DISTRIBUTED OBJECT-BASED SLAM

A Thesis
Presented to
The Academic Faculty

by

Siddharth Choudhary

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology
December 2017

DISTRIBUTED OBJECT-BASED SLAM

Approved by:

Professor Henrik I. Christensen, Advisor
School of Interactive Computing
Georgia Institute of Technology

Professor Frank Dellaert, Co-Advisor
School of Interactive Computing
Georgia Institute of Technology

Professor James M. Rehg
School of Interactive Computing
Georgia Institute of Technology

Professor Patricio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor John Leonard
Department of Mechanical Engineering
Massachusetts Institute of Technology

Date Approved: August 1, 2017

DEDICATION

Dedicated to Mummy and Papa

ACKNOWLEDGEMENTS

This thesis would not be possible without the support and contribution of many people I collaborated with over the course of my Ph.D. First of all, I would like to thank my advisor, Prof. Henrik Christensen who always ensured that I look at the big picture while selecting a problem to solve for my Ph.D. thesis. He taught me how to scientifically break down difficult problems into solvable subsets and how to solve a problem using a multipronged approach. Despite his busy schedule he always found time to advise me all these years. I'm also in debt to my co-advisor, Prof. Frank Dellaert for teaching me the rigor required at each stage of problem-solving, from writing unit tests during coding to properly writing mathematical equations while writing research papers. I'm also thankful to Prof. Luca Carlone, who I collaborated with on multiple papers during my thesis. Luca taught me the organizational skills required to take an idea and convert it into a successful research project. His detailed feedback on research papers to coding style has been very valuable through out my Ph.D. I'm grateful to my first advisor Prof. P J Narayanan who instilled the excitement of research in me during my undergraduate days and with whom I had the first research experience. I'm also thankful to my other collaborators which include Vadim Indelman, Alexander J.B. Trevor, Carlos Nieto, John Rogers, Varun Murali and Zhen Liu who all have taught me so much.

I will also miss my lab mates in CogRob and BORG lab working on a paper or a demo deadline with them. I wish to acknowledge my colleagues at IRIM and CPL, Rahul, Natesh, Tapo, Abhijit, Nam, Pushkar, Himanshu, Shray, Samarth, Ruffin, Priyam, Varun, Niharika, Carlos and many others, who made my time here enjoyable. I'll miss the people at Asha For Education running group.

I am also grateful for the support of Army Research Lab MAST program which has

funded this research. I'm thankful for the resources provided by the ARL including robots and state of art facilities for my research. I'll miss the time spent during my last year working with the fantastic researchers at ARL.

In the end, I wish to thank my parents who have always supported my dreams and my decision to do a Ph.D. They always motivated me during the bad times and cheered me during the good times. Thank you, mummy and papa, for believing in me. I couldn't have asked for more. I am thankful to bhaiya and bhabhi who sacrificed a lot, kept everyone together during times of crisis and are a true inspiration for me. I wish to thank my wife Akanksha, who left everything familiar back in India and moved to U.S. to live with me. She had to spend most of the last year in the lab with me while I focused on finishing up the thesis. I couldn't have done my thesis in time without her help and I will always be in debt to her.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The use of multiple cooperative robots or mobile devices has the potential to enable fast information gathering, and more efficient coverage and monitoring of large areas. In particular, distributed SLAM, i.e., the cooperative construction of a model of the environment explored by the robots or mobile devices, is fundamental to geotag sensor data (e.g., for pollution monitoring, surveillance and search and rescue), and to gather situational awareness. For military applications, multi-robot systems promise more efficient operation and improved robustness to adversarial attacks. In civil applications (e.g., pollution monitoring, surveillance, search and rescue), the use of several inexpensive, heterogeneous, agile platforms is an appealing alternative to monolithic single robot systems.

In this thesis, we aim at designing a technique that allows each robot or mobile device to build its own object level map while asking for minimal knowledge of the map of the teammates. In particular, we make the following three major contributions:

1. We present a distributed algorithm based on Distributed Gauss-Seidel to estimate the 3D trajectories of multiple cooperative robots from relative pose measurements. This approach has several advantages. It requires minimal information exchange, which is beneficial in the presence of communication and privacy constraints. It has an anytime flavor: after few iterations, the trajectory estimates are already accurate, and they asymptotically convergence to the centralized estimate. The DGS approach scales well to large teams, is resistant to noise and it has a straightforward implementation. We test the approach in simulations and field tests, demonstrating its advantages over related techniques.

2. We present an approach for distributed SLAM which uses object landmarks in a distributed mapping framework. We show that this approach further reduces the information exchange among robots (as compared to feature based DGS), results in a compact, human

understandable map, and has lower computational complexity as compared to low-level feature based distributed mapping.

3. Finally, we extend the previous work to the case where object models are previously unknown and are modeled jointly with Distributed Object-based SLAM. We show that this approach further reduces the memory required to store the object models while maintaining the accuracy at the same level as the state of art RGB-D mapping approaches.

As a future work, we are extending this work in different directions. First, our current approach is based on a nonlinear least squares formulation which is not robust to gross outliers. As a future work, we will focus on designing more general algorithms that are robust to spurious measurements. Second, we plan to extend our experimental evaluation to flying robots. While we demonstrate the effectiveness of our approach in large teams of ground robots, we believe that the next grand challenge is to enable coordination and distributed mapping in swarms of agile micro aerial vehicles with limited communication and computation resources.

CHAPTER I

INTRODUCTION

The deployment of distributed systems in the real world poses many technical challenges, ranging from coordination and formation control, to task allocation and distributed sensor fusion. In this work we tackle a specific instance of the sensor fusion problem. We consider the case in which a team of robots or mobile devices explores an unknown environment and each robot or mobile device has to estimate its trajectory from its own sensor data and leveraging information exchange with the teammates. Trajectory estimation is relevant as it constitutes the backbone for many estimation and control tasks (e.g., geo-tagging sensor data, 3D map reconstruction, position-aware task allocation). Indeed, in our application, trajectory estimation enables distributed 3D reconstruction and localization.

We consider a realistic scenario, in which the robots or mobile devices can only communicate when they are within a given distance. Moreover, also during a rendezvous (i.e., when the robots or mobile devices are close enough to communicate) they cannot exchange a large amount of information, due to bandwidth constraints (e.g., there exist an upper bound on the number of bytes that the robots or mobile devices can exchange).

Moreover, we aim at a technique that allows each robot or mobile device to estimate its own trajectory, while asking for minimal knowledge of the trajectory of the teammates. This "privacy constraint" has a clear motivation in a military application: in case one robot is captured, it cannot provide sensitive information about the areas covered by the other robots in the team. Similarly, in civilian applications, one may want to improve the localization of a device (e.g., a smart phone) by exchanging information with other devices, while respecting users' privacy.

Dealing with bandwidth constraints is challenging for two reasons. First, most approaches for distributed SLAM imply a communication burden that grows quadratically in the number of locations co-observed by the robots [29]; these approaches are doomed to quickly hit the bandwidth constraints. In chapter 3, we alleviated this issue by proposing an approach, based on the distributed Gauss-Seidel method [23], which requires linear communication.

The second issue regards the communication cost of establishing loop closures among robots. When the robots or mobile devices are not able to directly detect each other, loop closures have to be found by comparing raw sensor data; in our setup the robots or mobile devices are equipped with an RGBD camera and exchanging multiple 3D point clouds quickly becomes impractical in presence of communication bounds. We address the second issue in chapter 4 by using an object-level map representation.

Finally, the approach introduced in chapter 4 assumes that the object model of each instance that each mapped in an environment is known in advance. However it can be challenging to store a model of all the object instances due to large intra-class variation. Searching through all the object models for object pose estimation can be computationally demanding. It won't generalize to new unseen instances of the same object category as well.

Therefore, in chapter 5, we proposed an approach to SLAM with joint object modeling and mapping. This approach requires less memory as compared to the state of art RGB-D mapping approaches while maintaining the accuracy level of the state of art RGB-D mapping approaches. In chapter 6, we integrate the joint object modeling and mapping framework with distributed object based SLAM approach and show that this approach extends the previous work [23, 24, 25] to the case where object models are previously unknown. We shows that this approach further reduces the memory required to store the object models. In chapter 7 we conclude the thesis and discuss the future work.

The overall thesis statement is as follows:

Thesis Statement. *Using objects as landmarks in a distributed SLAM framework optimized using the state of art distributed optimizer both reduces the communication bandwidth and the memory used by each robot, outputs a human understandable map and improves the robustness and scalability of distributed SLAM.*

Contributions. We support the above thesis statement using the following contributions:

- **Distributed Gauss-Seidel Algorithm [23].** We present a distributed algorithm to estimate the 3D trajectories of multiple cooperative robots or mobile devices from relative pose measurements. Our approach leverages recent results [20] which show that the maximum likelihood trajectory is well approximated by a sequence of two quadratic subproblems. The main contribution of this work is to show that these subproblems can be solved in a distributed manner, using the *distributed Gauss-Seidel* (DGS) algorithm. This approach has several advantages. It requires minimal information exchange, which is beneficial in presence of communication and privacy constraints. It has an anytime flavor: after few iterations the trajectory estimates are already accurate, and they asymptotically convergence to the centralized estimate. The DGS approach scales well to large teams, is resistant to noise and it has a straightforward implementation. We test the approach in simulations and field tests, demonstrating its advantages over related techniques.
- **Distributed Object based SLAM with Known Object Models [24].** Traditional approach for distributed mapping typically make use of feature-based maps which are composed of low level primitives like points and lines which model space based on its geometric shape [32]. These maps become memory intensive for long-term operation, contain a lot of redundant information (useless to represent a planar surface with thousands of points), lack the semantic information necessary for performing

wider range of tasks (eg. manipulation tasks).

To solve these issues, we present an approach for Distributed SLAM which uses object landmarks [108] in a distributed mapping framework [23, 25]. We show that this approach further reduces the information exchange among robots or mobile devices (as compared to feature based DGS), results in compact, human understandable map, and has lower computational complexity as compared to low level feature based distributed mapping. As compared to other object based SLAM approaches [108, 95], we show results in a larger scale environment using a large number of object categories applied to multi robot setting.

- **Distributed Object based SLAM with Joint Object Modeling and Mapping [25].** The previous approach assumes that the object model of each instance that each mapped in an environment is known in advance. However it can be challenging to store a model of all the object instances due to large intra-class variation. Searching through all the object models for object pose estimation can be computationally demanding. It won't generalize to new unseen instances of the same object category as well. Therefore, we extend the previous work [23, 24, 25] to the case where object models are previously unknown and are modeled jointly with Distributed Object based SLAM. We use the off-the-shelf lightweight convolutional network based object detectors to detect object at categorical level which are then modeled at instance level by integrating detection across frames. The modeled object instance are then data associated against other instances seen by the same robot or other robots to generate object-object constraint. We show that this approach generalizes distributed object-based SLAM to unseen object models while further reducing the memory required to store the object models.

CHAPTER II

RELATED WORK

2.1 Distributed Estimation

Distributed estimation in multi robot systems is currently an active field of research, with special attention being paid to communication constraints [94], heterogeneous teams [9, 59], estimation consistency [8], and robust data association [58, 37]. The robotics literature offers distributed implementations of different estimation techniques, including Extended Kalman filters [103, 144], information filters [124], and particle filters [57, 18]. More recently, the community reached a large consensus on the use of maximum likelihood (ML) estimation (maximum a-posteriori, in presence of priors), which, applied to trajectory estimation, is often referred to as *pose graph optimization* or *pose-based SLAM*. ML estimators circumvent well-known issues of Gaussian filters (e.g., build-up of linearizion errors) and particle filters (e.g., particle depletion), and frame the estimation problem in terms of non-linear optimization. In multi robot systems, ML trajectory estimation can be performed by collecting all measurements at a centralized inference engine, which performs the optimization [4, 66, 9]. Variants of these techniques invoke partial exchange of raw or preprocessed sensor data [76, 58].

In many applications, however, it is not practical to collect all measurements at a single inference engine. When operating in a hostile environment, a single attack to the centralized inference engine (e.g., one of the robots) may threaten the operation of the entire team. Moreover, the centralized approach requires massive communication and large bandwidth. Furthermore, solving trajectory estimation over a large team of robots can be too demanding for a single computational unit. Finally, the centralized approach poses privacy concerns as it requires to collect all information at a single robot; if an enemy robot

is able to deceive the other robots and convince them that it is part of the team, it can easily gather sensitive information (e.g., trajectory covered and places observed by every robot). These reasons triggered interest towards *distributed trajectory estimation*, in which the robots only exploit local communication, in order to reach a consensus on the trajectory estimate. Nerurkar *et al.* [86] propose an algorithm for cooperative localization based on distributed conjugate gradient. Franceschelli and Gasparri [43] propose a gossip-based algorithm for distributed pose estimation and investigate its convergence in a noiseless setup. Aragues *et al.* [5] use a distributed Jacobi approach to estimate a set of 2D poses, or the centroid of a formation [6]. Araguez *et al.* [7] investigate consensus-based approaches for map merging. Knuth and Barooah [68] estimate 3D poses using distributed gradient descent. Cunnigham *et al.* [30] use Gaussian elimination, and develop an approach, called DDF-SAM, in which each robot exchange a Gaussian marginal over the *separators* (i.e., the variables observed by multiple robots); the approach is further extended in [29], to avoid storage of redundant data, thought the use of *anti-factors*.

The literature on parallel computing and hierarchical approaches is also relevant: the idea is still based on Schur complement, which has been also expoited as a key component for hierarchical approaches for large-scale mapping [89, 50, 116]. *Decoupled stochastic mapping* was one of the earliest approach for submapping proposed by Leonard and Feder [78]. Leonard and Newman [79] propose a constant time SLAM solution which achieves near-optimal result under the assumption that the robot makes repeated visits to all regions of the environment. Frese et al. [46] proposed multi-level relaxation resulting in a linear time update. Frese [45] proposed TreeMap algorithm which is similar to Thin junction tree filter (TJTF). It divides the environment into a parts-whole-hierarchy represented as a binary tree. Since it uses a balanced tree, update requires only $\mathcal{O}(k^3 \log n)$ time. Estrada et al. [39] presented an hierarchical SLAM framework which consist of a set of local maps connected by arcs labelled with relative location between the maps. As compared to previous approaches it maintains loop consistency when calculating the optimal estimate at

global level. Ni et al. [88] presented an exact submapping approach within a smoothing and mapping framework, and propose to cache the factorization of the submaps to speed-up computation. Grisetti et al. [50] propose hierarchical updates on the map: whenever an observation is acquired, the highest level of the hierarchy is modified and only the areas which are substantially modified are changed at lower levels. Ni and Dellaert [89] extended their previous approach to multiple levels and used nested dissection to minimize the dependence between two subtrees. Grisetti et al. [51] proposed a robust optimization approach using solution of submaps to provide good initial estimate for global alignment. Condensed measurements computed from partial solutions have large convergence basin. Zhao et al. [140] present a approximation strategy for large scale SLAM by solving a sequence of submaps and joining them in a divide and conquer manner using linear least squares. Suger et al. [116] present an approximate SLAM approach based on hierarchical decomposition to reduce the memory consumption required to solve the complete graph.

While Gaussian elimination has become a popular approach it has two major shortcomings. First, the marginals to be exchanged among the robots are dense, and the communication cost is quadratic in the number of separators. This motivated the use of sparsification techniques to reduce the communication cost [94]. The second reason is that Gaussian elimination is performed on a linearized version of the problem, hence these approaches require good linearization points and complex bookkeeping to ensure consistency of the linearization points across the robots [29]. The need of a linearization point also characterizes gradient-based techniques [68].

Related Work in Other Communities. Distributed position and orientation estimation is a fertile research area in other communities, including sensor networks, computer vision, and multi agent systems. In these works, the goal is to estimate the state (e.g. position or orientation) of an agent (e.g., a sensor or a camera) from relative measurements among the agents. A large body of literature deals with distributed localization from distance measurements, see [3, 16, 111, 135] and the references therein. The case of position estimation

from linear measurements is considered in [12, 13, 105, 21, 126, 44]; the related problem of *centroid estimation* is tackled in [6]. Distributed rotation estimation has been studied in the context of attitude synchronization [125, 52, 93], camera network calibration [132, 130], sensor network localization [96], and distributed consensus on manifolds [109, 131].

