \hspace{1cm} \triangleright \text{Draw a node which is close enough}

if $(D_1, N_1, D_2, N_2, \cdot) \in Z_{\text{outliers}}^{IE} \cup Z_{\text{inliers}}^{IE}$ then

continue  \hspace{1cm} \triangleright \text{If it already exists, try to sample again}

$x \leftarrow \text{UniformDrawContinuous}(-x_{\text{max}}, x_{\text{max}})$

$y \leftarrow \text{UniformDrawContinuous}(-y_{\text{max}}, y_{\text{max}})$

$\theta \leftarrow \text{UniformDrawContinuous}(0, 2\pi)$

$z^k \leftarrow (x, y, \theta)$

$Z_{\text{outliers}}^{IE} \leftarrow Z_{\text{outliers}}^{IE} \cup \{(A_1, N_1, A_2, N_2, z^k)\}$

$i \leftarrow 1$

while $i \leq G$ and $\#Z_{\text{outliers}}^{IE} \neq N_{\text{outliers}}$ do

$N_1 \leftarrow N_1 + 1$

$N_2 \leftarrow N_2 + 1$  \hspace{1cm} \triangleright \text{Select the next nodes}

$z^k \leftarrow (\otimes z_{D_1,N_1}^{O}) \oplus z^k \oplus z_{D_2,N_2}^{O}$  \hspace{1cm} \triangleright \text{Compose odometry with measurement}

$Z_{\text{outliers}}^{IE} \leftarrow Z_{\text{outliers}}^{IE} \cup \{(D_1, N_1, D_2, N_2, z^k)\}$

$i \leftarrow i + 1$

After generating the outlier loop closures, the measurement set is composed of the odometry measurements, the intra-agent loop closures and the inter-agent loop closures, $Z = Z^{O} \cup Z_{\text{IA}}^{IE} \cup Z_{\text{IE}}^{IE}$. The inter-agent loop
closures are composed of the inliers and outliers, \( Z^{IE}_{inliers} \cup Z^{IE}_{outliers} \). Thus, the pose graph is completely generated and diverse data can be generated. This allows extensive validation of the baseline and the developed solutions, furthering the statistical significance of the results obtained in this work.

### 3.4 Summary

In this chapter, the implementation details for the baseline solution are given as well as the method for result evaluation. Together with the data generation method, this enables the reproducibility of all the results presented in our work.

This sets the stage for what is done in Chapter 4, where the impact of outliers on the baseline solution described in this chapter is evaluated. Additionally, the impact of outliers on naive robust solutions, resulting from the direct application of the methods described in Section 2.4 to the baseline solution, is also studied.

The study of the impact of outliers on these different solutions is used as motivation for the development of a novel solution, which is presented in Chapter 5.
Chapter 4

Outlier Impact on Server-Assisted C-SLAM

In this chapter, the effect of outliers on the estimation accuracy and computational performance of Server-Assisted C-SLAM is studied. Besides the baseline implementation discussed in Chapter 3, two naive robust implementations are also introduced. These naive implementations consist of the direct application of two robust PGO methods, DCS and GNC, to the Server-Assisted C-SLAM architecture.

It is shown how the direct application of DCS and GNC, which are designed for single device SLAM, do not yield sufficiently good results in the Server-Assisted C-SLAM case. They are either too computationally expensive or too inaccurate unless priors on all devices’ initial relative poses are available.

The importance of priors in every device’s initial relative pose is discussed. An intuitive explanation is given for their importance and it is then validated with an experiment.

This is the first of two chapters presenting the contributions of this thesis. In the following chapter, the knowledge of the importance of the priors on each device’s initial relative pose is used to develop a novel solution based on robustly estimating these priors from the inter-agent loop closures.

4.1 Robust C-SLAM Implementation

Of the robust methods discussed in Section 2.4, we restrict ourselves to a subset of them for practical reasons. Given that not all of them have available open-source implementations, DCS and GNC were chosen because they are already
implemented in the PGO library used in the baseline implementation, GTSAM [31]. This choice is also motivated by the fact that these two methods are quite different from each other, presenting a different set of trade-offs between accuracy and computational performance.

DCS is a local solver, in the sense that it requires an initial guess in order to start attempting to solve the PGO problem. It also is very simple to integrate into iSAM2, allowing it to run in an incremental fashion.

GNC is a global solver, in the sense that it does not require an initial guess. However, as discussed previously, due to the NP-hard nature of the robust estimation problem, it is not global in the sense that it is guaranteed to return the global optimal solution. Unlike DCS, the original version of GNC [12], cannot be easily integrated into iSAM2 or other incremental algorithms. That means that GNC has to run in batch mode, completely reoptimising the whole pose graph at each iteration. riSAM [27] proposes to address this issue, but an implementation is not available at the time of writing and thus it is not considered.

DCS and GNC, and in fact all M-Estimators as formulated in (2.25), are equivalent to an optimisation problem over the residuals and weights on the residuals, due to the Black-Rangaraj duality [12]. Typically, this problem is solved through an alternating optimisation scheme over the residuals and the weights. The two problems in the alternating optimisation are a weighted least squares problem on the residuals,

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \sum_{z_i \in Z^O} ||r_i(z_i, \mathcal{X})||_{\mathcal{X}}^2 + \sum_{z_j \in Z^L} w_j ||r_j(z_j, \mathcal{X})||_{\mathcal{X}}^2,$$  

and a problem that is related to the choice of the robust cost function,

$$w^* = \arg\min_{w \in [0,1]} \sum_{z_j \in Z^L} w_j ||r_j(z_j, \mathcal{X})||_{\mathcal{X}}^2 + \Phi_\rho(w_j),$$  

where $w = w_j$ and the function $\Phi_\rho$ is related to the robust kernel, $\rho$. This function is what prevents the optimal solution to be just setting the weights to zero, since that would make the cost zero as well. In this way, $\Phi_\rho$ can be thought of as being related to some prior on the weight values.

Given that the weighted terms are multiplicative in the squared Mahalanobis distance, this weighting is equivalent to rescaling the covariance of the measurements,

$$w ||r(z, \mathcal{X})||_{\Sigma}^2 = r^T(z, \mathcal{X})(w \Sigma^{-1})r(z, \mathcal{X}) = ||r(z, \mathcal{X})||_{w^{-1}\Sigma}^2.$$  

(4.3)
For low weights, the inverse of the covariance gets smaller, therefore the covariance gets larger and the measurement has a lower impact in the estimation.

This weighting allows the server to compute a robust solution and communicate it to the devices by providing them with a weighted graph, instead of the simple unweighted graph used in the baseline. This means that for the server, the implementation described in Algorithm 2 changes only in the PGO solver used and how the reduced graph is generated.

For DCS, the PGO solver changes to a modified version of iSAM2 that uses DCS. Then, the robust estimation alters the measurements by rescaling their covariance. This means the pose graph remains unchanged in terms of its implementation since the weighting was done through the covariance matrices. The reduced pose graph generation remains the same and the devices do not need to change anything in order to accept this weighted graph. DCS is characterised by a single parameter, $\Phi$, which controls how likely DCS is to attribute a low weight to a measurement (not to be confused with the outlier process, $\Phi_\rho$). Thus, $\Phi$ can be thought of as a prior on the likelihood that measurements are outliers, with a small value for $\Phi$ meaning that the prior belief is that measurements are mostly inliers and a large value meaning the opposite. Through a parameter sweep, it was found that the value that yielded the best compromise between accuracy and reliability was $\Phi = 5$, which is neither a small nor large value (DCS has been tested with values for $\Phi$ between 0.1 and 100 [13]), leading to a close to uniform belief on whether measurements are inliers or outliers.

For GNC, the PGO solver changes to the batch GNC solver. With GNC, the pose graph is not directly weighted through the covariance matrices, with a weight vector being provided instead. However, these weights can be included in the covariances using (4.3), such that a weighted pose graph is produced without changing its implementation. This means that GNC can also produce a pose graph that the device can receive without changing its implementation. Although GNC supports different robust kernels, both preliminary experiments and the original source [12] suggest that Truncated Least Squares (TLS) achieves the best results. The threshold in (2.21) is set to the 99% quantile of the $\chi^2$ distribution with the same number of dimensions as the measurement.