2.2 RGB-D Mapping

One of the earliest systems that performed real-time 3D model reconstruction using structured light sensor was proposed by Rusinkiewicz et al. [104]. In their system, user rotates the object by hand and sees a continuously updated model as the object is scanned. Their system used a real-time variant of ICP algorithm based of point-plane metric and projective data association to perform live reconstruction of small models. Subsequently, Weise et al. [136] improved on their system and performed high quality object reconstruction using fix time of flight (ToF) sensor and moving object. Cui et al. [28] showed reconstruction results using a moving ToF sensor.

Henry et al. [53] introduced *an RGB-D mapping* framework to generate dense 3D models of large indoor environments. Live sensor motion between consecutive frames was estimated using ICP between depth scans which was initialized by RGB feature matching. Pose graph optimization over the frames was used to generate globally consistent maps. Hornung et al.[56] proposed an octree based occupancy mapping framework which uses octree compression techniques to scale to large areas. Newcombe et al. [87] proposed KinectFusion system which used the reconstructed implicit surface model to track the current live frame instead of estimating the current pose using frame-to-frame ICP. The model is then updated by fusing the current depth information with the reconstructed model. However, the KinectFusion system cannot perform loop closure since it does not maintain a pose graph to check for loop closures and was designed for real time performance in small workspaces useful for augmented reality. Another issue with these implicit volumetric methods is their lack of scalability due to their reliance on uniform grid which

is limited by the amount of GPU memory. Whelan et al. [137, 2, 1] addressed the scalability and loop closure issue using a rolling cyclical buffer for moving the voxel grid in and out of GPU memory and included loop closure using place recognition and pose graph optimization. Zhou et al. [143] used a GPU-based octree to perform Poisson surface reconstruction on 300K vertices at interactive rates. Niessner et al. [90] proposed a voxel hashing technique that compresses space and allows for real-time access and updates for implicit surface data, without the need for regular or hierarchical grid structure.

Other SLAM systems include DVO-SLAM proposed by Kerl et al. [64, 113, 65]. DVO-SLAM estimates frame-to-frame motion and performs no explicit map reconstruction and uses pose graph optimization to estimate consistent reconstruction. Meilland and Comport use fused keyframes of mapped environment to predict current frame's pose and they also use pose graph optimization to close large loops. MRSMap by Stuckler and Behnke registers octree encoded surfel maps together for pose estimation. After pose graph optimisation the final map is created by merging key surfel views [114]. ORB-SLAM system proposed by Raul Mur-Artal generates sparse map by associating and triangulating ORB features and use the reconstructed map for pose estimation. The system works in real time on CPU in a wide variety of environments [84, 85].

Recent works have focussed on generating dense consistent reconstructions at scale and in real time. Choi et al. [22] used line processes to robustify the optimization to erroneous data association. Whelan et al. [138] build a surfel based map of the environment. This is a map-centric approach that forget poses and performs loop closing applying a non-rigid deformation to the map, instead of a standard pose-graph optimization. The detailed reconstruction and localization accuracy of this system is impressive, but the current implementation is limited to room-size maps as the complexity scales with the number of surfels in the map. Dai et al. [31] propose a novel online real-time 3D reconstruction approach that provides robust tracking and implicitly solves the loop closure problem by globally optimizing the trajectory for every captured frame.

2.3 Object-level Mapping

Semantic mapping using high-level object-based representation has gathered a large amount of interest from the robotics community. Kuipers et al. [72] model the environment as a spatial semantic hierarchy, where each level expresses states of partial knowledge corresponding to different level of representations. Nieto et al. [91] employed automated recognition and classification of spaces into separate semantic (Gaussian) regions and used the spatial information for the generation of a topological map of the environment. Ranganathan and Dellaert [98] present a 3D generative model for representing places using objects. The object models are learned in a supervised manner. Civera et al. [26] propose a semantic SLAM algorithm that annotates the low-level 3D point based maps with precomputed object models. Rogers et al. [101] recognize door signs and read their text labels (e.g., room numbers) which are used as landmarks in SLAM. Trevor et al. [127] use planar surfaces corresponding to walls and tables as landmarks in a mapping system. Bao et al. [11] model semantic structure from motion as a joint inference problem where they jointly recognize and estimate the location of high-level semantic scene components such as regions and objects in 3D. SLAM++, proposed by Moreno et al. [108], train domain-specific object detectors corresponding to repeated objects like tables and chairs. The learned detectors are integrated inside the SLAM framework to recognize and track those objects resulting in a semantic map. Similarly, Kim et al. [67] use learned object models to reconstruct dense 3D models from single scan of the indoor scene. Choudhary et al. [25] proposed an approach for online object discovery and object modeling, and extend a SLAM system to utilize these discovered and modeled objects as landmarks to help localize the robot in an online manner. Pillai et al. [95] develop a SLAM-aware object recognition system which result in a considerably stronger recognition performance as compared to related techniques. Lopez et al. [47] present a real-time monocular object-based SLAM using a large database of 500 3D objects and show exploiting object rigidity both improve

the map and find its real scale. Another body of related work is in the area of dense semantic mapping where the goal is to categorize each voxel or 3D point with a category label. Tateno et al. [122] proposed a framework for simultaneous reconstruction, segmentation and recognition, which incrementally segments and recognizes full 3D objects out of a KinectFusion reconstruction, and yields robust object recognition and 3D pose estimation. Mu et al. [83] solves the object level data association problem using a novel nonparametric pose graph that models data association and SLAM in a single framework. They also developed an algorithm that alternates between inferring data association and performing SLAM. Sunderhauf et al. [119] proposed an approach to model individual object entities as point clouds using SLAM and therefore generating an enriched object map of the environment.

2.4 Dense Semantic Mapping

Dense semantic mapping is also related to our approach. Related work in dense semantic mapping include [92, 69, 97, 41, 73, 134, 133, 82] and the references therein. Collet et al. used domain knowledge in the form of metadata and use it as constraints to generate object candidates [27]. Using RGBD sensor, Koppula et al. [69] used graphical models capturing various image feature and contextual relationship to semantically label the point cloud with object classes and used that on a mobile robot for finding objects in a large cluttered room. Karpathy et al. [62] decompose a scene into candidate segments and ranks them according to their objectness properties. Valentin et al. [133] used a CRF and a perpixel labelling from a variant of TextonBoost to reconstruct semantic maps of both indoor and outdoor scenes. McCormac et al. [82] combine Convolutional Neural Networks and ElasticFusion [138] to fuse semantic predictions from multiple view points into a consistent surfel map. Hoiem and Savarese [54] did a survey of additional recent work in the area of 3D scene understanding and 3D object recognition.

Convolutional Neural Networks for Semantic Mapping. Conventional machine learning algorithms required a lot of engineering and domain expertise to design a feature extractor which served as an input to the learning subsystem. Deep learning methods are representation-learning methods which avoid these issues by learning nested hierarchy of representation with each levels learning a more abstract representation than the level below it. It allows the computer to build complex concepts out of simpler concepts [14].

Convolutional neural networks (CNNs) [77] are a special kind of multilayer neural network representation suited for processing structured data like images or videos which has grid like topology. Convolutional layer is used in place of general matrix multiplication in at least one of the layers. Convolutional layer consists of various filters which are convolved with feature map of the previous layer. This layer helps in detecting highly correlated local patches. CNNs use the same weight of a filter (shared weights) when convolved with different parts of an image. This ensures translational invariance of an object in an image. Shared weights can therefore help detect the same pattern in different parts of the image.

CNNs exploit the property that many natural signals are compositional hierarchies, in which higher-level features are obtained by composing lower level ones. In images, group of pixels for edges (detected by first convolutional layer), local combination of edges form motifs, motifs assemble into parts, group of parts from objects and various objects describe a scene. Multiple convolutional, pooling, activation layers stacked on top of each other in CNN are well suited to learn such hierarchy of concepts.

Recently CNN has received considerable success in general vision tasks like image classification [121, 71], object detection [49, 48] and scene recognition [141, 142]. Open-source packages like Caffe [60] are available to train/test state-of-art networks without much effort.

In context of semantic SLAM, convolutional neural networks have been used for place recognition using CovNet features [120, 117, 118] using networks trained on ImageNet dataset [36] or Places dataset [142]. Per-image place categorization is embedded into

Bayesian filter framework to perform semantic mapping [117]. Similarly CNNs are also used for RGB image localization which learns the mapping from image to pose, given the SLAM/SfM output as the training data [63]. McCormac et al. [82] combine Convolutional Neural Networks and ElasticFusion [138] to fuse semantic predictions from multiple view points into a consistent surfel map.

CHAPTER III

DISTRIBUTED POSE GRAPH OPTIMIZATION WITH PRIVACY AND COMMUNICATION CONSTRAINTS

3.1 Introduction

We consider a distributed ML trajectory estimation problem in which the robots have to collaboratively estimate their trajectories while minimizing the amount of exchanged information. We focus on a fully 3D case, as this setup is of great interest in many robotics applications (e.g., navigation on uneven terrain, UAVs). We also consider a fully distributed setup, in which the robots can communicate and acquire relative measurements only during rendezvous events. Our approach can be understood as a distributed implementation of the *chordal initialization* discussed in [20]. The chordal initialization [20] consists in approximating the ML trajectory estimate by solving two quadratic optimization subproblems. In particular, we investigate distributed implementations of the Jacobi Over-Relaxation and the Successive Over-Relaxation. These distributed solvers imply a communication burden that is linear in the number of rendezvous among the robots. Moreover, they do not rely on the availability of an accurate initial guess as in related work (see Chap 2). In Section 3.4, we discuss conditions under which the distributed algorithms converge to the same estimate of the chordal initialization [20], which has been extensively shown to be accurate and resilient to measurement noise.

We perform extensive experimental evaluation including realistic simulations in Gazebo and field tests in a military facility. This contribution is presented in Section 3.6. The experiments demonstrate that one of the proposed algorithms, namely the *Distributed Gauss-Seidel method*, provides accurate trajectory estimates, reduces communication overhead, scales to large teams, has any-time flavour and is robust to noise.

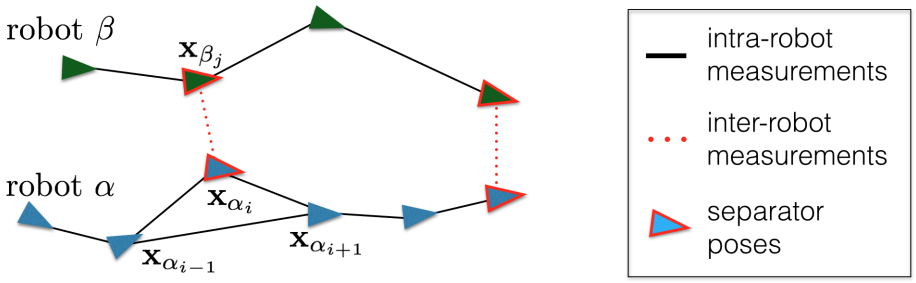


Figure 1: An instance of multi robot trajectory estimation: two robots (α in blue, and β in dark green) traverse an unknown environment, collecting intra-robot measurements (solid black lines). During rendezvous, each robot can observe the pose of the other robot (dotted red lines). These are called inter-robot measurements and relate two *separators* (e.g., $\boldsymbol{x}_{\alpha_i}, \boldsymbol{x}_{\beta_j}$). The goal of the two robots is to compute the ML estimate of their trajectories.

The first contribution of this thesis is to devise distributed algorithms that the robots can implement to reach consensus on a globally optimal trajectory estimate using minimal communication. Section 3.2 introduces the mathematical notation and formalizes the problem. Section 3.3 presents a centralized algorithm, while Section 3.4 presents the corresponding distributed implementations.

3.2 Problem Formulation: Distributed Pose Graph Optimization

We consider a multi robot system and we denote each robot with a Greek letter, such that the set of robots is $\boldsymbol{\Omega} = \{\alpha, \beta, \gamma, \ldots\}$. The goal of each robot is to estimate its own trajectory using the available measurements, and leveraging occasional communication with other robots. The trajectory estimation problem and the nature of the available measurements are made formal in the rest of this section.

We model each trajectory as a finite set of poses (triangles in Fig. 1); the pose assumed by robot α at time i is denoted with $\boldsymbol{x}_{\alpha_i}$ (we use Roman letters to denote time indices). We are interested in a 3D setup, i.e., $\boldsymbol{x}_{\alpha_i} \in \mathrm{SE}(3)$, where $\mathrm{SE}(3)$ is the Special Euclidean group of 3D rigid transformations; when convenient, we write $\boldsymbol{x}_{\alpha_i} = (\boldsymbol{R}_{\alpha_i}, \boldsymbol{t}_{\alpha_i})$, making explicit that each pose includes a rotation $\boldsymbol{R}_{\alpha_i} \in \mathrm{SO}(3)$, and a position $\boldsymbol{t}_{\alpha_i} \in \mathbb{R}^3$. The trajectory of robot α is then denoted as $\boldsymbol{x}_\alpha = [\boldsymbol{x}_{\alpha_1}, \boldsymbol{x}_{\alpha_2}, \ldots]$.

Measurements. We assume that each robot acquires relative pose measurements. In practice these are obtained by post-processing raw sensor data (e.g., scan matching on 3D laser scans). We consider two types of measurements: intra-robot and inter-robot measurements. The *intra-robot measurements* involve the poses of a single robot at different time instants; common examples of intra-robot measurements are odometry measurements (which constrain consecutive robot poses, e.g., $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{x}_{\alpha_{i+1}}$ in Fig. 1) or loop closures (which constrain non-consecutive poses, e.g., $\boldsymbol{x}_{\alpha_{i-1}}$ and $\boldsymbol{x}_{\alpha_{i+1}}$ in Fig. 1). The *inter-robot measurements* are the ones relating the poses of different robots. For instance, during a rendezvous, robot α (whose local time is i), observes a second robot β (whose local time is j) and uses on-board sensors to measure the relative pose of the observed robot in its own reference frame. Therefore, robot α acquires an inter-robot measurement, describing the relative pose between $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{x}_{\beta_j}$ (red links in Fig. 1). We use the term *separators* to refer to the poses involved in an inter-robot measurement.

While our classification of the measurements (inter `vs` intra) is based on the robots involved in the measurement process, all relative measurements can be framed within the same measurement model. Since all measurements correspond to noisy observation of the relative pose between a pair of poses, say $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{x}_{\beta_j}$, a general measurement model is:

$$\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i} \doteq (\bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i}, \bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i}), \quad \text{with:} \begin{cases} \bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i} = (\boldsymbol{R}_{\alpha_i})^{\mathsf{T}} \boldsymbol{R}_{\beta_j} \boldsymbol{R}_{\epsilon} \\ \bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i} = (\boldsymbol{R}_{\alpha_i})^{\mathsf{T}} (\boldsymbol{t}_{\beta_j} - \boldsymbol{t}_{\alpha_i}) + \boldsymbol{t}_{\epsilon} \end{cases} \tag{1}$$

where the relative pose measurement $\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i}$ includes the relative rotation measurements $\bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i}$, which describes the attitude $\boldsymbol{R}_{\beta_j}$ with respect to the reference frame of robot α at time i, "plus" a random rotation $\boldsymbol{R}_{\epsilon}$ (measurement noise), and the relative position measurement $\bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i}$, which describes the position $\boldsymbol{t}_{\beta_j}$ in the reference frame of robot α at time i, plus random noise $\boldsymbol{t}_{\epsilon}$. According to our previous definition, intra robot measurements are in the form $\bar{\boldsymbol{z}}_{\alpha_k}^{\alpha_i}$, for some robot α and for two time instants $i \neq k$; inter-robot measurements, instead, are in the form $\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i}$ for two robots $\alpha \neq \beta$.

In the following, we denote with $\mathcal{E}_I^{\alpha}$ the set of intra-robot measurements for robot α,

while we call $\mathcal{E}_I$ the set of intra-robot measurements for all robots in the team, i.e., $\mathcal{E}_I = \cup_{\alpha \in \mathbf{\Omega}} \mathcal{E}_I^\alpha$. The set of inter-robot measurements involving robot α is denoted with $\mathcal{E}_S^\alpha$ (S is the mnemonic for "separator"). The set of all inter-robot measurements is denoted with $\mathcal{E}_S$. The set of all available measurements is then $\mathcal{E} = \mathcal{E}_I \cup \mathcal{E}_S$. Note that each robot only has access to its own intra and inter-robot measurements $\mathcal{E}_I^\alpha$ and $\mathcal{E}_S^\alpha$.

ML trajectory estimation. Let us collect all robot trajectories in a single (to-be-estimated) set of poses $\boldsymbol{x} = [\boldsymbol{x}_\alpha, \boldsymbol{x}_\beta, \boldsymbol{x}_\gamma, \ldots]$. The ML estimate for $\boldsymbol{x}$ is defined as the maximum of the measurement likelihood:

$$\boldsymbol{x}^\star = \arg\max_{\boldsymbol{x}} \prod_{(\alpha_i, \beta_j) \in \mathcal{E}} \mathcal{L}(\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i} \mid \boldsymbol{x}) \tag{2}$$

where we took the standard assumption of independent measurements. The expression of the likelihood function depends on the distribution of the measurements noise, i.e., $\boldsymbol{R}_\epsilon, \boldsymbol{t}_\epsilon$ in (1). We follow the path of [19] and assume that translation noise is distributed according to a zero-mean Gaussian with information matrix $\omega_t^2 \mathbf{I}_3$, while the rotation noise follows a Von-Mises distribution with concentration parameter ω_R^2.

Under these assumptions, it is possible to demonstrate [19] that the ML estimate $\boldsymbol{x} \doteq \{(\boldsymbol{R}_{\alpha_i}, \boldsymbol{t}_{\alpha_i}), \forall \alpha \in \Omega, \forall i\}$[1] can be computed as solution of the following optimization problem:

$$\min_{\substack{\boldsymbol{t}_{\alpha_i} \in \mathbb{R}^3, \boldsymbol{R}_{\alpha_i} \in \mathrm{SO}(3) \\ \forall \alpha \in \mathbf{\Omega}, \forall i}} \sum_{(\alpha_i, \beta_j) \in \mathcal{E}} \omega_t^2 \left\| \boldsymbol{t}_{\beta_j} - \boldsymbol{t}_{\alpha_i} \boldsymbol{R}_{\alpha_i} \bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i} \right\|^2 + \frac{\omega_R^2}{2} \left\| \boldsymbol{R}_{\beta_j} - \boldsymbol{R}_{\alpha_i} \bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i} \right\|_{\mathtt{F}}^2$$

The peculiarity of (3) is the use of the *chordal distance* $\|\boldsymbol{R}_{\beta_j} - \boldsymbol{R}_{\alpha_i} \bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i}\|_{\mathtt{F}}$ to quantify rotation errors, while the majority of related works in robotics uses the *geodesic distance*, see [20] for details.

A centralized approach to solve the multi robot pose graph optimization problem (3) works as follows. A robot collects all measurements $\mathcal{E}$. Then, the optimization problem (3) is solved using iterative optimization on manifold [35], fast approximations [20], or convex relaxations [102].

[1] In order to simplify notations, $\forall i$ refers to the list of time indices for each robot.

In this thesis we consider the more interesting case in which it is not possible to collect all measurements at a centralized estimator, and the problem has to be solved in a distributed fashion. More formally, the problem we solve is the following.

Problem 1 (Distributed Trajectory Estimation) *Design an algorithm that each robot α can execute during a rendezvous with a subset of other robots $\Omega_r \subseteq \Omega \setminus \{\alpha\}$, and that*

- *takes as input: (i) the intra-robot measurements $\mathcal{E}_I^\alpha$ and (ii) the subset of inter-robot measurements $\mathcal{E}_S^\alpha$, (iii) partial estimates of the trajectory of robots $\beta \in \Omega_r$;*
- *returns as output: the ML estimate $\boldsymbol{x}_\alpha^\star$, which is such that $\boldsymbol{x}^\star = [\boldsymbol{x}_\alpha^\star, \boldsymbol{x}_\beta^\star, \boldsymbol{x}_\gamma^\star, \ldots]$ is a minimizer of* (3).

While the measurements $\mathcal{E}_I^\alpha$ and $\mathcal{E}_S^\alpha$ are known by robot α, gathering the estimates from robots $\beta \in \Omega_r$ requires communication, hence we want our distributed algorithm to exchange a very small portion of the trajectory estimates.

The next sections present our solution to Problem 1. To help readability, we start with a centralized description of the approach, which is an adaptation of the chordal initialization of [20] to the multi robot case. Then we tailor the discussion to the distributed setup in Section 3.4.

3.3 Two-Stage Pose Graph Optimization: Centralized Description

The present work is based on two key observations. The first one is that the optimization problem (3) has a quadratic objective; what makes (3) hard is the presence of non-convex constraints, i.e., $\boldsymbol{R}_{\alpha_i} \in \mathrm{SO}(3)$. Therefore, as already proposed in [20] (for the single robot, centralized case), we use a two-stage approach: we first solve a relaxed version of (3) and get an estimate for the rotations $\boldsymbol{R}_{\alpha_i}$ of all robots, and then we recover the full poses and top-off the result with a Gauss-Newton (GN) iteration. The second key observation is that each of the two stages can be solved in distributed fashion, exploiting existing distributed linear system solvers. In the rest of this section we review the two-stage approach of [20], while we discuss the use of distributed solvers in Section 3.4.

The two-stage approach of [20] first solves for the unknown rotations, and then recovers the full poses via a single GN iteration. The two stages are detailed in the following.

Stage 1: rotation initialization via relaxation and projection. The first stage computes a good estimate of the rotations of all robots by solving the following rotation subproblem:

$$\min_{\substack{\boldsymbol{R}_{\alpha_i}\in\mathrm{SO}(3)\\ \forall\alpha\in\boldsymbol{\Omega},\forall i}} \sum_{(\alpha_i,\beta_j)\in\mathcal{E}} \omega_R^2 \left\| \boldsymbol{R}_{\beta_j} - \boldsymbol{R}_{\alpha_i} \bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i} \right\|_{\mathrm{F}}^2 \tag{3}$$

which amounts to estimating the rotations of all robots in the team by considering only the relative rotation measurements (the second summand in (3)).

While problem (3) is nonconvex (due to the nonconvex constraints $\boldsymbol{R}_{\alpha_i} \in \mathrm{SO}(3)$), many algorithms to approximate its solution are available in literature. Here we use the approach proposed in [81] and reviewed in [20]. The approach first solves the quadratic relaxation obtained by dropping the constraints $\boldsymbol{R}_{\alpha_i} \in \mathrm{SO}(3)$, and then projects the relaxed solution to $\mathrm{SO}(3)$. In formulas, the quadratic relaxation is:

$$\min_{\boldsymbol{R}_{\alpha_i},\forall\alpha\in\boldsymbol{\Omega},\forall i} \sum_{(\alpha_i,\beta_j)\in\mathcal{E}} \omega_R^2 \left\| \boldsymbol{R}_{\beta_j} - \boldsymbol{R}_{\alpha_i} \bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i} \right\|_{\mathrm{F}}^2 \tag{4}$$

which simply rewrites (3) without the constraints. Since (4) is quadratic in the unknown rotations $\boldsymbol{R}_{\alpha_i}, \forall\alpha \in \boldsymbol{\Omega}, \forall i$, we can rewrite it as:

$$\min_{\boldsymbol{r}} \|\boldsymbol{A}_r \boldsymbol{r} - \boldsymbol{b}_r\|^2 \tag{5}$$

where we stacked all the entries of the unknown rotation matrices $\boldsymbol{R}_{\alpha_i}, \forall\alpha \in \boldsymbol{\Omega}, \forall i$ into a single vector $\boldsymbol{r}$, and we built the (known) matrix $\boldsymbol{A}_r$ and (known) vector $\boldsymbol{b}_r$ accordingly (the presence of a nonzero vector $\boldsymbol{b}_r$ follows from setting one of the rotations to be the reference frame, e.g., $\boldsymbol{R}_{\alpha_1} = \mathbf{I}_3$).

Since (4) is a linear least-squares problem, its solution can be found by solving the normal equations:

$$(\boldsymbol{A}_r^{\mathsf{T}} \boldsymbol{A}_r)\boldsymbol{r} = \boldsymbol{A}_r^{\mathsf{T}} \boldsymbol{b}_r \tag{6}$$

Let us denote with $\breve{\boldsymbol{r}}$ the solution of (6). Rewriting $\breve{\boldsymbol{r}}$ in matrix form, we obtain the matrices $\breve{\boldsymbol{R}}_{\alpha_i}$, $\forall \alpha \in \boldsymbol{\Omega}, \forall i$. Since these rotations were obtained from a relaxation of (3), they are not guaranteed to satisfy the constraints $\boldsymbol{R}_{\alpha_i} \in \mathrm{SO}(3)$; therefore the approach [81] projects them to $\mathrm{SO}(3)$, and gets the rotation estimate $\hat{\boldsymbol{R}}_{\alpha_i} = \texttt{project}(\breve{\boldsymbol{R}}_{\alpha_i})$, $\forall \alpha \in \boldsymbol{\Omega}, \forall i$. The projection only requires to perform an SVD of $\breve{\boldsymbol{R}}_{\alpha_i}$ and can be performed independently for each rotation [20].

Stage 2: full pose recovery via single GN iteration. In the previous stage we obtained an estimate for the rotations $\hat{\boldsymbol{R}}_{\alpha_i}, \forall \alpha \in \boldsymbol{\Omega}, \forall i$. In this stage we use this estimate to reparametrize problem (3). In particular, we rewrite each unknown rotation $\boldsymbol{R}_{\alpha_i}$ as the known estimate $\hat{\boldsymbol{R}}_{\alpha_i}$ "plus" an unknown perturbation; in formulas, we rewrite each rotation as $\boldsymbol{R}_{\alpha_i} = \hat{\boldsymbol{R}}_{\alpha_i} \mathrm{Exp}\left(\boldsymbol{\theta}_{\alpha_i}\right)$, where $\mathrm{Exp}\left(\cdot\right)$ is the exponential map for $\mathrm{SO}(3)$, and $\boldsymbol{\theta}_{\alpha_i} \in \mathbb{R}^3$ (this is our new parametrization for the rotations). With this parametrization, eq. (3) becomes:

$$\begin{aligned} \min_{\substack{\boldsymbol{t}_{\alpha_i}, \boldsymbol{\theta}_{\alpha_i} \\ \forall \alpha \in \boldsymbol{\Omega}, \forall i}} & \sum_{(\alpha_i, \beta_j) \in \mathcal{E}} \omega_t^2 \left\| \boldsymbol{t}_{\beta_j} - \boldsymbol{t}_{\alpha_i} - \hat{\boldsymbol{R}}_{\alpha_i} \mathrm{Exp}\left(\boldsymbol{\theta}_{\alpha_i}\right) \bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i} \right\|^2 \\ & + \frac{\omega_R^2}{2} \left\| \hat{\boldsymbol{R}}_{\beta_j} \mathrm{Exp}\left(\boldsymbol{\theta}_{\beta_j}\right) - \hat{\boldsymbol{R}}_{\alpha_i} \mathrm{Exp}\left(\boldsymbol{\theta}_{\alpha_i}\right) \bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i} \right\|_{\mathrm{F}}^2 \end{aligned} \tag{7}$$

Note that the reparametrization allowed to drop the constraints (we are now trying to estimate vectors in $\mathbb{R}^3$), but moved the nonconvexity to the objective ($\mathrm{Exp}\left(\cdot\right)$ is nonlinear in its argument). In order to solve (7), we take a quadratic approximation of the cost function. For this purpose we use the following first-order approximation of the exponential map:

$$\mathrm{Exp}\left(\boldsymbol{\theta}_{\alpha_i}\right) \simeq \mathbf{I}_3 + \boldsymbol{S}(\boldsymbol{\theta}_{\alpha_i}) \tag{8}$$

where $\boldsymbol{S}(\boldsymbol{\theta}_{\alpha_i})$ is a skew symmetric matrix whose entries are defined by the vector $\boldsymbol{\theta}_{\alpha_i}$. Substituting (8) into (7) we get the desired quadratic approximation:

$$\begin{aligned} \min_{\substack{\boldsymbol{t}_{\alpha_i}, \boldsymbol{\theta}_{\alpha_i} \in \mathbb{R}^3 \\ \forall \alpha \in \boldsymbol{\Omega}, \forall i}} & \sum_{(\alpha_i, \beta_j) \in \mathcal{E}} \omega_t^2 \left\| \boldsymbol{t}_{\beta_j} - \boldsymbol{t}_{\alpha_i} - \hat{\boldsymbol{R}}_{\alpha_i} \bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i} - \hat{\boldsymbol{R}}_{\alpha_i} \boldsymbol{S}(\boldsymbol{\theta}_{\alpha_i}) \bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i} \right\|^2 \\ & + \frac{\omega_R^2}{2} \left\| \hat{\boldsymbol{R}}_{\beta_j} - \hat{\boldsymbol{R}}_{\alpha_i} \bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i} + \hat{\boldsymbol{R}}_{\beta_j} \boldsymbol{S}(\boldsymbol{\theta}_{\beta_j}) - \hat{\boldsymbol{R}}_{\alpha_i} \boldsymbol{S}(\boldsymbol{\theta}_{\alpha_i}) \bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i} \right\|_{\mathrm{F}}^2 \end{aligned} \tag{9}$$

Rearranging the unknown $\boldsymbol{t}_{\alpha_i}, \boldsymbol{\theta}_{\alpha_i}$ of all robots into a single vector $\boldsymbol{p}$, we rewrite (9) as a linear least-squares problem:

$$\min_{\boldsymbol{p}} \ \|\boldsymbol{A}_p \, \boldsymbol{p} - \boldsymbol{b}_p\|^2 \tag{10}$$

whose solution can be found by solving the linear system:

$$(\boldsymbol{A}_p^{\mathsf{T}} \boldsymbol{A}_p)\boldsymbol{p} = \boldsymbol{A}_p^{\mathsf{T}} \boldsymbol{b}_p \tag{11}$$

From the solution of (11) we can build our trajectory estimate: the entries of $\boldsymbol{p}$ directly define the positions $\boldsymbol{t}_{\alpha_i}, \forall \alpha \in \boldsymbol{\Omega}, \forall i$; moreover, $\boldsymbol{p}$ includes the rotational corrections $\boldsymbol{\theta}_{\alpha_i}$, from which we get our rotation estimate as: $\boldsymbol{R}_{\alpha_i} = \hat{\boldsymbol{R}}_{\alpha_i} \mathrm{Exp}\,(\boldsymbol{\theta}_{\alpha_i})$.

Remark 1 (Advantage of Centralized Two-Stage Approach) *The approach reviewed in this section has three advantages. First, as shown in [20], in common problem instances (i.e., for reasonable levels of measurement noise) it returns a solution that is very close to the ML estimate. Second, the approach only requires to solve two linear systems (the cost of projecting the rotations is negligible), hence it is computationally efficient. Finally, the approach does not require an initial guess, therefore, it is able to converge even when the initial trajectory estimate is inaccurate (in those instances, iterative optimization tends to fail [20]) or is unavailable.* ■

3.4 Distributed Pose Graph Optimization

In this section we show that the two-stage approach described in Section 3.3 can be implemented in a distributed fashion. Since the approach only requires solving two linear systems, every distributed linear system solver can be used as workhorse to split the computation among the robots. For instance, one could adapt the Gaussian elimination approach of [30] to solve the linear systems (6) and (11). In this section we propose alternative approaches, based on the Distributed Jacobi Over-Relaxation and the Distributed Successive Over-Relaxation algorithms, and we discuss their advantages.