Since DCS proves to be computationally inexpensive, all robust solutions run DCS in the devices as well. This means that the implementation in Algorithm 1 also changes, since its PGO solver becomes the modified version of iSAM2 with DCS. This is done regardless of whether the server is running
DCS or GNC. GNC is computationally too demanding to be run in the devices. From this point on, when we refer to DCS or GNC we are referring to the naive robust solutions just described, which result from directly applying these methods to the Server-Assisted C-SLAM architecture.

### 4.2 Robust SLAM vs Robust C-SLAM

For both Single-Device and Collaborative SLAM odometry is assumed to be outlier-free. This means that for every intra-agent loop closure the alternate source of truth formed by the odometric chain can be used to accept or discard the loop closure as an inlier. This insight is not novel and existing solutions, e.g., PCM [28], have used it to attempt to solve the problem of identifying inlier inter-agent loop closures.

However, for inter-agent loop closures, an outlier-free odometry chain is not enough to identify if an edge is an inlier or an outlier. For that, priors on every device’s initial relative pose are necessary. If priors for the initial relative poses of all devices are available, and they are assumed to be outlier-free, then these priors form an alternate source of truth that allows to accept or discard an inter-agent loop closure.

Consider the first inter-agent loop closure in a two-device scenario. Without priors on each device’s initial relative pose, this loop closure can neither be accepted nor rejected, since it cannot be compared to anything else. Even as more loop closures accumulate, they cannot be compared to a source of truth. The only way to determine which edges are inliers and which are outliers is to find the largest set of mutually agreeing loop closures.

This highlights the two fundamental ways in which the lack of priors in robust C-SLAM manifests: without priors, there is no absolute source of truth and a single loop closure cannot be accepted or rejected by itself.

Thus, the absence of priors on the device’s initial relative pose should make it much harder to solve the robust C-SLAM. It can be expected that as the inter-agent loop closures accumulate, the inlier set can more reliably be detected. This means that it can be expected that the robust solvers have a harder time in the beginning when devices begin to collaborate but there are still not enough loop closures for a reliable estimate. This hardship in the early stages can be troublesome for local solvers since they are susceptible to getting stuck in local minima.

In order to validate or disprove this hypothesis, one of the experiments that follow looks at the impact that the availability of priors has on the estimation accuracy, all else equal.
4.3 Experimental Design & Results

For evaluating the impact of outliers in Server-Assisted C-SLAM solutions, we design three experiments. The first two evaluate how an increasing amount of inter-agent loop closures being outliers impacts the estimation accuracy and the computational performance of the baseline solution, as well as the naive robust implementations introduced in this chapter. This is done for both Random and Local Grouped outliers. The third experiment evaluates the impact that priors on the devices’ initial relative poses have on the robust implementations when there are outliers.

In order to obtain statistically significant results, each experiment parameter combination is run 10 times with datasets generated with different random seeds. For experiments where the only variations on the datasets are the outliers, the odometry and inlier loop closures are the same across experiments. The reference trajectories for evaluation are the ground truth trajectories of each device.

The parameters for the dataset generation which are not related to outliers are the same for all experiments and are shown in Table 4.1.

<table>
<thead>
<tr>
<th>Number of devices ($N$)</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steps before turning ($s$)</td>
<td>4</td>
</tr>
<tr>
<td>Total number of steps ($T$)</td>
<td>4000</td>
</tr>
<tr>
<td>Odometry position standard deviation ($\sigma_l$)</td>
<td>0.023</td>
</tr>
<tr>
<td>Odometry orientation standard deviation ($\sigma_\theta$)</td>
<td>0.0023</td>
</tr>
<tr>
<td>Loop closure position standard deviation</td>
<td>0.023</td>
</tr>
<tr>
<td>Loop closure orientation standard deviation</td>
<td>0.0023</td>
</tr>
</tbody>
</table>

All experiments are run in a system equipped with an Intel Xeon E5-2680@2.40GHz CPU with 128GB of RAM. GTSAM was built in Release mode with multi-threading enabled.

**Experiment 1: Impact of Random outliers**

In order to investigate the susceptibility of the C-SLAM solutions to inter-agent loop closure outliers, the datasets are generated with a varying outlier ratio. The outlier ratio is defined as

$$\text{Outlier Ratio} \% = \frac{N_{\text{outliers}}}{N_{\text{outliers}} + N_{\text{inliers}}} \times 100.$$  (4.4)
Thus, for an outlier-free dataset with a given number of inlier loop closures, $N_{\text{inliers}}$, the number of outliers to be generated, $N_{\text{outliers}}$, is determined from the outlier ratio.

For a varying outlier ratio, the number of inlier loop closures is the same. Consequently, increasing the outlier ratio also increases the total number of inter-agent loop closure edges. This means that the results can also be influenced by the number of measurements and not just the fact that they are inliers or outliers. However, this is deliberate, since it mimics, e.g., reduced thresholding in the data association module in the front-end or a higher failure rate. This would have the effect of creating a higher number of measurements and a higher proportion of outliers, which is what happens in our experiments.

In this experiment, the outlier ratio varies between 0% and 50%, in 10% intervals, and the outlier generation mode is the Random mode. There are no priors on the initial relative poses of the devices.

## Results

In Figure 4.1, the mean and median Individual Error for the Random outliers are shown. The mean and median Collaborative Error are shown in Figure 4.2.

![Figure 4.1: Individual Error metrics for Baseline, DCS and GNC with varying amounts of Random outliers.](image)

The first observation to be made from the results is the necessity of this work. It can be seen that with any amount of outliers, the non-robust baseline fails completely. Both the Individual and the Collaborative error have high mean and median. This contrasts with DCS and GNC, that have low mean and median Individual Error, indicating that the Individual Error is distributed very closely around zero at all times.

However, both DCS and GNC have a high mean Collaborative Error, with DCS typically showing a bigger error. This means that the devices’ trajectories...
are being correctly estimated, with respect to their shape, but their poses in the global frame are being incorrectly estimated.

Although both robust solutions have a large mean Collaborative Error, GNC has a low median error, while the median error is also high for DCS. This allows us to conclude that even though GNC sometimes fails to estimate the correct frame transform between devices, it is able to recover a correct estimate, yielding a low median error. In contrast, due to being a local incremental solver, when DCS gets stuck in a local minimum it fails to escape it.

In Figure 4.3 the device and server mean iteration times are shown for the Random outliers. In Figure 4.4, the same results are shown, without GNC which is $20 \times$ more expensive than the remaining methods.

It can be seen that the computational cost of GNC is up to $20 \times$ higher than that of DCS or the non-robust baseline. Additionally, DCS does not
Figure 4.4: Devices and server mean iteration times with varying amounts of Random outliers, without GNC which is $20 \times$ more expensive than the remaining methods.

incur a high computational cost over the baseline. These results highlight the infeasibility of using GNC as the robust PGO solver for the server since its computational cost far exceeds the available budget of doubling the computational time, with regards to the baseline.

**Experiment 2: Impact of Local Grouped outliers**

This experiment is nearly identical to the previous one, with the only change being in the outlier generation mode and the respective parameters. These parameters are shown in Table 4.2.

<table>
<thead>
<tr>
<th>Outlier generation mode</th>
<th>Local Grouped</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group length ($G$)</td>
<td>20</td>
</tr>
<tr>
<td>Maximum distance ($R$)</td>
<td>50</td>
</tr>
</tbody>
</table>

**Results**

In Figure 4.5, the mean and median Individual Error for the Local Grouped outliers are shown. The Collaborative Error is shown in Figure 4.6.

The iteration times for the Local Grouped outliers are shown in Figure 4.7.