In both (6) and (11) we need to solve a linear system where the unknown vector can be partitioned into subvectors, such that each subvector contains the variables associated to a single robot in the team. For instance, we can partition the vector $\boldsymbol{r}$ in (6), as $\boldsymbol{r} = [\boldsymbol{r}_\alpha, \boldsymbol{r}_\beta, \ldots]$, such that $\boldsymbol{r}_\alpha$ describes the rotations of robot α. Similarly, we can partition $\boldsymbol{p} = [\boldsymbol{p}_\alpha, \boldsymbol{p}_\beta, \ldots]$ in (11), such that $\boldsymbol{p}_\alpha$ describes the trajectory of robot α. Therefore, (6) and (11) can be framed in the general form:

$$\boldsymbol{H}\boldsymbol{y} = \boldsymbol{g} \Leftrightarrow \begin{bmatrix} \boldsymbol{H}_{\alpha\alpha} & \boldsymbol{H}_{\alpha\beta} & \ldots \\ \boldsymbol{H}_{\beta\alpha} & \boldsymbol{H}_{\beta\beta} & \ldots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_\alpha \\ \boldsymbol{y}_\beta \\ \vdots \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}_\alpha \\ \boldsymbol{g}_\beta \\ \vdots \end{bmatrix} \tag{12}$$

where we want to compute the vector $\boldsymbol{y} = [\boldsymbol{y}_\alpha, \boldsymbol{y}_\beta, \ldots]$ given the (known) block matrix $\boldsymbol{H}$ and the (known) block vector $\boldsymbol{g}$; on the right of (12) we partitioned the square matrix $\boldsymbol{H}$ and the vector $\boldsymbol{g}$ according to the block-structure of $\boldsymbol{y}$.

In order to introduce the distributed algorithms, we first observe that the linear system (12) can be rewritten as:

$$\sum_{\delta \in \boldsymbol{\Omega}} \boldsymbol{H}_{\alpha\delta}\boldsymbol{y}_\delta = \boldsymbol{g}_\alpha \qquad \forall \alpha \in \boldsymbol{\Omega}$$

Taking the contribution of $\boldsymbol{y}_\alpha$ out of the sum, we get:

$$\boldsymbol{H}_{\alpha\alpha}\boldsymbol{y}_\alpha = -\sum_{\delta \in \boldsymbol{\Omega}\setminus\{\alpha\}} \boldsymbol{H}_{\alpha\delta}\boldsymbol{y}_\delta + \boldsymbol{g}_\alpha \qquad \forall \alpha \in \boldsymbol{\Omega} \tag{13}$$

The set of equations (13) is the same as the original system (12), but clearly exposes the contribution of the variables associated to each robot. The equations (13) constitute the basis for the *Successive Over-Relaxation* (SOR) and the *Jacobi Over-Relaxation* (JOR) methods that we describe in the following sections.

3.4.1 Distributed Jacobi Over-Relaxation (JOR):

The distributed JOR algorithm [15] starts at an arbitrary initial estimate $\boldsymbol{y}^{(0)} = [\boldsymbol{y}_\alpha^{(0)}, \boldsymbol{y}_\beta^{(0)}, \ldots]$ and solves the linear system (12) by repeating the following iterations:

$$\boldsymbol{y}_\alpha^{(k+1)} = (1-\gamma)\boldsymbol{y}_\alpha^{(k)} + (\gamma)\boldsymbol{H}_{\alpha\alpha}^{-1}\left(-\sum_{\delta \in \boldsymbol{\Omega}\setminus\{\alpha\}} \boldsymbol{H}_{\alpha\delta}\boldsymbol{y}_\delta^{(k)} + \boldsymbol{g}_\alpha\right) \quad \forall \alpha \in \boldsymbol{\Omega} \tag{14}$$

where γ is the *relaxation factor*. Intuitively, at each iteration robot α attempts to solve eq. (13) (the second summand in (14) is the solution of (13) with the estimates of the other robots kept fixed), while remaining close to the previous estimate $\boldsymbol{y}_\alpha^{(k)}$ (first summand in (14)). If the iterations (14) converge to a fixed point, say $\boldsymbol{y}_\alpha \ \forall \alpha$, then the resulting estimate solves the linear system (13) exactly [15, page 131]. To prove this fact we only need to rewrite (14) after convergence:

$$\boldsymbol{y}_\alpha = (1-\gamma)\boldsymbol{y}_\alpha + (\gamma)\boldsymbol{H}_{\alpha\alpha}^{-1}\left(-\sum_{\delta\in\boldsymbol{\Omega}\setminus\{\alpha\}}\boldsymbol{H}_{\alpha\delta}\boldsymbol{y}_\delta + \boldsymbol{g}_\alpha\right)$$

which can be easily seen to be identical to (13).

In our multi robot problem, the distributed JOR algorithm can be understood in a simple way: at each iteration, each robot estimates its own variables ($\boldsymbol{y}_\alpha^{(k+1)}$) by assuming that the variables of the other robots are constant ($\boldsymbol{y}_\delta^{(k)}$); iterating this procedure, the robots reach an agreement on the estimates, and converge to the solution of eq. (12). Using the distributed JOR approach, the robots solve (6) and (11) in a distributed manner. When $\gamma = 1$, the distributed JOR method is also known as the *distributed Jacobi* (DJ) method.

We already mentioned that when the iterations (14) converge, then they return the exact solution of the linear system. So a natural question is: *when do the Jacobi iteration converge?* A general answer is given by the following proposition.

Proposition 2 (Convergence of JOR [15]) *Consider the linear systems* (12) *and define the block diagonal matrix* $\boldsymbol{D} \doteq \mathrm{diag}\,(\boldsymbol{H}_{\alpha\alpha}, \boldsymbol{H}_{\beta\beta}, \ldots)$. *Moreover, define the matrix:*

$$\boldsymbol{M} = (1-\gamma)\mathbf{I} - \gamma\boldsymbol{D}^{-1}(\boldsymbol{H}-\boldsymbol{D}) \tag{15}$$

where $\mathbf{I}$ *is the identity matrix of suitable size. Then, the JOR iterations* (14) *converge from any initial estimate if and only if* $\rho(\boldsymbol{M}) < 1$, *where* $\rho(\cdot)$ *denotes the spectral radius (maximum of absolute value of the eigenvalues) of a matrix.*

The proposition is the same of Proposition 6.1 in [15] (the condition that $\mathbf{I} - \boldsymbol{M}$ is invertible is guaranteed to hold as noted in the footnote on page 144 of [15]).

It is non-trivial to establish whether our linear systems (6) and (11) satisfy the condition of Proposition 2. In the experimental section, we empirically observe that the Jacobi iterations indeed converge whenever $\gamma \leq 1$. For the SOR algorithm, presented in the next section, instead, we can provide stronger theoretical convergence guarantees.

3.4.2 Distributed Successive Over-Relaxation (SOR)

The distributed SOR algorithm [15] starts at an arbitrary initial estimate $\boldsymbol{y}^{(0)} = [\boldsymbol{y}_\alpha^{(0)}, \boldsymbol{y}_\beta^{(0)}, \ldots]$ and, at iteration k, applies the following update rule, for each $\alpha \in \boldsymbol{\Omega}$:

$$\boldsymbol{y}_\alpha^{(k+1)} = (1-\gamma)\boldsymbol{y}_\alpha^{(k)} + (\gamma)\boldsymbol{H}_{\alpha\alpha}^{-1}\left(-\sum_{\delta\in\boldsymbol{\Omega}_\alpha^+}\boldsymbol{H}_{\alpha\delta}\boldsymbol{y}_\delta^{(k+1)} - \sum_{\delta\in\boldsymbol{\Omega}_\alpha^-}\boldsymbol{H}_{\alpha\delta}\boldsymbol{y}_\delta^{(k)} + \boldsymbol{g}_\alpha\right) \tag{16}$$

where γ is the *relaxation factor*, $\boldsymbol{\Omega}_\alpha^+$ is the set of robots that already computed the $(k+1)$-th estimate, while $\boldsymbol{\Omega}_\alpha^-$ is the set of robots that still have to perform the update (16), excluding node α (intuitively: each robot uses the latest estimate). As for the JOR algorithm, by comparing (16) and (13), we see that if the sequence produced by the iterations (16) converges to a fixed point, then such point satisfies (13), and indeed solves the original linear system (12). When $\gamma = 1$, the distributed SOR method is known as the *distributed Gauss-Seidel* (DGS) method.

The following proposition, whose proof trivially follows from [15, Proposition 6.10, p. 154] (and the fact that the involved matrices are positive definite), establishes when the distributed SOR algorithm converges to the desired solution.

Proposition 3 (Convergence of SOR) *The SOR iterations* (16) *applied to the linear systems* (6) *and* (11) *converge to the solution of the corresponding linear system (from any initial estimate) whenever* $\gamma \in (0, 2)$*, while the iterations do no converge to the correct solution whenever* $\gamma \notin (0, 2)$.

According to [15, Proposition 6.10, p. 154], for $\gamma \notin (0, 2)$, the SOR iterations (16) do not converge to the solution of the linear system in general, hence also in practice, we

restrict the choice of γ in the open interval $(0, 2)$. In the experimental section, we show that the choice $\gamma = 1$ ensures the fastest convergence.

3.4.3 Communication Requirements for JOR and SOR

In this section we observe that to execute the JOR and SOR iterations (14)(16), robot α only needs its intra and inter-robot measurements $\mathcal{E}_I^\alpha$ and $\mathcal{E}_S^\alpha$, and an estimate of the separators, involved in the inter-robot measurements in $\mathcal{E}_S^\alpha$. For instance, in the graph of Fig. 2 robot α only needs the estimates of $\boldsymbol{y}_{\beta_1}$ and $\boldsymbol{y}_{\beta_3}$, while does not require any knowledge about the other poses of β.

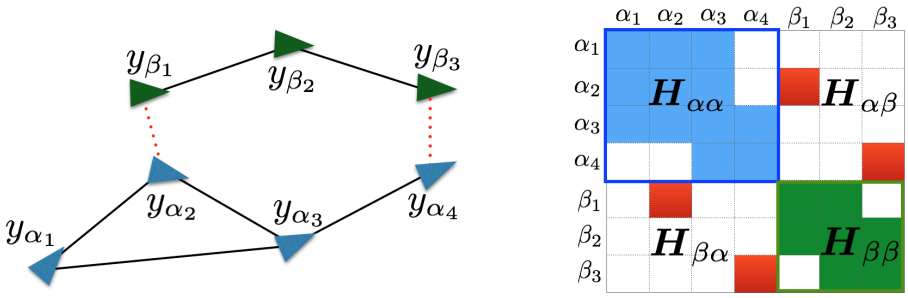


Figure 2: Example: (left) trajectory estimation problem and (right) corresponding block structure of the matrix $\boldsymbol{H}$.

To understand this fact, we note that both (6) and (11) model an estimation problem from pairwise relative measurements. It is well known that the matrix $\boldsymbol{H}$ (sometimes called the *Hessian* [33]) underlying these problems has a block structure defined by the Laplacian matrix of the underlying graph [13]. For instance, Fig. 2 (right) shows the block sparsity of the matrix $\boldsymbol{H}$ describing the graph on the left: off-diagonal block-elements in position (α_i, β_j) are non zero if and only if there is an edge (i.e., a measurement) between α_i and β_j.

By exploiting the block sparsity of $\boldsymbol{H}$, we can further simplify the JOR (14) iterations as:

$$\boldsymbol{y}_\alpha^{(k+1)} = (1-\gamma)\boldsymbol{y}_\alpha^{(k)} + (\gamma)\boldsymbol{H}_{\alpha\alpha}^{-1}\left(-\sum_{(\alpha_i,\delta_j)\in\mathcal{E}_S^\alpha} \boldsymbol{H}_{\alpha_i\delta_j}\boldsymbol{y}_{\delta_j}^{(k)} + \boldsymbol{g}_\alpha\right), \quad \forall\alpha\in\boldsymbol{\Omega} \tag{17}$$

where we simply removed the contributions of the zero blocks from the sum in (14).

Similarly we can simplify the SOR (16) iterations as:

$$\boldsymbol{y}_\alpha^{(k+1)} = (1-\gamma)\boldsymbol{y}_\alpha^{(k)} + (\gamma)\boldsymbol{H}_{\alpha\alpha}^{-1}\left(-\sum_{(\alpha_i,\delta_j)\in\mathcal{E}_S^{\alpha+}} \boldsymbol{H}_{\alpha_i\delta_j}\boldsymbol{y}_{\delta_j}^{(k+1)} - \sum_{(\alpha_i,\delta_j)\in\mathcal{E}_S^{\alpha-}} \boldsymbol{H}_{\alpha_i\delta_j}\boldsymbol{y}_{\delta_j}^{(k)} + \boldsymbol{g}_\alpha \right) \tag{18}$$

where we removed the contributions of the zero blocks from the sum in (16); the sets $\mathcal{E}_S^{\alpha+}$ and $\mathcal{E}_S^{\alpha-}$ satisfy $\mathcal{E}_S^{\alpha+} \cup \mathcal{E}_S^{\alpha-} = \mathcal{E}_S^{\alpha}$, and are such that $\mathcal{E}_S^{\alpha+}$ includes the inter-robot measurements involving robots which already performed the $(k+1)$-th iteration, while $\mathcal{E}_S^{\alpha-}$ is the set of measurements involving robots that have not performed the iteration yet (as before: each robot simply uses its latest estimate).

Eqs. (17) and (18) highlight that the JOR and SOR iterations (at robot α) only require the estimates for poses involved in its inter-robot measurements $\mathcal{E}_S^{\alpha}$. Therefore both JOR and SOR involve almost no "privacy violation": every other robot β in the team does not need to communicate any other information about its own trajectory, but only sends an estimate of its rendezvous poses.

3.4.4 Flagged Initialization

As we will see in the experimental section and according to Proposition 3, the JOR and SOR approaches converge from any initial condition when γ is chosen appropriately. However, starting from a "good" initial condition can reduce the number of iterations to converge, and in turns reduces the communication burden (each iteration (17) or (18) requires the robots to exchange their estimate of the separators).

In this work, we follow the path of [12] and adopt a *flagged initialization*. A flagged initialization scheme only alters the first JOR or SOR iteration as follows. Before the first iteration, all robots are marked as "uninitialized". Robot α performs its iteration (17) or (18) without considering the inter-robot measurements, i.e., eqs. (17)-(18) become $\boldsymbol{y}_\alpha^{(k+1)} = \boldsymbol{H}_{\alpha\alpha}^{-1}\boldsymbol{g}_\alpha$; then the robot α marks itself as "initialized". When the robot β performs its iteration, it includes only the separators from the robots that are initialized; after performing

the JOR or SOR iteration, also β marks itself as initialized. Repeating this procedure, all robots become initialized after performing the first iteration. The following iterations then proceed according to the standard JOR (17) or SOR (18) update. [12] show a significant improvement in convergence using flagged initialization. As discussed in the experiments, flagged initialization is also advantageous in our distributed pose graph optimization problem.

3.5 Implementation Details: Distributed Pose Graph Optimization

Ego-Motion Estimation. Each robot collects 3D scans using Velodyne 32E laser scanner, RGB-D scans using Orbbec Astra sensor, and inertial measurements using IMU and odometry measurements using wheel sensors. Fig. 22 shows the sensor layout on a Jackal robot. Relative pose estimates from all the sensors are fused together using OmniMapper[128] to estimate robot's ego-motion[2]. Fig. 3 shows the overview of the egomotion estimation pipeline.

Specifically, 3D scans from Veldoyne 32E are used to compute relative pose with respect to the previous scan using GICP (generalized iterative closest point [110]). Inertial measurements from IMU are fused with Wheel Odometry measurements to generate IMU corrected odometry estimates. Relative pose estimates using RGB-D scans are computed using ORB-SLAM [85]. Relative pose estimates are then fused using OmniMapper.

Loop Closure. If the RGB-D frame is considered as a key-frame by ORB-SLAM [85], we find loop closure candidates among other key-frames using bag-of-words vector. Thereafter, relative pose estimates against candidate key-frames are computed using RANSAC over 3D-3D correspondences. If more than 12 inliers are found, the relative pose with respect to the candidate key-frame is used to generate a loop closure constraint. The resulting loop closure constraint is then fused along with other constraints using OmniMapper (Fig. 3).