The results of this experiment are nearly identical to the experiment with Random outliers. This allows us to conclude that the existing solutions are mostly invariant to the kind of outliers. Motivated by this, experiments from
this point are conducted only with Random outliers, since it is shown that the generation mode seems to not be important and by keeping with Random outliers we do what is most typical in the literature.

**Experiment 3: Impact of priors on the initial relative devices’ poses**

In order to investigate the importance of priors on the accuracy of robust C-SLAM solutions, we evaluate DCS and GNC without priors on the initial relative poses and with priors with the correct initial relative poses. This is done with Random outliers, varying in the same way as the previous experiments.
Results

In Figure 4.8 the mean and median Collaborative Error for Experiment 3 are shown.

The results show the crucial role that correct priors, in the sense that the relative initial poses of the devices are correct, have in enabling accurate estimation. Without a prior on the initial relative poses of the devices, the results are the same as in experiment 1. However, when accurate priors on the devices’ initial relative poses are available, e.g., via initial GPS measurements for all devices, both robust solutions have a low mean and median Collaborative Error, even for increasing outlier ratios.
4.4 Summary

The results of the experiments conducted allow us to draw the following important conclusions:

- Robustification of the Server-Assisted C-SLAM solution is vital. The non-robust baseline is incapable of dealing with even a small amount of outliers.

- Robust solutions have low Individual Error, even for increasing amounts of outliers.

- If priors on the devices’ initial relative poses are available, DCS yields a low Collaborative Error as well as computational performance similar to the baseline solution.

- In the absence of accurate priors, only GNC yields a low Collaborative Error. However, this has too high of a computational cost to be used as the PGO solver for the optimiser in the main update loop in the server.

This motivates the idea presented in the next chapter, which is the core contribution of this thesis: attempt to use GNC to generate priors from the inter-agent loop closures by solving an easier problem. By using GNC on this easier problem, we hope that it becomes computationally feasible to use it for the critical task of estimating the priors. Having the priors, we saw that we can use DCS for the main optimisation and obtain a low Collaborative Error, with a low Individual Error being a given for any robust solution.
Outlier Impact on Server-Assisted C-SLAM
Chapter 5

Robust Frame Estimation

As shown in Chapter 4, the existence of priors for each agent’s initial relative pose in a shared frame is crucial in enabling the use of robust PGO solvers with real-time performance. In this chapter, a method is proposed to obtain a robust estimate of these priors. It is shown that these estimates allow the existing robust PGO solvers to achieve the desired accuracy while maintaining acceptable computational costs.

In Figure 5.1 a diagram of the system architecture for the C-SLAM back-end, with a pre-processor of the inter-agent loop closures, is shown. The proposed solutions for the preprocessor fulfil two functions: they generate priors for the initial devices’ relative poses and they estimate which inter-agent loop closures are inliers or outliers.

![Figure 5.1: Proposed system architecture for a robust C-SLAM back-end.](image)
Several solutions have been proposed for the pre-processor. PCM [28] proposes pre-filtering loop closure measurements before giving them to the main optimiser. However, since PCM accomplishes this through a graph-based method, an estimate of the initial relative pose priors is not available from the identified set of inlier loop closures.

In Kimera-Multi [9], the authors propose using GNC to get a robust initial estimate, which is then given to the main optimiser that also runs GNC. However, this solution is not fitting for an incremental method, given that we use these methods because we want to avoid full reoptimisation and in this case, initial estimates are not regularly given. Additionally, in Kimera-Multi the authors claim that since the main optimiser uses GNC, pre-filtering loop closures actually leads to decreased accuracy.

In this chapter, two novel inter-agent loop closure pre-processors are presented. Like PCM, the pre-processors estimate the subset of inter-agent loop closures which are inliers to give to the main optimiser. However, this is accomplished by solving a robust PGO problem, inspired by the robust frame initialiser in Kimera-Multi. Thus, an estimate of the initial pose for every device is obtained as well. This estimate is then passed on to the main optimiser as a prior factor.

The solutions presented are called Robust Frame Estimator (RFE) and More Relaxed Robust Frame Estimator (RFE:)) (the short form is read as RFE-happy). The robust PGO problem solved by each of these solutions is a relaxation of the full PGO problem in (2.25), with the problem for RFE:) being a further relaxation of the problem for RFE.

We show that the proposed methods present different trade-offs between accuracy and computational performance. While RFE is slow and not suitable for real-time application, it has very high accuracy. RFE:) on the other hand, is very fast, being suited for a real-time application and integration with incremental solvers. This comes at the cost of accuracy, compared to RFE, but it still outperforms PCM. By integrating our pre-processors with the main optimiser running DCS as the robust PGO solver, we show that the accuracy of the naive GNC implementation can be improved, with RFE, or approach, with RFE:), but with the latter having a much lower computational cost.

### 5.1 Robust Frame Estimator

RFE and RFE:) are based on relaxations of the main PGO problem. In order to help understand these relaxations consider the two-device pose graph example shown in Figure 5.2.
Given that each measurement specifies the variables involved in it and its probabilistic measurement function, the involved variable nodes and factor nodes are fully specified by a measurement. Thus, the set of measurements uniquely determines the pose graph and there exists a unique mapping, $K$, from a set of measurements to a pose graph, $K : Z \rightarrow \mathcal{G}$. The pose graph shown in Figure 5.2, $\mathcal{G}$, is obtained from the complete set of measurements, $Z$, i.e., $\mathcal{G} = K(Z)$.

As stated in Section 2.1, odometry is assumed to be outlier-free while the remaining measurements are not. Thus, in order to isolate the issue of estimating the device’s priors, intra-agent loop closures are ignored since they might contain outliers. The relaxed problem consists then of solving the robust PGO problem with the variable nodes and the subset of measurements consisting only of odometry and inter-agent loop closures, $Z' = Z^O \cup Z^{IE}$.

In Figure 5.3, this relaxation is applied to the pose graph of Figure 5.2, obtaining a relaxed pose graph, $\mathcal{G}' = K(Z')$.

Since the frame estimator only needs to estimate the prior for each device’s initial pose and determine which loop closures are inliers or outliers, not all variable nodes are needed.

Observing the example graph in Figure 5.3, it can be seen that there are chains formed exclusively by variable nodes and odometry factor nodes. These chains can be condensed into a single odometry factor node and all intermediate nodes can be discarded. The mechanism used for this condensation is described in detail in Appendix B. Additionally, variable...
Figure 5.3: Relaxation of the pose graph from Figure 5.2 through the removal of all intra-agent loop closures.

Figure 5.4: Simplification of the relaxed pose graph from Figure 5.3 through the condensation of odometry measurements.

This simplified pose graph, $\hat{G}$, is the pose graph that RFE attempts to solve the robust PGO problem over. This pose graph has some desirable properties when compared to the original pose graph, $G$. First, it grows proportionally to the number of inter-agent loop closures, $L$. It can have at most $N(L + 1)$ variable nodes and $3L$ factor nodes, where $N$ is the number of devices. Thus, even if the C-SLAM algorithm runs for a long time, this problem scales more favourably than the main problem since it is not proportional to the number of steps. Second, the graph remains sparse, allowing the use of sparse solvers.

After solving the robust PGO, RFE also imposes an additional condition for identifying priors and letting two devices establish loop closures with each other. As discussed in Section 4.2, in the absence of priors on the initial pose,
a single loop closure can neither be accepted nor rejected. Therefore, it is not enough that the robust PGO solver classifies a loop closure as an inlier, it must also be part of a cycle in the pose graph where every loop closure is an inlier. Otherwise, the identified inlier is not consistent with any other inlier and cannot be trusted. Having identified all the verifiable inliers, RFE can determine the initial relative pose priors.

The implementation for RFE is described in Algorithm 5.

Algorithm 5 Implementation for RFE

Input: $Z^O, Z^{IE}$
Output: $\hat{X}_0, \hat{Z}^{IE}_{\text{inliers}}, \hat{Z}^{IE}_{\text{outliers}}$

$\text{PoseGraph} \leftarrow \emptyset$
$\text{PoseGraph} \leftarrow \text{UpdatePoseGraph}(\text{PoseGraph}, Z^{IE})$

$Z^{O'} \leftarrow \text{CondenseOdometry}(\text{PoseGraph}, Z^O)$
$\text{PoseGraph} \leftarrow \text{UpdatePoseGraph}(\text{PoseGraph}, Z^{O'})$

$\hat{X}_0, \hat{Z}^{IE}_{\text{inliers}}, \hat{Z}^{IE}_{\text{outliers}} \leftarrow \text{RobustPGOSolver}(\text{PoseGraph})$
$\hat{Z}^{IE}_{\text{inliers}}, \hat{Z}^{IE}_{\text{outliers}} \leftarrow \text{RemoveInliersOutsideCycles}(\text{PoseGraph}, \hat{Z}^{IE}_{\text{inliers}}, \hat{Z}^{IE}_{\text{outliers}})$

RFE begin by building the condensed pose graph, $\hat{G}$, from the odometry and inter-agent loop closure measurements. Then, GNC is used for PGO, giving robust estimates for the priors and the inlier and outlier sets of inter-agent loop closures. These inlier and outlier sets are then further processed to guarantee that all inliers form at least one cycle with another inlier, such that there is mutual consistency between inliers.

5.2 Further Relaxation of the Robust Frame Estimator

The relaxed problem derived for RFE is significantly simpler than the main problem. However, it still estimates more poses than those which are necessary, since not only the first node of each device is estimated.

In order to further relax the problem and estimate only those nodes, measurements are treated independently of each other. For each inter-agent loop closure, the path composed of odometry – loop closure – odometry that connects both devices’ initial pose nodes is considered. These paths, for the graph in Figure 5.4, are shown in Figure 5.5.
By considering each loop closure independently from the others, these paths form chains and can be condensed as previously described. Thus, each loop closure is transformed into an estimate of the corresponding frame transform, by combining it with both devices’ odometric estimate \([9, 28]\).

Letting \(\hat{x}_{ij}^k\) be the odometric estimate of the \(i\)-th pose of the \(j\)-th device and denoting the loop closure which connects device \(i\) at pose \(k\) to device \(j\) at pose \(l\), \(\tilde{z}_{i,k}^{j,l}\), the frame transform estimate is

\[
\hat{T}_{i,k}^{j,l} = \hat{x}_{k}^i \oplus \tilde{z}_{i,k}^{j,l} \ominus \hat{x}_{l}^j. \tag{5.1}
\]

In Figure 5.6, the relaxation used by RFE:) for the pose graph of Figure 5.2 is shown.

This pose graph, \(\hat{\mathcal{G}}\), always has \(N\) variable nodes and \(L\) factor nodes. However, contrary to the RFE pose graph, \(\hat{\mathcal{G}}\), it becomes dense over time. This can be seen already in Figure 5.6, since the relaxed pose graph, \(\hat{\mathcal{G}}\), is fully connected. This densification prevents the use of sparse solvers. However, we verified that this does not significantly impact computational performance. The implementation for RFE:) is shown in Algorithm 6.

The functioning of RFE:) is very similar to RFE, with the main difference
Algorithm 6 Implementation for RFE:

**Input:** $Z^O, Z^{1E}$

**Output:** $\hat{X}_0, \hat{Z}^{1E}_{\text{inliers}}, \hat{Z}^{1E}_{\text{outliers}}$

\[
\text{PoseGraph} \leftarrow \emptyset \\
\hat{T} \leftarrow \text{FrameTransformEstimates}(Z^O, Z^{1E})
\]

\[
\text{PoseGraph} \leftarrow \text{UpdatePoseGraph}(\text{PoseGraph}, \hat{T})
\]

\[
\hat{X}_0, \hat{T}_{\text{inliers}}, \hat{T}_{\text{outliers}} \leftarrow \text{RobustPGOSolver}(\text{PoseGraph})
\]

\[
\hat{T}_{\text{inliers}}, \hat{T}_{\text{outliers}} \leftarrow \text{RemoveInliersOutsideCycles}(\text{PoseGraph}, \hat{T}_{\text{inliers}}, \hat{T}_{\text{outliers}})
\]

\[
\hat{Z}^{1E}_{\text{inliers}}, \hat{Z}^{1E}_{\text{outliers}} \leftarrow \text{LoopClosuresFromEstimates}(\hat{T}_{\text{inliers}}, \hat{T}_{\text{outliers}})
\]

being that the graph that is constructed is much more condensed. The other important difference is that the factors in this graph are not directly the inter-agent loop closures that should be passed to the main optimiser. Therefore, after running the optimisation and inlier cycle detection, the loop closure measurements have to be recovered from the frame transform factors.

### 5.3 Putting It All Together

The interaction between the frame estimator and the main optimiser is the same for both methods. Whenever a new inter-agent loop closure is detected, it is given to the frame estimator. The frame estimator then uses the inter-agent loop closures to estimate the priors for the devices as well as detect the set of inlier inter-agent loop closures.

These priors are then used by the main optimiser as priors on each device’s initial relative pose and only the inter-agent loop closures identified as being inliers by the frame estimator are used in the main optimiser. A loop closure can change status, from being classified as an inlier or an outlier, when new data becomes available. Thus, loop closures can be given or taken from the main optimiser by the frame estimator. A high-level description of this process can be seen in the diagram shown in Figure 5.7. The process is described in further detail in Algorithm 7.
Algorithm 7 Server Back-end with Robust Frame Estimator

\begin{algorithm}
\begin{algorithmic}
\STATE GlobalPoseGraph $\leftarrow \emptyset$
\STATE PreProcessor $\leftarrow \text{InitPreProcessor}()$
\FOR{each device $d$}
\STATE $\hat{X}^{d} \leftarrow \emptyset$
\STATE $\hat{X} \leftarrow \{\hat{X}^{1}, \ldots, \hat{X}^{N}\}$
\STATE $t \leftarrow 0$
\ENDFOR
\WHILE{devices are running}
\FOR{each device $d$}
\STATE $Z^{\text{new}} \leftarrow \text{GetMeasurementsFromDevice}(d)$
\STATE $Z_{\text{new}} \leftarrow Z_{\text{new}}^{1} \cup \ldots \cup Z_{\text{new}}^{N}$ \COMMENT{Get device measurements}
\STATE $Z_{\text{new}}^{O}, Z_{\text{new}}^{IA}, Z_{\text{new}}^{IE} \leftarrow Z_{\text{new}}$ \COMMENT{Separate inter-agent loop closures}
\ENDFOR
\STATE GlobalPoseGraph $\leftarrow \text{UpdatePoseGraph}(\text{GlobalPoseGraph}, Z_{\text{new}}^{O} \cup Z_{\text{new}}^{IA})$
\STATE PreProcessor $\leftarrow \text{UpdatePreProcessor}(\text{PreProcessor}, Z_{\text{new}}^{O} \cup Z_{\text{new}}^{IE})$
\IF{IsServerUpdateTime($t$)}
\STATE $\hat{Z}_{\text{inliers}}^{IE}, \hat{Z}_{\text{outliers}}^{IE}, \hat{X}_{0} \leftarrow \text{RobustFrameEstimation}(\text{PreProcessor})$
\STATE GlobalPoseGraph $\leftarrow \text{RemoveMeasurements}(\text{GlobalPoseGraph}, \hat{Z}_{\text{outliers}}^{IE})$
\STATE GlobalPoseGraph $\leftarrow \text{AddOrUpdatePriors}(\text{GlobalPoseGraph}, \hat{X}_{0})$
\STATE GlobalPoseGraph $\leftarrow \text{UpdatePoseGraph}(\text{GlobalPoseGraph}, \hat{Z}_{\text{inliers}}^{IE})$
\STATE $\hat{X} \leftarrow \text{PGOSolver}(\text{GlobalPoseGraph})$
\FOR{each device $d$}
\IF{ShouldGeneratePoseGraph($d, t$)}
\STATE PoseGraph$^{d} \leftarrow \text{GeneratePoseGraph}(d, \text{GlobalPoseGraph}, \hat{X})$
\STATE SendPoseGraph($d, \text{PoseGraph}^{d}$)
\ENDIF
\ENDFOR
\STATE $t \leftarrow t + 1$
\ENDIF
\end{algorithmic}
\end{algorithm}
5.4 Experimental Design & Results

Unless stated otherwise, all datasets generated for these experiments have the parameters which do not concern outlier generation presented in Table 4.1. For each experiment parameter combination, ten runs for each experiment with different random seeds are conducted.