[2] `https://github.com/CognitiveRobotics/omnimapper`

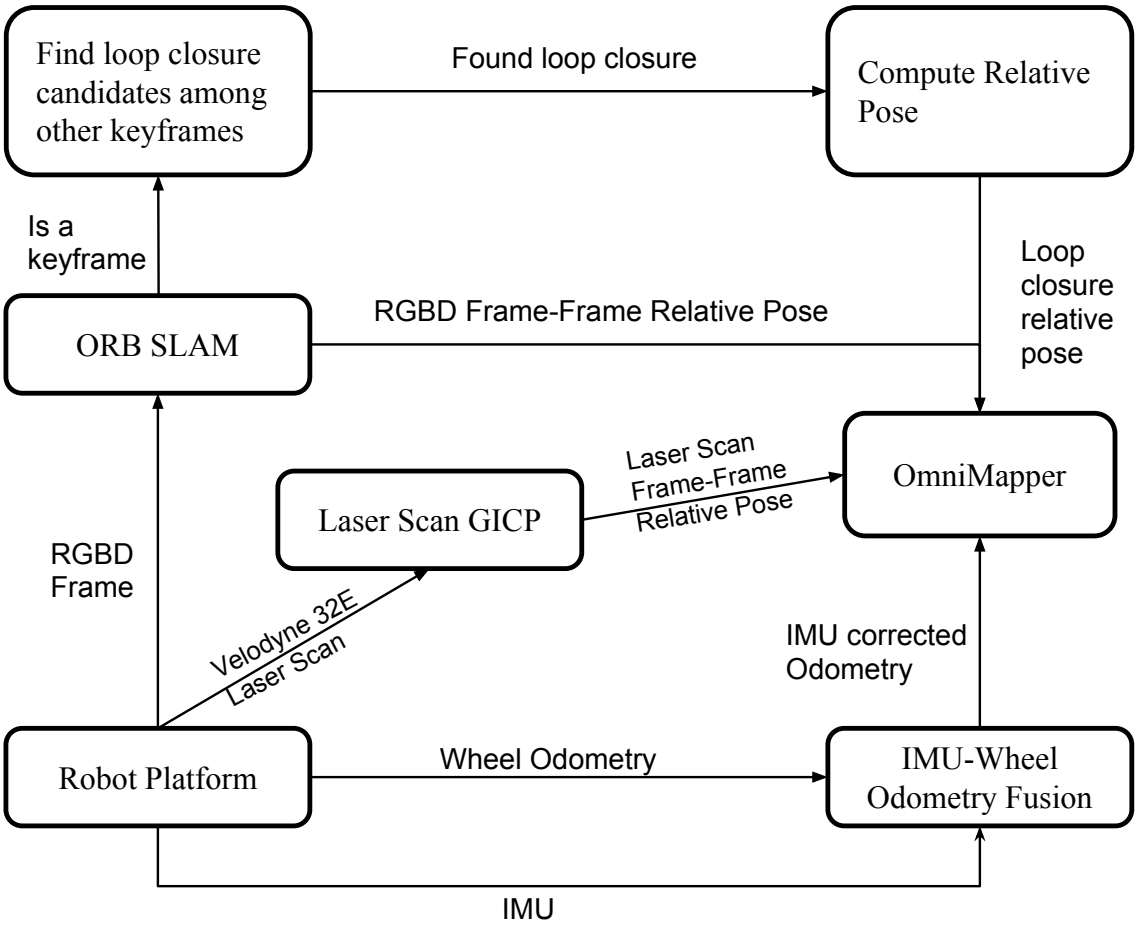


Figure 3: Overview of Ego-Motion estimation front-end

Robot Communication. During a rendezvous between robots α and β, robot α communicates the key-frames to robot β. Robot β finds candidates key-frames among its own key-frames for every received key-frames using bag-of-words vector. Thereafter, relative pose estimates against candidate key-frames are computed using RANSAC over 3D-3D correspondences. If more than 12 inliers are found, the relative pose with respect to the candidate key-frame is used to generate a loop closure constraint. The loop closure constraint is communicated back to the robot α and then optimized using the distributed optimizer. An overview of the pipeline is shown in Fig. 4.

Next we show the experimental evaluation which includes realistic Gazebo simulations and field experiments.

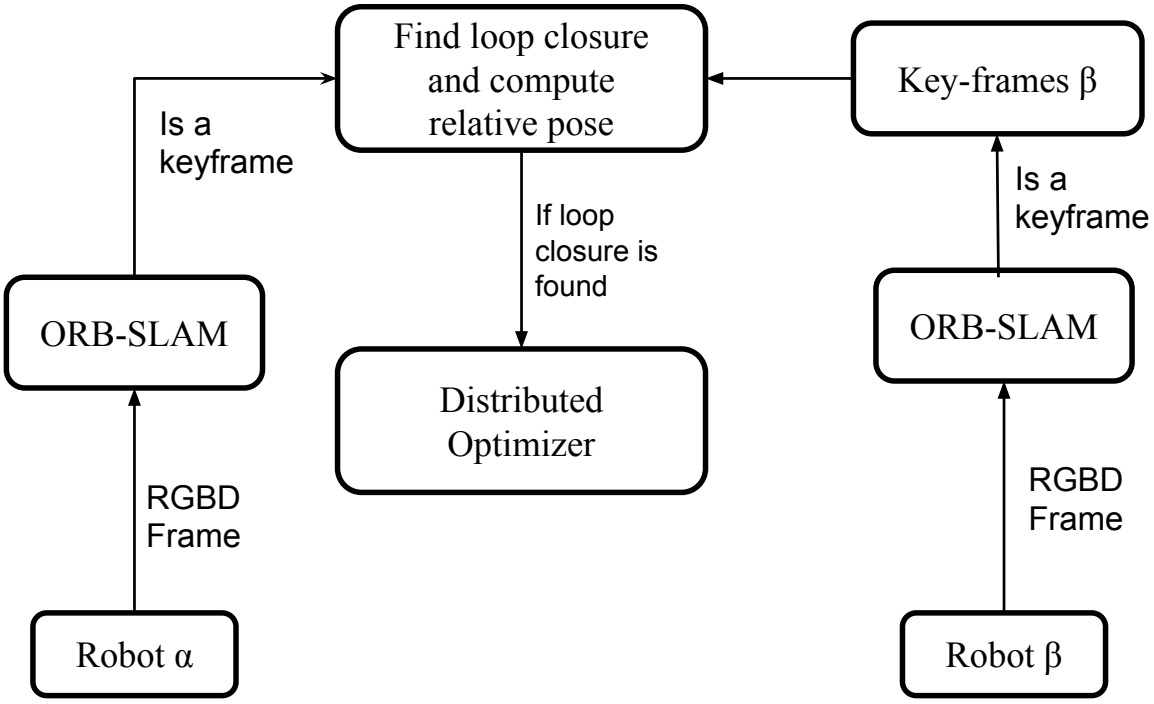


Figure 4: Overview of Distributed Pose Graph Communication

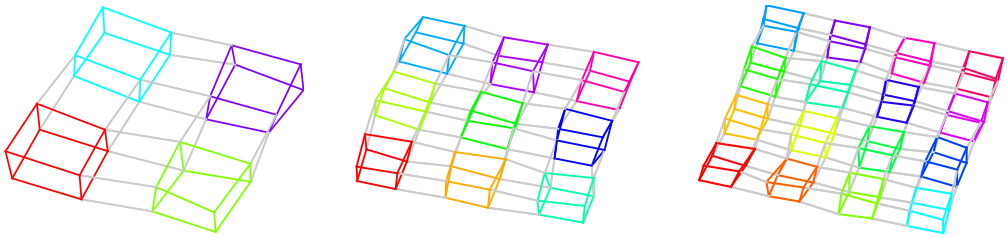
3.6 Experiments

We evaluate the distributed JOR and SOR along with DJ and DGS approaches in large-scale simulations (Section 3.6.1 and field tests (Section 3.6.2 The results demonstrate that (i) the DGS dominates the other algorithms considered in this chapter in terms of convergence speed, (ii) the DGS algorithm is accurate, scalable, and robust to noise, and (iii) the DGS requires less communication than techniques from related work (i.e., DDF-SAM).

3.6.1 Simulation Results: Distributed Pose Graph Optimization

In this section, we characterize the performance of the proposed approaches in terms of convergence, scalability (in the number of robots and separators), and sensitivity to noise.

Simulation setup and performance metrics. For our tests, we created simulation datasets in six different configurations with increasing number of robots: 4, 9, 16, 25, 36 and 49 robots. The robots are arranged in a 3D grid with each robot moving on a cube,



(a) 4 Robots (b) 9 Robots (c) 16 Robots

Figure 5: Simulated 3D datasets with different number of robots. Robots are shown in different colors. Gray links denote inter-robot measurements.

as shown in Fig. 5. When the robots are at contiguous corners, they can communicate (gray links). Unless specified otherwise, we generate measurement noise from a zero-mean Gaussian distribution with standard deviation $\sigma_R = 5^\circ$ for the rotations and $\sigma_t = 0.2$m for the translations. Results are averaged over 10 Monte Carlo runs.

In our problem, JOR or SOR are used to sequentially solve two linear systems, (6) and (11), which return the minimizers of (5) and (10), respectively. Defining, $m_r \doteq \min_{\boldsymbol{r}} \|\boldsymbol{A}_r \boldsymbol{r} - \boldsymbol{b}_r\|^2$, we use the following metric, named the *rotation estimation error*, to quantify the error in solving (6):

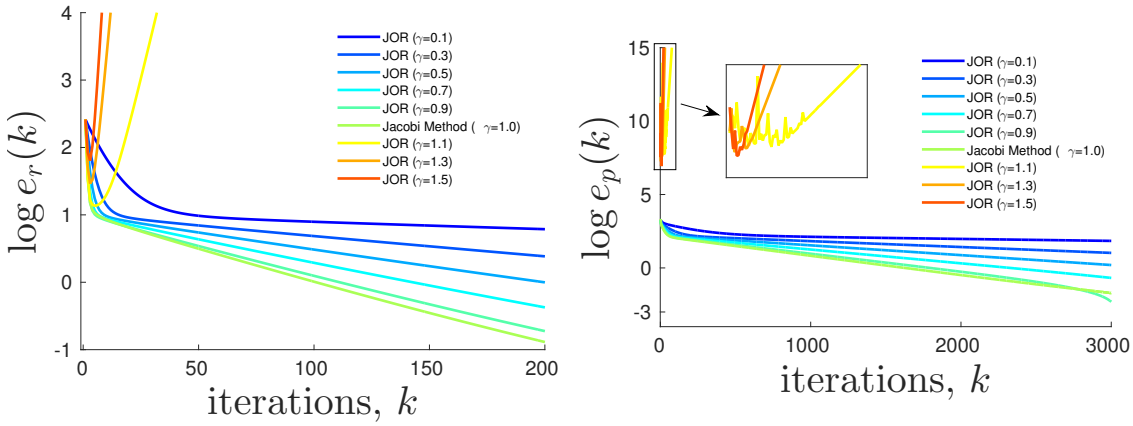
$$e_r(k) = \|\boldsymbol{A}_r \boldsymbol{r}^{(k)} - \boldsymbol{b}_r\|^2 - m_r \tag{19}$$

$e_r(k)$ quantifies how far is the current estimate $\boldsymbol{r}^{(k)}$ (at the k-th iteration) from the minimum of the quadratic cost. Similarly, we define the *pose estimation error* as:

$$e_p(k) = \|\boldsymbol{A}_p \boldsymbol{p}^{(k)} - \boldsymbol{b}_p\|^2 - m_p \tag{20}$$

with $m_p \doteq \min_{\boldsymbol{p}} \ \|\boldsymbol{A}_p \ \boldsymbol{p} - \boldsymbol{b}_p\|^2$. Ideally, we want $e_r(k)$ and $e_p(k)$ to quickly converge to zero for increasing k.

Ultimately, the accuracy of the proposed approach depends on the number of iterations, hence we need to set a stopping condition for the JOR or SOR iterations. We use the following criterion: we stop the iterations if the change in the estimate is sufficiently small. More formally, the iterations stop when $\|\boldsymbol{r}^{(k+1)} - \boldsymbol{r}^{(k)}\| \leq \eta_r$ (similarly, for the second linear system $\|\boldsymbol{p}^{(k+1)} - \boldsymbol{p}^{(k)}\| \leq \eta_p$). We use $\eta_r = \eta_p = 10^{-1}$ as stopping condition unless specified otherwise.

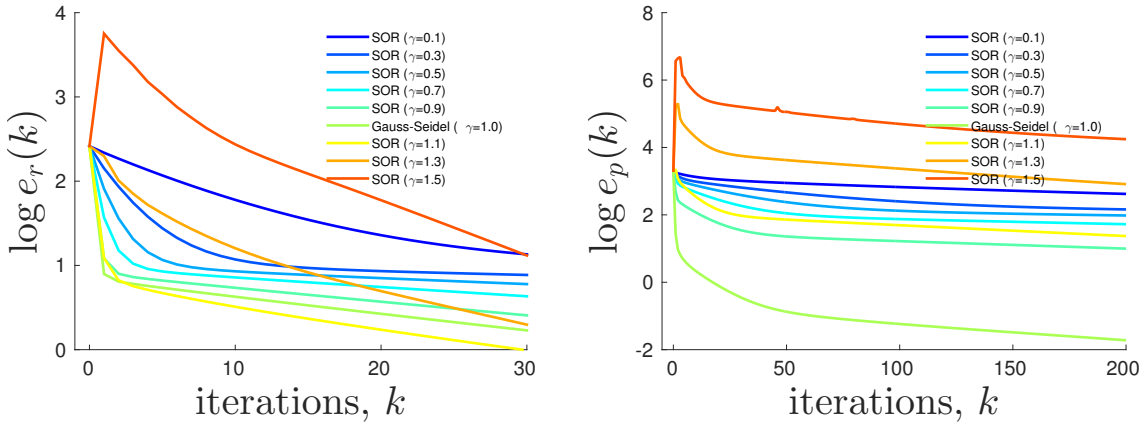


(a) Rotation Estimation (b) Pose Estimation

Figure 6: JOR: convergence of (a) rotation estimation and (b) pose estimation for different values of γ (grid scenario, 49 robots). In the case of pose estimation, the gap between the initial values of $\gamma > 1$ and $\gamma \leq 1$ is due to the bad initialization provided by the rotation estimation for $\gamma > 1$.

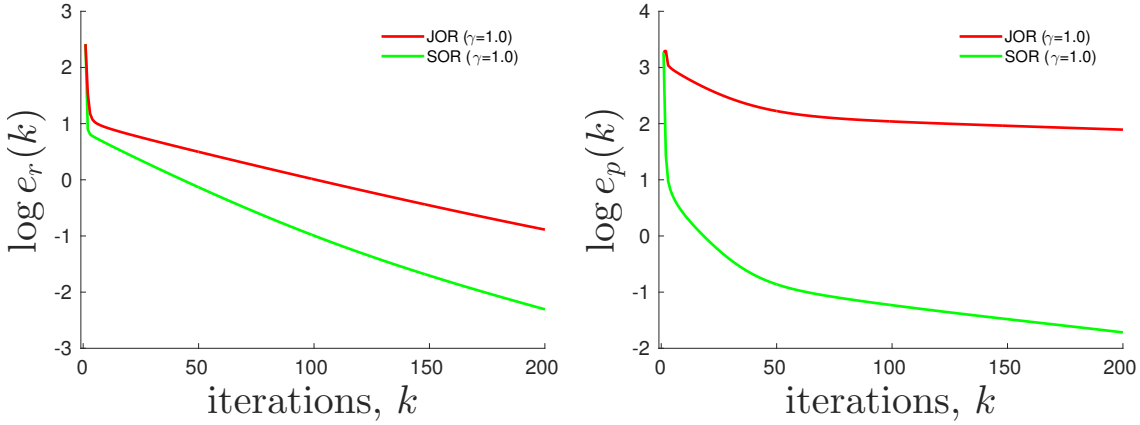
Comparisons among the distributed algorithms. In this section we consider the scenario with 49 robots. We start by studying the convergence properties of the JOR and SOR algorithms in isolation. Then we compare the two algorithms in terms of convergence speed. Fig. 6 shows the rotation and the pose error versus the number of iterations for different choices of the parameter γ for the JOR algorithm. Fig. 6a confirms the result of Proposition 2: JOR applied to the rotation subproblem converges as long as $\gamma \leq 1$. Fig. 6a shows that for any $\gamma > 1$ the estimate diverges, while the critical value $\gamma = 1$ (corresponding to the DJ method) ensures the fastest convergence rate. Fig. 7 shows the rotation and the pose error versus the number of iterations for different choices of the parameter $\gamma \in (0, 2)$ for the SOR algorithm. The figure confirms the result of Proposition 3: the SOR algorithm converges for any choice of $\gamma \in (0, 2)$. Fig. 7a shows that choices of γ close to 1 ensures fast convergence rates, while Fig. 7b established $\gamma = 1$ (corresponding to the DGS method) as the parameter selection with faster convergence. In summary, both JOR and SOR have top performance when $\gamma = 1$. Later in this section we show that $\gamma = 1$ is the best choice independently on the number of robots and the measurement noise.