Taking into account that the experiments in Chapter 4 show that all robust solutions have an Individual Error which is nearly identical to when there are no outliers, in these experiments only the Collaborative Error is considered when discussing accuracy.

**Experiment 1: RFE and RFE:) Tuning**

The accuracy of the proposed methods is dependent on the performance of GNC with TLS kernel as a robust PGO solver. Since the TLS kernel (2.21) has a tuning parameter, these solutions might require tuning. The GNC authors [12] claim that the threshold should be set to the 99% percentile of the $\chi^2$ distribution, but that is for the case where it is solving the full PGO problem, with $G$. Additionally, in [35] it is shown that a solution using PCM, when also using a $\chi^2$-based threshold, coupled with a less robust PGO solver than GNC benefits from a lower threshold. By having a lower threshold, PCM is more conservative in classifying a loop closure as an inlier, leading to less outliers being passed to the optimiser and better performance.

Therefore, since both RFE and RFE:) are meant to be used with DCS,
which is less robust than GNC, tuning them is likely to be important. In order to tune RFE and RFE:), identical experiments are run. Datasets are generated with Random outliers, with the outlier ratio varying between 0% and 50%, in increments of 10%.

In order to find an appropriate working point for each of the frame estimators, several values for the TLS quantile threshold, \( \gamma \), are tested, such that \( \gamma \in \{0.01, 0.1, 0.5, 0.9, 0.99\} \).

**Results**

In Figure 5.8, the mean and median Collaborative Error is shown for the proposed architecture using RFE with different tuning values. The same results are shown in Figure 5.9 for RFE:).

![Figure 5.8: Mean and median Collaborative Error for different RFE tuning parameters, with varying amounts of Random outliers.](image1)

![Figure 5.9: Mean and median Collaborative Error for different RFE:) tuning parameters, with varying amounts of Random outliers.](image2)
For both of the frame estimators, it can be seen that the Collaborative Error is generally lower for a lower threshold. However, the increased conservativeness of choosing a low $\chi^2$ quantile also means the number of loop closures which are given to the main optimiser decreases, including inliers. The reduced number of inlier loop closures might prevent devices from collaborating. Thus, we choose the highest threshold that yields a mean Collaborative Error which is no higher than 30 cm, for all the tested values for the outlier ratio.

Therefore, based on these results, in all subsequent experiments, the percentile used for RFE is 50% and for RFE:) it is 10%.

**Experiment 2: Incremental Approaches**

To evaluate the developed solutions, we compare them with the architecture of Figure 5.1 using PCM as the pre-processor, as well as with the naive DCS implementation. This allows us to draw conclusions about how the proposed solutions compare with the state of the art in incremental approaches. Here, the interest is the estimation accuracy and the computational performance, since all solutions are incremental and meant to run in real-time.

Like the previous experiment, datasets are generated with Random outliers and the outlier ratio varies between 0% and 50% in 10% increments.

**Results**

The Collaborative Error for this experiment is shown in Figure 5.10, with the computational performance being shown in Figure 5.11.

![Figure 5.10: Mean and median Collaborative Error for the different incremental approaches, with varying amounts of Random outliers.](image)
Figure 5.11: Mean agent and server iteration times for the different incremental approaches, with varying amounts of Random outliers.

As has already been discussed in Chapter 4, the naive DCS implementation has unacceptable mean and median Collaborative Error. It can be seen that all three of the incremental approaches with the proposed robust architecture improve on it.

The robust frame estimators proposed present a trade-off between estimation accuracy and computational performance. It can be seen that RFE is far slower than the remaining approaches, such that it cannot be considered suitable for a real-time application. However, RFE:) shows an improvement in accuracy over PCM while also presenting a lower computational cost. The computational performance of the methods is shown again in Figure 5.12, now with RFE removed in order to compare the remaining methods.

Figure 5.12: Mean agent and server iteration times for the different incremental approaches, without RFE and with varying amounts of Random outliers.

From Figure 5.12, it can be seen that RFE:) is actually faster than the naive DCS implementation, which was seen to be as performant as the non-robust baseline. Meanwhile, PCM is up to $6 \times$ slower than naive DCS while being
less accurate than RFE:).

**Experiment 3: Batch Approaches**

In order to compare with the state of the art in estimation accuracy, we also compare the proposed architecture with RFE and RFE:) with the naive GNC implementation, which is implemented in an incremental batch fashion when used in Algorithm 2. This way, conclusions can be drawn about how the proposed solutions compare to the state of the art in estimation quality, disregarding computational performance.

Datasets are generated with Random outliers, with the outlier ratio varying from 0% to 50% in 10% increments.

**Results**

In Figure 5.13, the Collaborative Error for the proposed methods is shown, as well as for GNC which is the state of the art in robust PGO.

![Figure 5.13: Mean and median Collaborative Error for the proposed methods and for GNC, with varying amounts of Random outliers.](image)

It can be seen that while the median Collaborative Error of the proposed methods and GNC are similar, the mean error is lower for both RFE and RFE:). This shows how the proposed methods succeed in outperforming the state of the art robust estimators in terms of accuracy. Additionally, RFE:) is similar in estimation accuracy with RFE while being real-time capable.

**Experiment 4: Scaling with the number of devices**

Since the aim of C-SLAM is enabling a multitude of devices to localise themselves in a shared map, it is important to evaluate the scaling of the
proposed solutions with the number of devices. The scaling is also compared with the existing state of the art solution in pre-filtering, PCM.

In order to attempt to evaluate the impact of the number of devices separate from the impact of the size of the pose graph, we define a normalised graph size, $G$. Then, for a given number of devices, the length of each device’s walk is given by

$$T = \frac{G}{N}.$$  \hfill (5.2)

For this experiment, the normalised graph size is set to $G = 5000$ and the number of devices varies between 2 and 5. The generated dataset has Random outliers and we fixed the outlier ratio at 30%.

**Results**

In Figure 5.14, the Collaborative Error for the incremental approaches is shown. The computational performance is shown in Figure 5.15.

![Figure 5.14: Mean and median Collaborative Error for the different incremental approaches, with a varying number of devices.](image)

It can be seen that the proposed methods scale poorly with the number of devices. Compared with PCM, the accuracy of RFE and RFE:) becomes worse for more than three devices. Additionally, RFE shows worse scaling in terms of computational performance than PCM, even though RFE:) manages to continue to have a lower computational cost for the device interval tested.

However, the architecture of the robust frame estimators and that of PCM must be considered. Both RFE and RFE:) are solving a robust PGO problem based on relaxations of the original Pose Graph SLAM problem. Thus, they are simultaneously attempting to solve all inlier/outlier detections and prior estimations for all devices that are actively collaborating. In contrast, PCM
Figure 5.15: Mean agent and server iteration times for the different incremental approaches, with a varying number of devices.

solves this problem in a pairwise manner, handling inter-agent loop closures between each pair of devices separately.

All previous experiments are targeting 2 devices and it is shown there that, in that case, RFE and RFE:) outperform PCM in accuracy and/or computational performance. Therefore, it is likely that adapting the architecture of the proposed methods to the same pairwise solution used by PCM would lead to an increase in performance.

5.5 Ablation Studies

RFE and RFE:) have two separate components: the initial relative pose prior estimation and the inter-agent loop closure filter. In order to determine the relative importance of these two components for the accuracy of the developed solutions, we now present ablation studies for both RFE and RFE:).

To run these ablation studies, we evaluate RFE/RFE:) against their versions with only the prior estimation or only the loop closure filtering. They are evaluated in a similar setup to the experiments shown in Section 5.4, with Random outliers varying from 10% to 50%, in increments of 10%.

RFE Results

In Figure 5.16, the Collaborative Error for the ablation study of RFE is shown.

From the results, we can immediately draw two conclusions: giving priors on the initial relative poses helps when compared to doing nothing (cf. with DCS in Figure 5.10) and that filtering is responsible for most of the accuracy of RFE. However, it is hard to evaluate the impact of giving priors when also
In order to be able to analyse the results of the solutions with filtering, in Figure 5.17 the Collaborative Error is shown for the same ablation study, removing the solution with only the prior estimation.

When comparing RFE with its version with only filtering, the results are quite similar. Given that only 10 different runs of each experiment were done, the small differences that can be observed in accuracy are not conclusive. Thus, given that in the event of a failure in the filtering component priors are beneficial and when filtering is functioning correctly priors do not harm accuracy, it seems good to give priors with RFE and it seems essential to filter loop closures.
RFE:) Results

In Figure 5.18, the Collaborative Error for the ablation study of RFE:) is shown.

![Mean Collaborative Error](image1)

![Median Collaborative Error](image2)

Figure 5.18: Mean and median Collaborative Error for the ablation study of RFE:), with varying amounts of Random outliers.

Like RFE, for RFE:) filtering is crucial for its accuracy. However, giving priors is still helpful when compared to just running DCS.

In order to be able to analyse the results of the solutions with filtering, in Figure 5.19 the Collaborative Error is shown for the same ablation study, removing the solution with only the prior estimation.

![Mean Collaborative Error](image3)

![Median Collaborative Error](image4)

Figure 5.19: Mean and median Collaborative Error for the ablation study of RFE:), with varying amounts of Random outliers. Only the solutions with filtering are shown since the version without filtering has much lower accuracy and prevents the analysis of the remaining results.

In this case, the results are quite different from RFE and from what we might expect. It seems that for RFE:), giving priors can have a detrimental effect when compared to only doing filtering. This goes against what we would
expect given that the results from Section 4.3 showed that priors made DCS accurate. This indicates that the priors given by RFE:) are too inaccurate. However, it seems that as the outlier ratio increases, giving priors can improve performance as we see with the 50% outlier ratio.

5.6 Summary

In this chapter the core contribution of this thesis was presented: RFE and RFE:). The proposed methods, together with the proposed robust C-SLAM back-end architecture from Figure 5.1, improve on the state of the art in robust C-SLAM.

When considering only real-time capable solutions, RFE:) is shown to outperform PCM in accuracy and computational cost for a small number of devices. However, as the number of devices increases the proposed methods worsen in comparison to the state of the art. Nonetheless, given the differences in the architectures of the proposed implementation for the robust frame estimators and that of PCM, it seems likely that RFE and RFE:) can easily be changed to improve their scaling.

When comparing to GNC, the state of the art in robust SLAM, RFE is shown to be more accurate. This, however, comes at the cost of RFE not being real-time capable.

The ablation studies showed that while the motivating results from Section 4.3 showed the importance of priors for allowing DCS to be accurate, the current implementations of RFE and RFE:) either do not benefit a lot from giving priors or even suffer. This opens up an avenue for exploring how to improve the prior estimation accuracy for the preprocessors in order to achieve even higher localisation accuracy in the full C-SLAM system.

Finalising, the two proposed methods, RFE and RFE:) present a design trade-off between accuracy and computational cost. In each of their target applications, they are shown to outperform the state of the art.
Chapter 6

Conclusions and Future work

In this chapter, we answer the research questions posed in Section 1.3. The limitations in the novel methods developed are discussed and future work, that can build and improve on this thesis is proposed.

6.1 Conclusions

This thesis posed two research questions:

- What is the impact of outliers in Server-Assisted C-SLAM solutions?

- How can outlier robustness be improved in Server-Assisted C-SLAM solutions?

In order to answer the first research question, in Chapter 4 the impact of outliers in a baseline non-robust solution and naive robust solutions are investigated. From the results shown we are able to conclude that outliers are highly detrimental to non-robust solutions, which fail with a minimum amount of outliers.

When the initial relative poses of devices are not known, for example in an indoor environment with no known artefacts that allow for initial localisation, naive robust solutions also fail. Incremental robust solutions, which are real-time capable, yield a very high estimation error, while batch robust solutions, which are accurate, are too slow for real-time.

However, we show that if the initial relative poses are known a priori, then both incremental and batch robust solutions are accurate.

Motivated by the answer to the first question, we answer the second question by designing two novel methods, RFE and RFE:), that give a robust
estimate of the initial relative poses between devices. This is shown to increase the accuracy of incremental solutions in situations where these initial poses are not known, achieving the desired effect of enabling a real-time capable outlier robust Server-Assisted C-SLAM solution.

6.1.1 Impact & Reflection

The results of our work enable robust C-SLAM to run on resource-constrained devices, due to the Server-Assisted architecture. This has the potential to bring a lot of benefits, by enabling the miniaturisation of devices that need C-SLAM, like mobile robots and XR devices, and the reduction of battery sizes. However, it also has the potential for enabling malicious applications, like mass surveillance of the world through everyday devices, like our smartphones.

This ambivalence of potential outcomes that can be achieved due to the solutions developed, both good and bad, means that caution must be taken in the deployment of these robust Server-Assisted C-SLAM systems. Constant maintenance and inspection are necessary to guarantee that these systems are not compromised and the wealth of information that they have available is not used for nefarious purposes.

6.2 Limitations & Future Work

The main limitation in the development of this thesis was the lack of available open-source implementations of many of the state of the art methods presented in the literature. In particular, a very promising robust incremental PGO solver, riSAM [27], was not available at the time of writing. This means that DCS was used instead. However, given the impressive estimation accuracy obtained by coupling DCS as the main optimiser with RFE or RFE:, we do not expect that using riSAM would improve estimation accuracy by a lot. Where we believe it has the potential of really improving our work is in the computational performance, since it can replace GNC.

Future work could look into several directions for improving RFE and RFE:). As was referred in Section 5.4, the developed methods present poor scaling with the number of devices. However, since they improve over existing methods for two or three devices, an architecture which only handles the devices in groups of two or three at a time could reap the benefits of our methods while improving scaling.
The issue that RFE faces with being too slow is due to the fact that the robust PGO solver it uses is GNC, which is a batch method. An avenue to explore in order to improve the computational performance of RFE, and RFE:) but to a lesser extent, is to change the PGO solver to riSAM, which claims to match or beat GNC in estimation accuracy, while keeping computational costs real-time capable.

Also exploring the potential of riSAM, the main optimiser could replace DCS with riSAM. This should make the main optimiser more robust and allow the use of a higher $\chi^2$ quantile threshold, which would help prevent RFE and RFE:) from delaying or hindering collaboration.

We saw from the ablation studies that the role that priors can have is not yet fully realised. Future work can look in the direction of improving the initial relative pose prior’s estimation accuracy and reducing the prior’s estimated covariance. Looking into dynamically deciding when to update priors can also be explored, e.g., looking for analogues to the rules designed for the loop closure filtering demanding that inlier loop closures belong to a cycle of inliers.
Conclusions and Future work
References


Appendix A

Outlier-Free Dataset Generation

We now described how the outlier-free pose graph generation works in Kollagen [33].

The dataset generation begins with the ground truth generation for each device. The ground truth trajectories are generated by taking a random walk in a 2D grid world, constrained to only being able to change direction every $s$ steps. Additionally, a parameter $n_d$ specifies whether the device can turn backwards or if it can only go forward or turn sideways. This means each pose, $x'[k] = (x'[k], y'[k], \theta'[k])$, which is composed by the position $(x'[k], y'[k]) \in \mathbb{Z}^2$ and the orientation $\theta'[k] \in \{-2, -1, 0, 1\} \cdot \frac{\pi}{2}$, is generated by

\begin{align*}
\theta'[k] &= f(\theta'[k - 1], s, n_d), \quad \text{(A.1)} \\
x'[k] &= x'[k - 1] + \cos(\theta'[k]), \quad \text{(A.2)} \\
y'[k] &= y'[k - 1] + \sin(\theta'[k]). \quad \text{(A.3)}
\end{align*}

The function $f$ randomly generates the next orientation, but guarantees that it obeys the parameters $s$ and $n_d$ and that it stays normalised in the set $\{-2, -1, 0, 1\} \cdot \frac{\pi}{2}$.