Let us now compare JOR and SOR in terms of convergence. Fig. 8 compares the



(a) Rotation Estimation (b) Pose Estimation

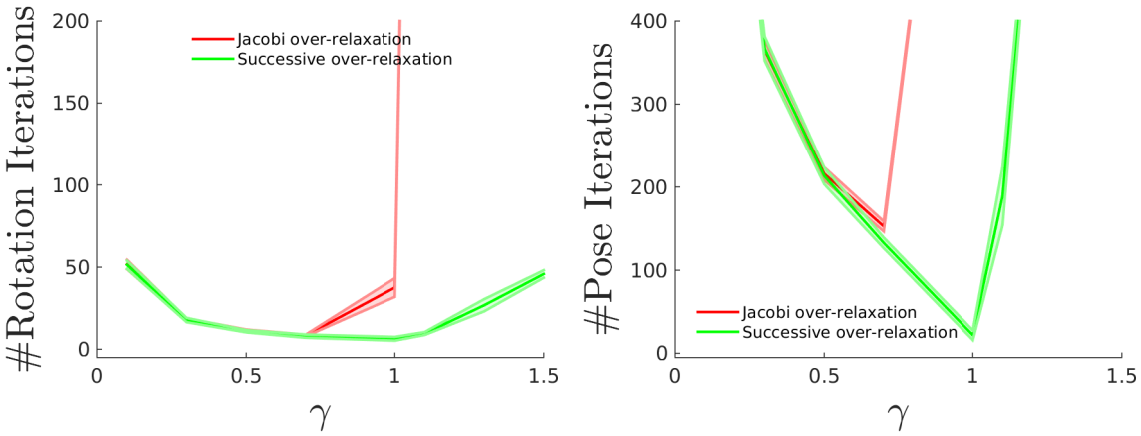
Figure 7: SOR: convergence of (a) rotation estimation and (b) pose estimation for different values of γ (grid scenario, 49 robots).



(a) Rotation Estimation (b) Pose Estimation

Figure 8: JORvs SOR: convergence of (a) rotation estimation and (b) pose estimation for the JOR and SOR algorithms with $\gamma = 1$ (grid scenario, 49 robots).

convergence rate of SOR and JOR for both the rotation subproblem (Fig. 8a) and the pose subproblem (Fig. 8b). We set $\gamma = 1$ in JOR and SOR since we already observed that this choice ensures the best performance. The figure confirms that SOR dominates JOR in both subproblems. Fig. 9 shows the number of iterations for convergence (according to our stopping conditions) and for different choices of the parameter γ. Once again, the figure confirms that the SOR with $\gamma = 1$ is able to converge in the smallest number of iterations, requiring only few tens of iterations in both the rotation and the pose subproblem.

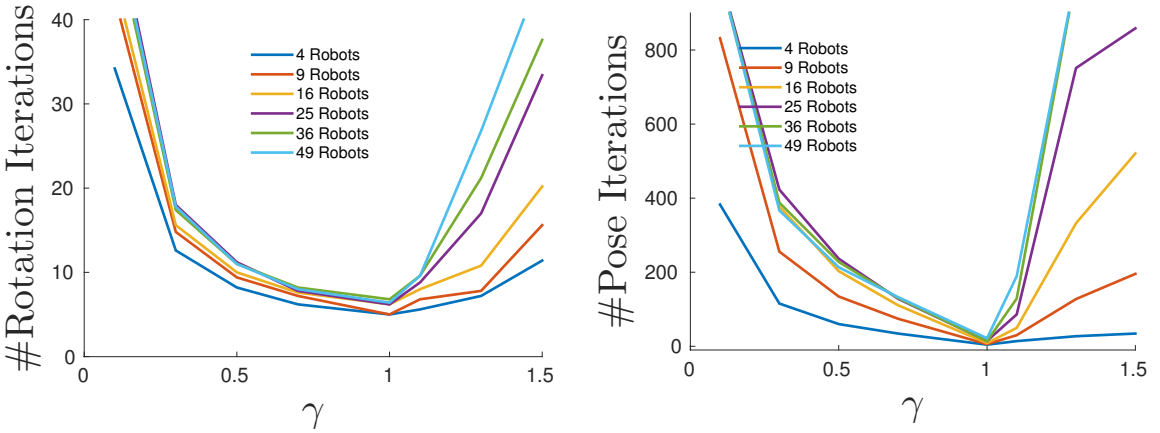


(a) Rotation Estimation (b) Pose Estimation

Figure 9: JORvs SOR: number of iterations required for (a) rotation estimation and (b) pose estimation for the JOR and SOR algorithms with $\gamma = 1$ (grid scenario, 49 robots). The average number of iterations is shown as a solid line, while the 1-sigma standard deviation is shown as a shaded area.

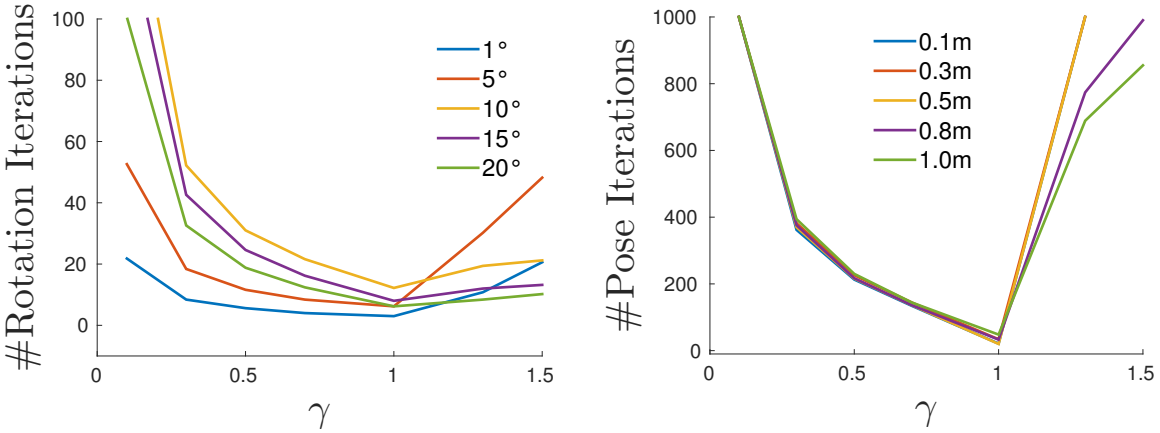
We conclude this section by showing that setting $\gamma = 1$ in SOR ensure faster convergence regardless the number of robots and the measurement noise. Fig. 10 compares the number of iterations required to converge for increasing number of robots for varying γ values. Similarly Fig. 11 compares the number of iterations required to converge for increasing noise for varying γ value. We can see that in both the cases $\gamma = 1$ has the fastest convergence (required the least number of iterations) irrespective of the number of robots and measurement noise. Since SOR with $\gamma = 1$, i.e., the DGS method, is the top performer in all test conditions, in the rest of the chapter we restrict our analysis to this algorithm.

Flagged initialization. In this paragraph we discuss the advantages of the flagged initialization. We compare the DGS method with flagged initialization against a naive initialization in which the variables ($\boldsymbol{r}^{(0)}$ and $\boldsymbol{p}^{(0)}$, respectively) are initialized to zero. The results, for the dataset with 49 robots, are shown in Fig. 12. In both cases the estimation errors go to zero, but the convergence is faster when using the flagged initialization. The speed-up is significant for the second linear system (Fig. 12b). We noticed a similar advantage across all tested scenarios. Therefore, in the rest of the chapter we always adopt the flagged initialization.



(a) Rotation Estimation (b) Pose Estimation

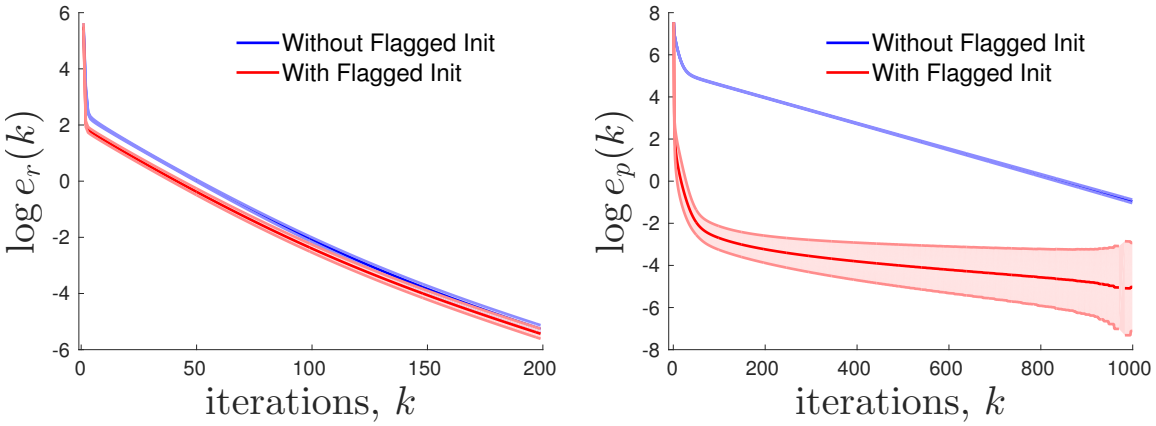
Figure 10: SOR: number of iterations required for (a) rotation estimation and (b) pose estimation in the SOR algorithm for different choices of γ and increasing number of robots.



(a) Rotation Estimation (b) Pose Estimation

Figure 11: SOR: number of iterations required for (a) rotation estimation and (b) pose estimation in the SOR algorithm for different choices of γ and increasing measurement noise.

Stopping conditions and anytime flavor. This section provides extra insights on the convergence of the DGS method. Fig. 13a-b show the evolution of the rotation and pose error for *each* robot in the 49-robot grid: the error associated to each robot (i.e., to each subgraph corresponding to a robot trajectory) is not monotonically decreasing and the error for some robot can increase to bring down the overall error. Fig. 13c-d report the change in the rotation and pose estimate for individual robots. Estimate changes become negligible within few tens of iterations. As mentioned at the beginning of the section, we stop the DGS iterations when the estimate change is sufficiently small (below the thresholds η_r and



(a) Rotation Error (b) Pose Error

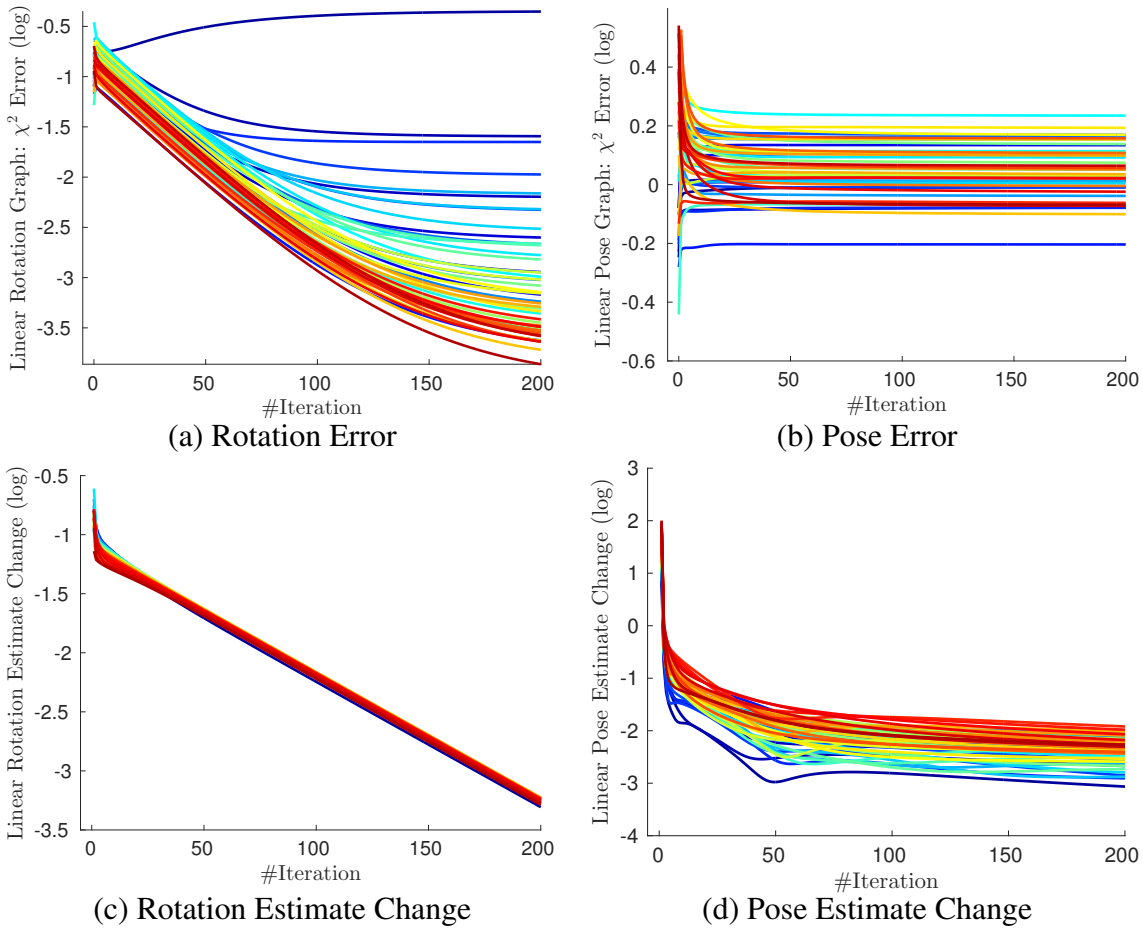
Figure 12: DGS: Comparison between flagged and non-flagged initialization on the grid scenario with 49 robots. Average estimation errors (solid line) and 1-sigma standard deviation (shaded area) are in log scale.

η_p).

Fig. 14 shows the estimated trajectory after 10 and 1000 iterations of the DGS algorithm for the 49-robot grid. The odometric estimate (Fig. 14a) is shown for visualization purposes, while it is not used in our algorithm. We can see that the estimate after 10 iterations is already visually close to the estimate after 1000 iterations. The DGS algorithm has an any-time flavor: the trajectory estimates are already accurate after few iterations and asymptotically converge to the centralized estimate.

Scalability in the number of robots. Fig. 15 shows the average rotation and pose errors for all the simulated datasets (4, 9, 16, 25, 36 and 49 robots). In all cases the errors quickly converge to zero. For large number or robots the convergence rate becomes slightly slower, while in all cases the errors is negligible in few tens of iterations.

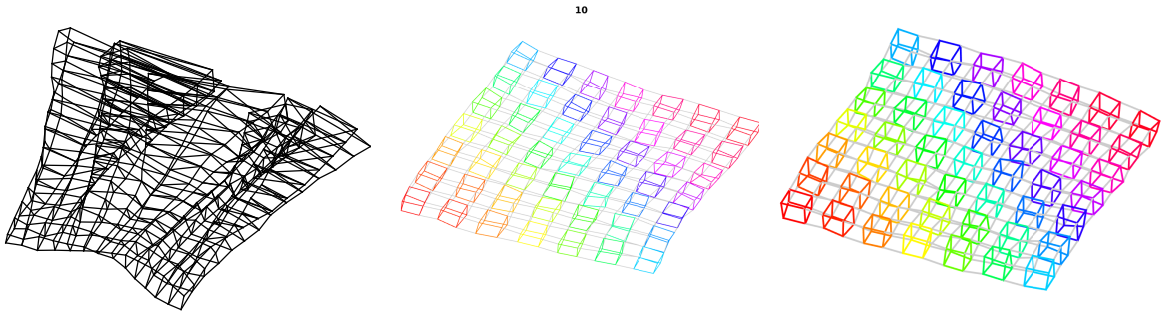
While so far we considered the errors for each subproblem ($e_r(k)$ and $e_p(k)$), we now investigate the overall accuracy of the DGS algorithm to solve our original problem (3). We compare the proposed approach against the centralized two-stage approach of [20] and against a standard (centralized) Gauss-Newton (GN) method, available in gtsam ([35]). We use the cost attained in problem (3) by each technique as accuracy metric (the lower the better). Table 1 reports the number of iterations and the cost attained in problem (3), for



(a) Rotation Error (b) Pose Error

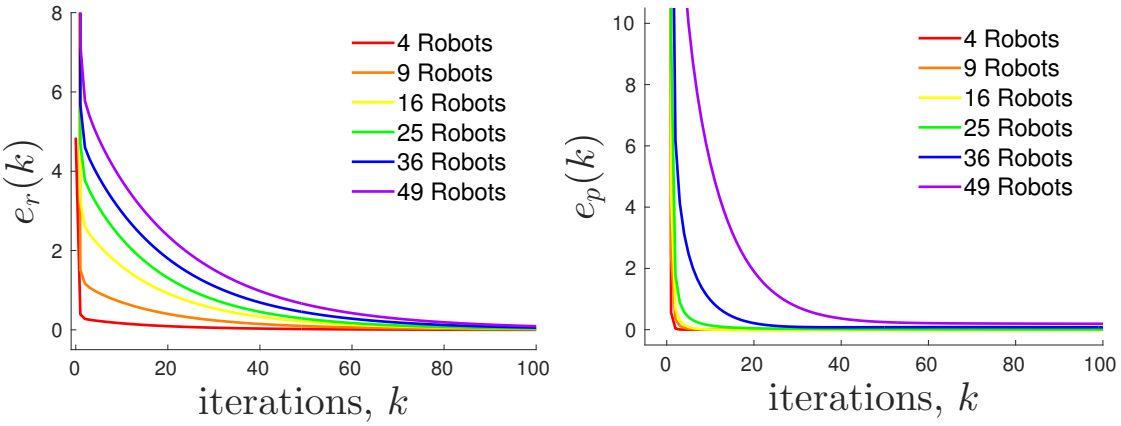
(c) Rotation Estimate Change (d) Pose Estimate Change

Figure 13: DGS: convergence statistics of rotation estimation and pose estimation for each robot (49 Robots). Robots are represented by different color lines.