After the random walk has been generated, a parameter $c \in \{0, 1\}$ controls whether to centre each device’s trajectory in order for its centre of mass to be at the origin. Finally, the random walk is rescaled, such that the length of the $s$ steps before turning is a length $S$. With this rescaling, it is possible that the ground truth trajectory is not in $\mathbb{Z}^2$, but in $\mathbb{R}^2$. Thus, the ground truth poses, $x[k]$, are
\[ \theta[k] = \theta'[k], \quad (A.4) \]
\[ x[k] = \frac{S}{s} \left( x'[k] - \frac{c}{T} \sum_{i=1}^{N} x'[i] \right), \quad (A.5) \]
\[ y[k] = \frac{S}{s} \left( y'[k] - \frac{c}{T} \sum_{i=1}^{N} y'[i] \right), \quad (A.6) \]

where \( T \) is the length of the generated trajectory. The centring of the devices’ trajectories promotes more encounters between different devices, making it a useful setting for testing in C-SLAM. In Figure A.1 a ground truth trajectory for a single device on this grid world is shown.

Figure A.1: Ground truth trajectory on a grid world for a single example device.

After having generated the ground truth trajectories, the measurements, \( Z \), which form the pose graph are generated. Odometry measurements, \( Z^O \), are composed of the distance travelled and the change of direction, \( \Delta \hat{\ell}[k] \) and \( \Delta \hat{\theta}[k] \), respectively. Since odometry is outlier-free, these measurements are generated by simply adding Gaussian noise to the true value

\[ \Delta \hat{\ell}[k] = ||x[k] - x[k-1]||_{2,p} + \tilde{\ell}[k], \quad (A.7) \]
\[ \Delta \hat{\theta}[k] = \theta[k] - \theta[k-1] + \tilde{\theta}[k], \quad (A.8) \]
where $|.|_{2,p}$ is the 2-norm of the position component of a pose, i.e., $|x[k]|_{2,p} = \sqrt{x^2[k] + y^2[k]}$. The noise is drawn from Gaussian distributions, $l[k] \sim \mathcal{N}(0, \sigma_l^2)$ and $\theta[k] \sim \mathcal{N}(0, \sigma_\theta^2)$, where the noise standard deviation for each component can be set to the desired value. This measurement can then be converted into a full transformation as

$$\Delta \hat{x}[k] = \Delta \hat{l}[k] \cos \Delta \hat{\theta}[k] \quad (A.9)$$
$$\Delta \hat{y}[k] = \Delta \hat{l}[k] \sin \Delta \hat{\theta}[k] \quad (A.10)$$
$$\Delta \hat{\theta}[k] = \Delta \hat{\theta}[k] \quad (A.11)$$

For inlier loop closure measurements, Kollagen begins by identifying pairs of nodes for which a loop closure is going to be generated. The method for this identification is based on the distance between nodes. Tuning parameters are available which give control over how the distance between nodes relates to the probability of a loop closure between them being made (See [33] for details).

Having chosen two nodes with poses $x_1$ and $x_2$, a noisy measurement is generated. Since this measurement is an inlier, Gaussian noise is simply added to the true value. Loop closure measurements give the full transform between the two poses, so they are composed of the relative position, $(\Delta x, \Delta y)$, and the relative orientation, $\Delta \theta$, of $x_2$ with respect to $x_1$. Each component of this measurement is then affected by Gaussian noise, with the standard deviation being specified when generating the data.

A loop closure measurement is then represented by the ordered tuple of the first node, the second node and the measurement relating them, with the identifier of a node being given by the device and the node. Thus, a loop closure is represented as $(D_1, N_1, D_2, N_2, z_{lc})$. For intra-agent loop closures, it is implicit which device they refer to and the device identifiers are dropped. The subsets of inlier intra- and inter-agent loop closures are $Z_{inliers}^{IA}$ and $Z_{inliers}^{IE}$, respectively.

In our work, the trajectories are always centred ($c = 1$), the length of $s$ steps is always $1$ ($S = 1$) and the devices can always turn backwards.
Appendix A: Outlier-Free Dataset Generation
Appendix B

3-Dimensional Special Euclidean Group

In this thesis, we make use of the properties of the 3-Dimensional Special Euclidean Group, $SE(3)$, since elements of this group are very convenient to represent poses [36].

An element of $SE(3)$ has a position component, which is a vector in $\mathbb{R}^3$, and an orientation component, which is an element of the 3-Dimensional Orthogonal Group, $SO(3)$. Typically, the orientation component is represented as a rotation matrix in $\mathbb{R}^{3\times3}$.

Therefore, $SE(3)$ can be defined as $SE(3) = \{(x, R) : x \in \mathbb{R}^3 \land R \in SO(3)\}$. Elements of $SE(3)$ are typically represented as matrices in $\mathbb{R}^{4\times4}$. An $SE(3)$ element, $X$, with position component $t$ and orientation component $R$ is represented by the following matrix

$$X = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}. \quad (B.1)$$

The group operation is denoted by $\oplus$. Considering two $SE(3)$ elements, $X_1$ and $X_2$, and their respective matrix representations, the group operation corresponds to the matrix multiplication of their respective representations. The group operation is not commutative, therefore the matrices must be multiplied in the correct order. Thus, $X_1 \oplus X_2$ is given by

$$X_1 \oplus X_2 = \begin{bmatrix} R_1 & t_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_2 & t_2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_1 R_2 & R_1 t_2 + t_1 \\ 0 & 1 \end{bmatrix}. \quad (B.2)$$

The group action on $\mathbb{R}^3$ is also denoted by $\oplus$. Consider a group element, $X$,
and a vector in $\mathbb{R}^3$, $\mathbf{u}$. We can first extend $\mathbf{u}$ to 4 dimensions by adding a 1, obtaining $\tilde{\mathbf{u}}$. Then, the result of the application of $X$ to $\mathbf{u}$ is given by the first 3 components of the result of the multiplication of the matrix representation of $X$ with $\tilde{\mathbf{u}}$,

$$
\begin{bmatrix}
\mathbf{R} & \mathbf{t} \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
\mathbf{u} \\
1
\end{bmatrix} =
\begin{bmatrix}
\mathbf{Ru} + \mathbf{t} \\
1
\end{bmatrix}
\Rightarrow
X \oplus \mathbf{u} = \mathbf{Ru} + \mathbf{t}.
$$

(B.3)

For every group element, $X$, there is an inverse, $Y$, such that

$$
X \oplus Y = I_{4 \times 4} \Leftrightarrow Y = \begin{bmatrix}
\mathbf{R}^T & -\mathbf{R}^T \mathbf{t} \\
0 & 1
\end{bmatrix},
$$

(B.4)

where $I_{4 \times 4}$ is the identity matrix of $\mathbb{R}^{4 \times 4}$. The inverse of $X$ is denoted by $\ominus X$. For simplicity, the composition of an element, $X_1$, with the inverse of another element, $X_2$, is written as $X_1 \ominus X_2$, instead of the long form, $X_1 \ominus (\ominus X_2)$.

Since $SE(3)$ is a Lie group, not only is it a group, it is also a differentiable manifold. This means an exponential map exists, which maps $\mathbb{R}^6$ to $SE(3)$. This exponential map is denoted as $\exp(x)$, where $x \in \mathbb{R}^6$. This map is injective, meaning there is a complete covering of $SE(3)$, but not surjective, which means that more than one vector in $\mathbb{R}^6$ can map to the same group element.