(a) Initial (b) 10 iterations (c) 1000 iterations

Figure 14: DGS: Trajectory estimates for the scenario with 49 robots. (a) Odometric estimate (not used in our approach and only given for visualization purposes), (b)-(c) DGS estimates after given number of iterations.



(a) Rotation Error (b) Pose Error

Figure 15: DGS: convergence for scenarios with increasing number of robots.

the compared techniques. The number of iterations is the sum of the number of iterations required to solve (6) and (11). The cost of the DGS approach is given for two choices of the thresholds η_r and η_p. As already reported in [20], the last two columns of the table confirm that the centralized two-stage approach is practically as accurate as a GN method. When using a strict stopping condition ($\eta_r = \eta_p = 10^{-2}$), the DGS approach produces the same error as the centralized counterpart (difference smaller than 1%). Relaxing the stopping conditions to $\eta_r = \eta_p = 10^{-1}$ implies a consistent reduction in the number of iterations, at a small loss in accuracy (cost increase is only significant for the scenario with 49 robots). In summary, the DGS algorithm (with $\eta_r = \eta_p = 10^{-1}$) ensures accurate estimation within few iterations, even for large teams.

Sensitivity to measurement noise. Fig. 16 shows the average rotation and pose errors for increasing levels of noise in the scenario with 49 robots. Also in this case, while larger noise seems to imply longer convergence tails, the error becomes sufficiently small after few tens of iterations.

Table 2 evaluates the performance of the DGS method in solving problem (3) for increasing levels of noise, comparing it against the centralized two-stage approach of [20] and the Gauss-Newton method. The DGS approach is able to replicate the accuracy of the centralized two-stage approach, regardless the noise level, while the choice of thresholds

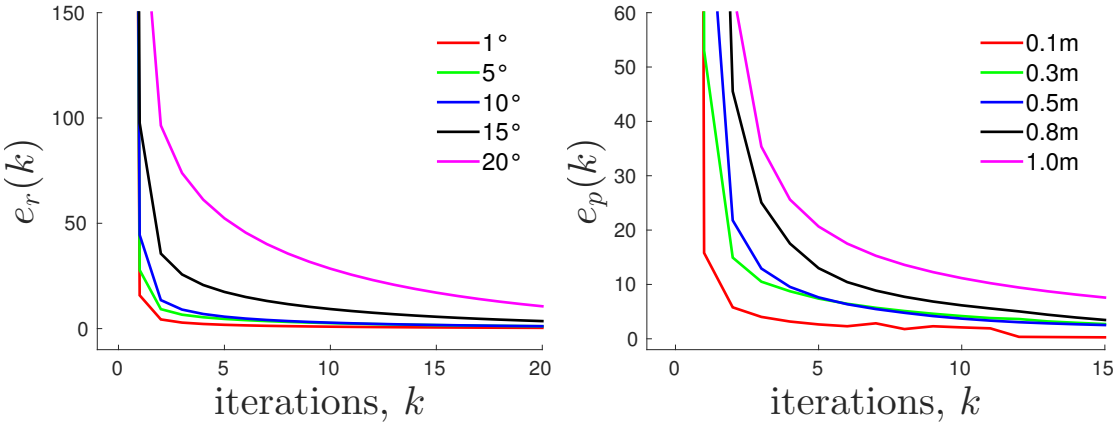
Table 1: Number of iterations and cost attained in problem (3) by the DGS algorithm (for two choices of the stopping conditions), versus a centralized two-stage approach and a GN method. Results are shown for scenarios with *increasing number of robots*. Measurement noise is generated from a Gaussian distribution with standard deviation $\sigma_R = 5^\circ$ for the rotations and $\sigma_t = 0.2$m for the translations. Results are averaged over 10 Monte Carlo runs.

#Robots	Distributed Gauss-Seidel						Centralized	
	$\eta_r = \eta_p = 10^{-1}$			$\eta_r = \eta_p = 10^{-2}$			Two-Stage Cost	GN Cost
	#Iter	Cost	% Diff. w/ GN	#Iter	Cost	% Diff. w/ GN		
4	10	1.9	0	65	1.9	0	1.9	1.9
9	14	5.3	1.9	90	5.2	0	5.2	5.2
16	16	8.9	2.2	163	8.8	1.14	8.8	8.7
25	17	16.2	1.88	147	16.0	0.62	16.0	15.9
36	28	22.9	1.77	155	22.7	0.88	22.6	22.5
49	26	35.1	8.0	337	32.9	1.23	32.7	32.5

$\eta_r = \eta_p = 10^{-1}$ ensures accurate estimation within few tens of iterations.

Scalability in the number of separators. In order to evaluate the impact of the number of separators on convergence, we simulated two robots moving along parallel tracks for 10 time stamps. The number of communication links were varied from 1 (single communication) to 10 (communication at every time), hence the number of separators (for each robot) ranges from 1 to 10. Fig. 17a shows the number of iterations required by the DGS algorithm ($\eta_r = \eta_p = 10^{-1}$), for increasing number of communication links. The number of iterations is fairly insensitive to the number of communication links.

Fig. 17b compares the information exchanged in the DJ algorithm against a state-of-the-art algorithm, DDF-SAM ([30]). In DDF-SAM, each robot sends $K_{GN}\ \left[s\,B_p + (s\,B_p)^2\right]$ bytes, where K_{GN} is the number of iterations required by a GN method applied to problem (3) (we consider the best case $K_{GN} = 1$), s is the number of separators and B_p is the size of a pose in bytes. In the DGS algorithm, each robots sends $K^r_{\text{DGS}}\ (s\,B_r) + K^p_{\text{DGS}}\ (s\,B_p)$ bytes, where K^r_{DGS} and K^p_{DGS} are the number of iterations required by the DGS algorithm



(a) Rotation Noise (b) Translation Noise

Figure 16: DGS: convergence for increasing levels of noise (scenario with 49 Robots). (a) Average rotation error for $\sigma_R = \{1, 5, 10, 15, 20\}^\circ$. (b) Average pose error for $\sigma_t = \{0.1, 0.3, 0.5, 0.8, 1.0\}$m.

Table 2: Number of iterations and cost attained in problem (3) by the DGS algorithm (for two choices of the stopping conditions), versus a centralized two-stage approach and a GN method. Results are shown for *increasing measurement noise* in a scenario with 49 robots .

Measurement noise		Distributed Gauss-Seidel						Centralized	
		$\eta_r = \eta_p = 10^{-1}$			$\eta_r = \eta_p = 10^{-2}$			Two-Stage	GN
$\sigma_r(^\circ)$	σ_t(m)	#Iter	Cost	% Diff. w/ GN	#Iter	Cost	% Diff. w/ GN	Cost	Cost
1	0.05	8.5	2.1	16.0	51.0	1.8	0	1.8	1.8
5	0.1	21.8	14.8	6.47	197.8	14.0	0.71	14.0	13.9
10	0.2	35.6	58.4	4.28	277.7	56.6	1.07	56.6	56.0
15	0.3	39.8	130.5	3.57	236.8	128.4	1.90	129.3	126.0

to solve the linear systems (6) and (11), respectively, and B_r is the size of a rotation (in bytes). We assume $B_r = 9$ doubles (72 bytes)[3] and $B_p = 6$ doubles (48 bytes). The number of iterations K^r_{DGS} and K^p_{DGS} are the one plotted in Fig. 17a. From Fig. 17b we see that the communication burden of DDF-SAM quickly becomes unsustainable, while the linear increase in communication of the DGS algorithm implies large communication saving.

Realistic simulations in Gazebo. We tested our DGS-based approach in two scenarios

[3]In the linear system (6) we relax the orthogonality constraints hence we cannot parametrize the rotations with a minimal 3-parameter representation.

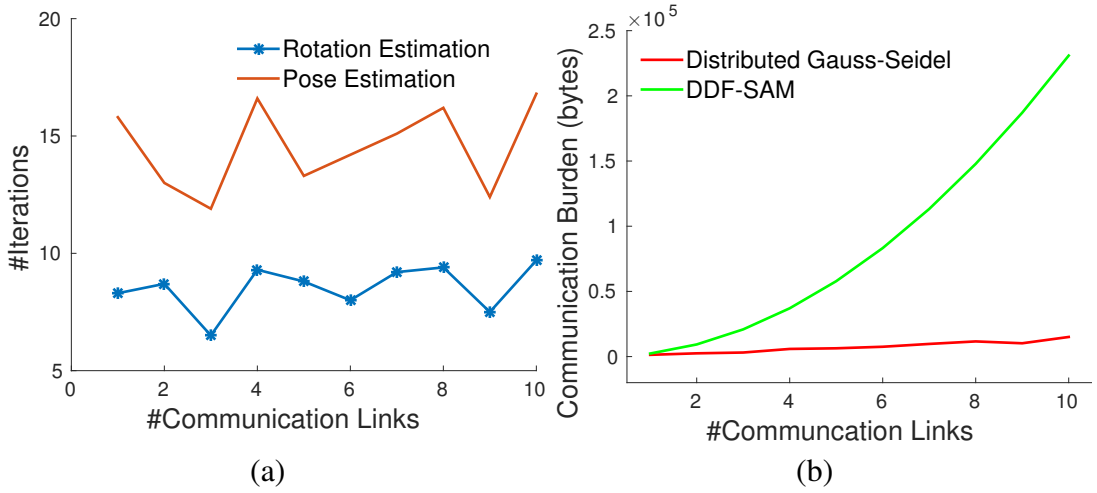


Figure 17: DGS vs DDF-SAM: (a) average number of iterations versus number of separators for the DGS algorithm. (b) communication burden (bytes of exchanged information) for DGS and DDF-SAM, for increasing number of separators.

in Gazebo simulations as shown in Fig. 18. The robots start at fixed locations and explore the environment by moving according to a random walk. Each robot is equipped with a 3D laser range finder, which is used to intra-robot and inter-robot measurements via scan matching. In both scenarios, two robots communicate only when they are within close proximity of each other (0.5m in our tests). Results are average over 100 Monte-Carlo runs.

Fig. 18 shows the aggregated point cloud corresponding to the DGS trajectory estimate, for one of the runs. The point cloud closely resembles the ground truth environment shown in the same figure. Fig. 19a shows that number of steps required to explore the whole environment quickly decreases with increasing number of robots. This intuitive observation motivates our interest towards mapping techniques that can scale to large teams of robots. Fig. 19b reports trajectory samples for different robots in our Monte Carlo analysis.

3.6.2 Field Experiments: Distributed Pose Graph Optimization

We tested the DGS approach on field data collected by two to ten Jackal robots (Fig. 22), moving in a military training facility, Georgia Tech IRIM lab, and Klaus building (3rd Floor).

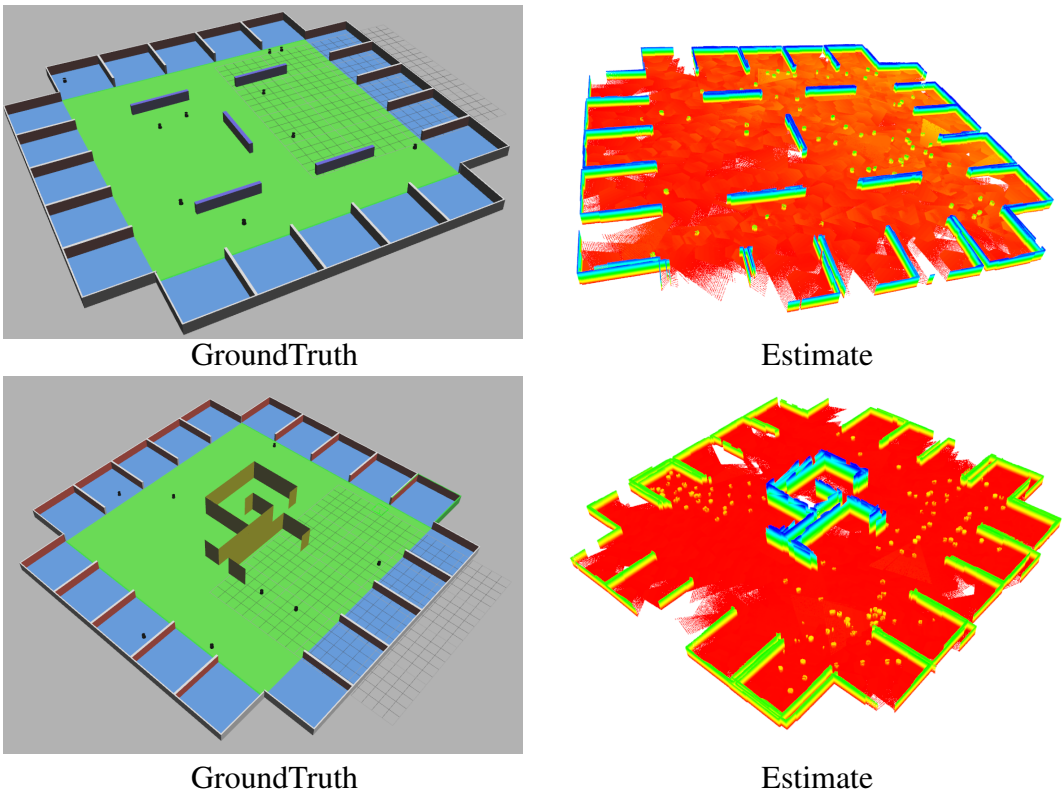
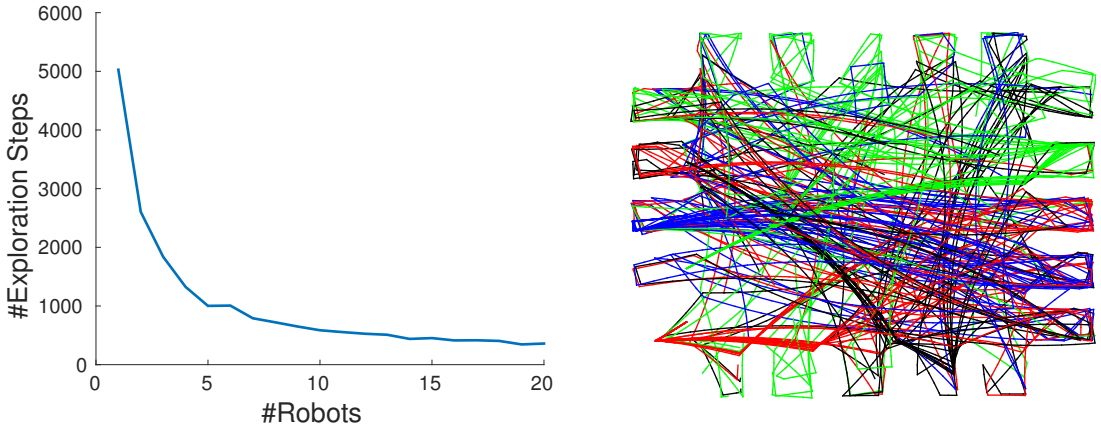


Figure 18: Gazebo tests: ground truth environments and aggregated point clouds corresponding to the DGS estimate.