A logarithmic map also exists, which maps group elements to $\mathbb{R}^6$ vectors. Since the exponential map is not surjective, there are several branches for the logarithmic map. The logarithmic map is denoted as $\log(X)$, where $X \in SE(3)$.

For every group element, $X$, there exists an adjoint map, $\text{Adj}_X$, that allows changing the composition of the exponential from one side to the other,

$$
X \oplus \exp(\varepsilon) = \exp(\text{Adj}_X \varepsilon) \oplus X, \quad \varepsilon \in \mathbb{R}^6.
$$

(B.5)

### B.1 Noisy Measurements

Consider now that we use $SE(3)$ elements to represent pose measurements. For pose measurements, a noisy measurement, $\tilde{\mathbf{z}}$, is said to be Gaussian if

$$
\tilde{\mathbf{z}} = \mathbf{z} \oplus \exp(\varepsilon), \quad \varepsilon \sim \mathcal{N}(\mathbf{0}, \Sigma_z),
$$

(B.6)

where $\mathbf{z}$ is the noise-free value of the quantity that is being measured and $\mathcal{N}(\mathbf{0}, \Sigma_z)$ is the 6 dimensional, zero-mean, Gaussian distribution.
Since the measurements are group elements, the linear adjoint map allows changing the exponential noise term to either side of the measurement,

\[ z \oplus \exp(\varepsilon) = \exp(\text{Adj}_z \cdot \varepsilon) \oplus z. \]  
(B.7)

In Figure B.1 an example of a chain of odometry measurements is shown. Assuming each measurement is Gaussian, it can be written like (B.6), \( \tilde{z}_i = z_i \oplus \exp(\varepsilon_i) \). We also assume the measurements have independent noise.

\[ 
\begin{array}{c}
\text{A0} \\
\bullet \\
\text{A1} \\
\text{A2} \\
\bullet \\
\text{A3} \\
\end{array}
\]

Figure B.1: Example of a chain of odometry measurements.

Condensing measurements \( \tilde{z}_1 \) and \( \tilde{z}_2 \) yields

\[ \tilde{z}_1 \oplus \tilde{z}_2 = z_1 \oplus \exp(\varepsilon_1) \oplus z_2 \oplus \exp(\varepsilon_2) = z_1 \oplus z_2 \oplus \exp(\text{Adj}_z \varepsilon_1) \oplus \exp(\varepsilon_2). \]  
(B.8)

By taking a second-order approximation of the composition of exponentials, the condensed measurement can be simplified into

\[ z_1 \oplus z_2 \oplus \exp(\text{Adj}_z \varepsilon_1) \oplus \exp(\varepsilon_2) \approx z_1 \oplus z_2 \oplus \exp(\text{Adj}_z \varepsilon_1 + \varepsilon_2). \]  
(B.9)

It should be noted that this is one possible approximation, but others exist [37], allowing for greater precision. However, we found that this approximation led to good results while being very straightforward to implement, eliminating the need for a more precise but more complicated approximation.

Since the adjoint is a linear map, \( \varepsilon' = \text{Adj}_z \varepsilon_1 + \varepsilon_2 \) is a Gaussian random variable. Additionally, since the measurement noise is assumed to be independent, it is distributed according to

\[ \varepsilon' \sim \mathcal{N}(0, \text{Adj}_z \Sigma_{z_1} \text{Adj}_z^T + \Sigma_{z_2}). \]  
(B.10)

Thus, the composition of two noisy measurements can be approximated analytically. However, when composing actual measurements the ground truth pose is not available. Therefore, the composition is done by taking the adjoint of the measured value, introducing another degree to this approximation.

In order to extend this composition method to more measurements, one must simply consider the composed measurement, \( \tilde{z}_1 \oplus \tilde{z}_2 \), and the next measurement, \( \tilde{z}_3 \), as \( \tilde{z}_1 \) and \( \tilde{z}_2 \) in the process described here. Thus, an
odometry chain of arbitrary length similar to the one in Figure B.1 can be condensed into only two variable nodes and a single odometry factor node.
In order to be able to perform many tasks, autonomous devices need to understand their environment and know where they are in this environment. Simultaneous Localisation and Mapping (SLAM) is a solution to this problem.

When several devices attempt to jointly solve this problem they use Collaborative SLAM (C-SLAM), but this is a very resource-demanding process. In order to enable resource-constrained devices, like small mobile robots or eXtended Reality (XR) devices, to run C-SLAM we look towards a Server-Assisted C-SLAM architecture to lift the computational burden from these devices.

In a real-world scenario, sensors might fail, the devices might process sensor data wrongly or a malicious actor might inject wrong data into the system. In order for these solutions to be reliable, they must be able to deal with these outliers.
This thesis looks into the impact of outliers in Server-Assisted C-SLAM algorithms and presents two novel solutions for a robust algorithm, based on robust estimation of the initial device poses. We show the novel solutions outperform the state of the art both in estimation accuracy, yielding better estimates of the real device trajectories, and computational performance, making it suitable for device-constrained devices.

Keywords[eng]: SLAM, Robust Estimation, Multi-Device Algorithms.

Abstract[sw]: För att kunna utföra flertalet uppgifter måste autonoma enheter förstå sin miljö och veta var de befinner sig i den här miljön. Simultaneous Localization and Mapping (SLAM) är en lösning på detta problem. När flera enheter försöker lösa detta problem tillsammans använder de Samarbetande SLAM (C-SLAM), men detta är en mycket resurskrävande process. För att möjliggöra att resursbegränsade enheter, så som exempelvis små mobila robotar eller extened Reality (XR)-enheter, ska kunna köra C-SLAM föreslås en serverassisterar C-SLAM-arkitektur beräkningsbörjan kan lyftas från dessa enheter till servern.

I ett verkligt scenario kan sensorer vara felaktiga, enheter behandla sensordata felaktigt eller illvilliga aktörer injicera felaktig data i systemet. Därför undersöker detta arbete effekten av outliers i Serverassisterade C-SLAM-algoritmer och presenterar två nya lösningar för en robust algoritm, baserad på robusta uppskattningar av enhetens initiala positioner. Denna lösning visar sig överträffa likartade lösningar i litteraturen både vad gäller uppskattningsnoggrannhet, vilket ger bättre uppskattningar av den verkliga enhetsbanor och beräkningsprestanda, vilket gör den lämplig för enheter med begränsade resurser.

Keywords[sw]: SLAM, Robust uppskattnings, Algoritmer för flera enheter.
acronyms.tex

%%% Local Variables:
%%% mode: LaTeX
%%% TeX-master: t
%%% End:
% The following command is used with glossaries-extra
%\setabbreviationstyle[acronym]{long-short}
%\glsdisablehyper
% The form of the entries in this file is \newacronym{label}{acronym}{phrase}
% or \newacronym[options]{label}{acronym}{phrase}
% see "User Manual for glossaries.sty" for the details about the options, one example is shown below
% note the specification of the long form plural in the line below
% The following example also uses options
\newacronym{SLAM}{SLAM}{Simultaneous Localization and Mapping}
\newacronym{C-SLAM}{C-SLAM}{Collaborative SLAM}
\newacronym{XR}{XR}{eXtended Reality}
\newacronym{VIO}{VIO}{Visual-Inertial Odometry}
\newacronym{MAP}{MAP}{Maximum a Posteriori}
\newacronym{GNC}{GNC}{Graduated Non-Convexity}
\newacronym{DCS}{DCS}{Dynamic Covariance Scaling}
\newacronym{PCM}{PCM}{Pairwise Consistency Maximisation}
\newacronym{IRLS}{IRLS}{Iteratively Reweighted Least Squares}
\newacronym{ATE}{ATE}{Absolute Trajectory Error}
\newacronym{RMSE}{RMSE}{Root Mean Squared Error}
\newacronym{TLS}{TLS}{Truncated Least Squares}
\newacronym{iCATE}{iCATE}{Incremental Collaborative ATE}
\newacronym{RFE}{RFE}{Robust Frame Estimator}
\newacronym{RFE:)}{RFE:)}{More Relaxed Robust Frame Estimator}