Figs. 26, 25, 27 show the aggregated 3D point clouds (left), the estimates trajectories (center), and the aggregated occupancy grid map (right) over multiple runs in a military training facility. The central part of the figures compares the DGS estimate against the DDF-SAM estimate and the corresponding centralized estimate. Note that the test scenarios cover a broad set of operating conditions. For instance Fig. 25 corresponds to experiments with 4 robots moving in indoor environment, while Fig. 26 corresponds to tests performed in a mixed indoor-outdoor scenario (with robots moving on gravel when outdoor, Fig. 23). The four tests of Fig. 27 correspond to early results with 2 robots for which we do not have a comparison against DDF-SAM. Fig. 28 corresponds to the additional tests done with 10 robots in the military training facility. Fig. 30 corresponds to the tests done with 11 robots in the IRIM lab. Fig. 32 corresponds to the tests done with 5 robots in Klaus building.



(a) Exploration steps (b) Monte Carlo Runs

Figure 19: (a) Number of exploration steps required to explore a fixed-sized grid with increasing number of robots. (b) Samples of robot trajectories from our Gazebo-based Monte Carlo experiments.

Quantitative results are given in Table 3, which reports the cost attained by the DGS algorithm as compared to the centralized GN cost and DDF-SAM. Number of iterations, ATE* and ARE* are also shown.

These metrics ATE* and ARE* are formally defined below.

Absolute Translation Error (ATE).* Similar to the formulation by Sturm et al. [115], the average translation error measures the absolute distance between the trajectory poses estimated by our approach versus the centralized GN method. The ATE* is defined as:

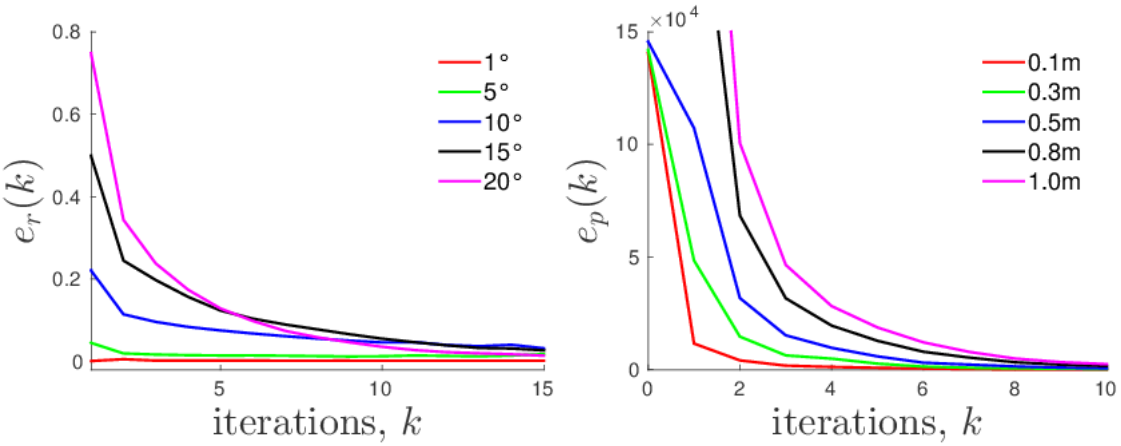
$$ATE^* = \left(\frac{1}{\sum_{\alpha \in \Omega} n_\alpha} \sum_{\alpha \in \Omega} \sum_{i=1}^{n_\alpha} \| \boldsymbol{t}_{\alpha_i} - \boldsymbol{t}^*_{\alpha_i} \|^2 \right)^{\frac{1}{2}} \tag{21}$$

where $\boldsymbol{t}_{\alpha_i}$ is the position estimate for robot α at time i, $\boldsymbol{t}^*_{\alpha_i}$ is the corresponding estimate from GN, and n_α is the number of poses in the trajectory of α.

Absolute Rotation Error (ARE).* The average rotation error is computed by evaluating the angular mismatch between the trajectory rotations produced by the proposed approach versus a centralized GN method:

$$ARE^* = \left(\frac{1}{\sum_{\alpha \in \Omega} n_\alpha} \sum_{\alpha \in \Omega} \sum_{i=1}^{n_\alpha} \| \text{Log} \left((\boldsymbol{R}^*_{\alpha_i})^\mathsf{T} \boldsymbol{R}_{\alpha_i} \right) \|^2 \right)^{\frac{1}{2}} \tag{22}$$

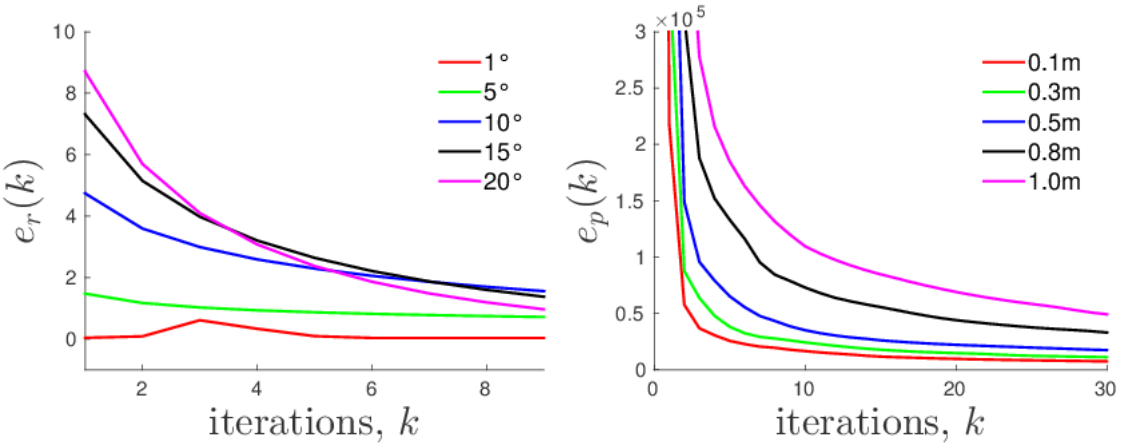
where $\boldsymbol{R}_{\alpha_i}$ is the rotation estimate for robot α at time i, $\boldsymbol{R}^*_{\alpha_i}$ is the corresponding estimate from GN.



(a) Rotation Noise (b) Translation Noise

Figure 20: Convergence for increasing levels of noise (scenario with 2 Robots in Gazebo). (a) Average rotation estimation error for $\sigma_R = \{1, 5, 10, 15, 20\}^\circ$. (b) Average pose estimation error for $\sigma_t = \{0.1, 0.3, 0.5, 0.8, 1.0\}$m.

Each line of the table shows statistics for each of the 15 field tests in the military training facility. The first four rows (tests 0 to 3) correspond to tests performed in a mixed indoor-outdoor scenario (Fig. 26). The next seven rows (tests 4 to 10) correspond to tests performed with 4 robots in an indoor environment. The last four rows (tests 11 to 14) correspond to early results with 2 robots. Higher ATE* and ARE* in the first few rows is due to the fact that the robots move on gravel in outdoors which introduces larger odometric errors. Consistently with what we observed in the previous sections, larger measurement errors may induce the DGS algorithm to perform more iterations to reach consensus (e.g., test 3). The columns "#vertices" and "#edges" describe the size of the overall factor graph (including all robots), while the column "#links" reports the total number of rendezvous events. In all the tests DDF-SAM performed worse than DGS which is shown by higher cost attained by DDF-SAM as compared to DGS. This is because DDF-SAM requires good linearization points to generate condensed graphs and instead bad linearization points during communication can introduce linearization errors resulting in higher cost. Fig. 24 shows the corresponding histogram visualization comparing the cost attained by the DGS



(a) Rotation Noise (b) Translation Noise

Figure 21: Convergence for increasing levels of noise (scenario with 4 Robots in Gazebo). (a) Average rotation estimation error for $\sigma_R = \{1, 5, 10, 15, 20\}^\circ$. (b) Average pose estimation error for $\sigma_t = \{0.1, 0.3, 0.5, 0.8, 1.0\}$m.

algorithm and centralized Two-Stage and GN algorithm. Fig 29, 31 and 33 show the per-robot ATE* comparison w.r.t centralized for the data collection corresponding to additional tests as shown in Fig. 28, 30 and 32. We can see that ATE* in all the three scenarios is less than 0.1 cm which shows that the distributed estimate is accurate as the centralized estimate even when number of robots is scaled up to more than 10.

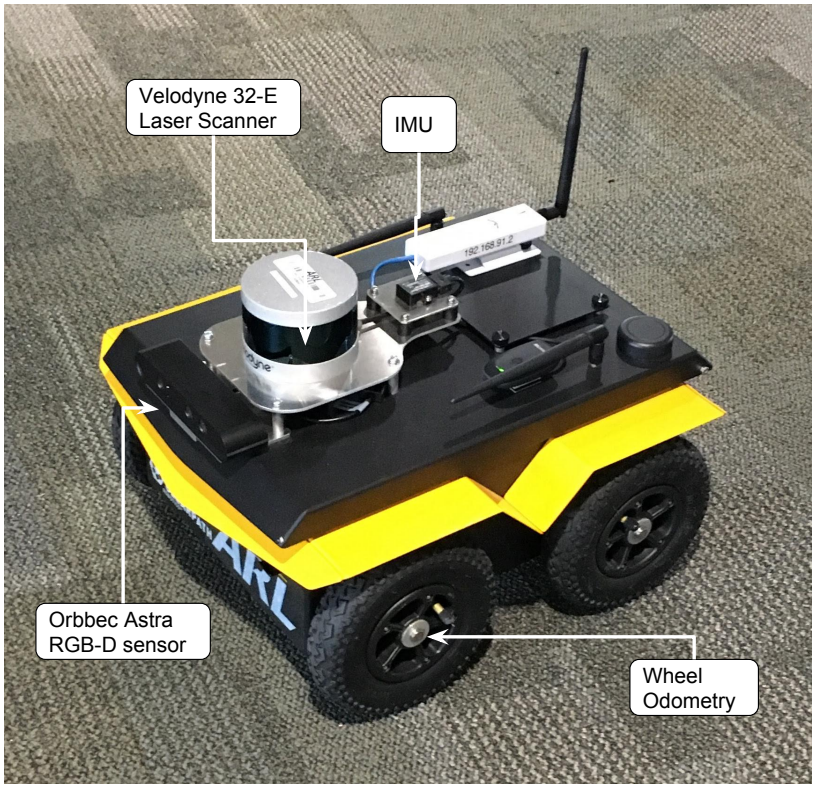


Figure 22: Clearpath Jackal robot used for the field tests: platform and sensor layout;

Figure 23: Clearpath Jackal robot moving on gravel.

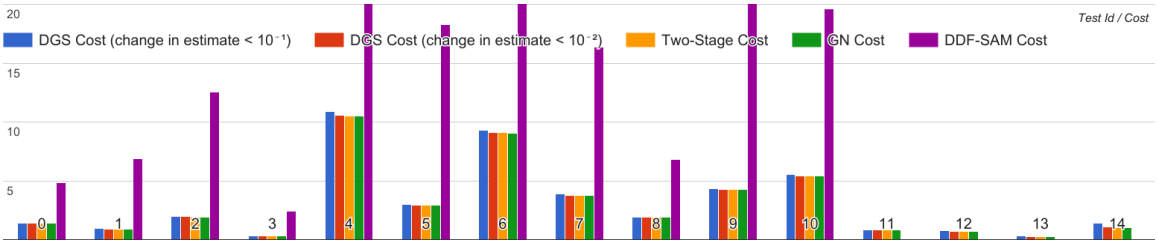


Figure 24: Histogram visualization comparing the cost attained by DGS algorithm on field data as compared to the centralized Two-Stage, GN and DDF-SAM method. It shows that all the proposed approaches are close to GN, while DDF-SAM has worse performance. The length of y-axis (cost) is limited to 20 for visualization purposes. Additional quantitative results are given in Table 3.

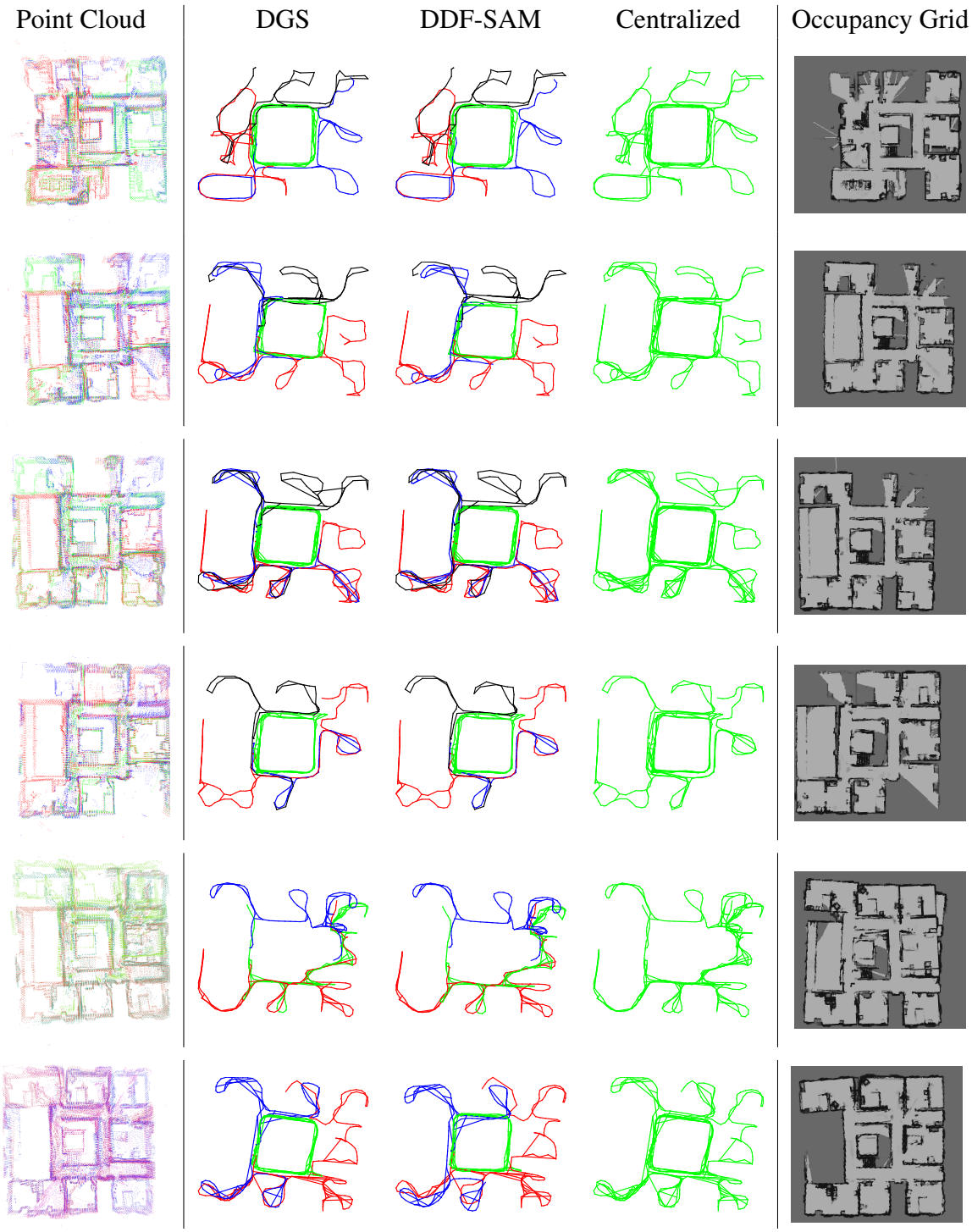


Figure 25: Indoor scenarios: (Left) aggregated point cloud obtained from the DGS trajectory estimate. (Center) estimated trajectories for DGS, GN and DDF-SAM (robots shown in red, blue, green and black for the distributed techniques). (Right) overall occupancy grid map obtained from the DGS trajectory estimate.

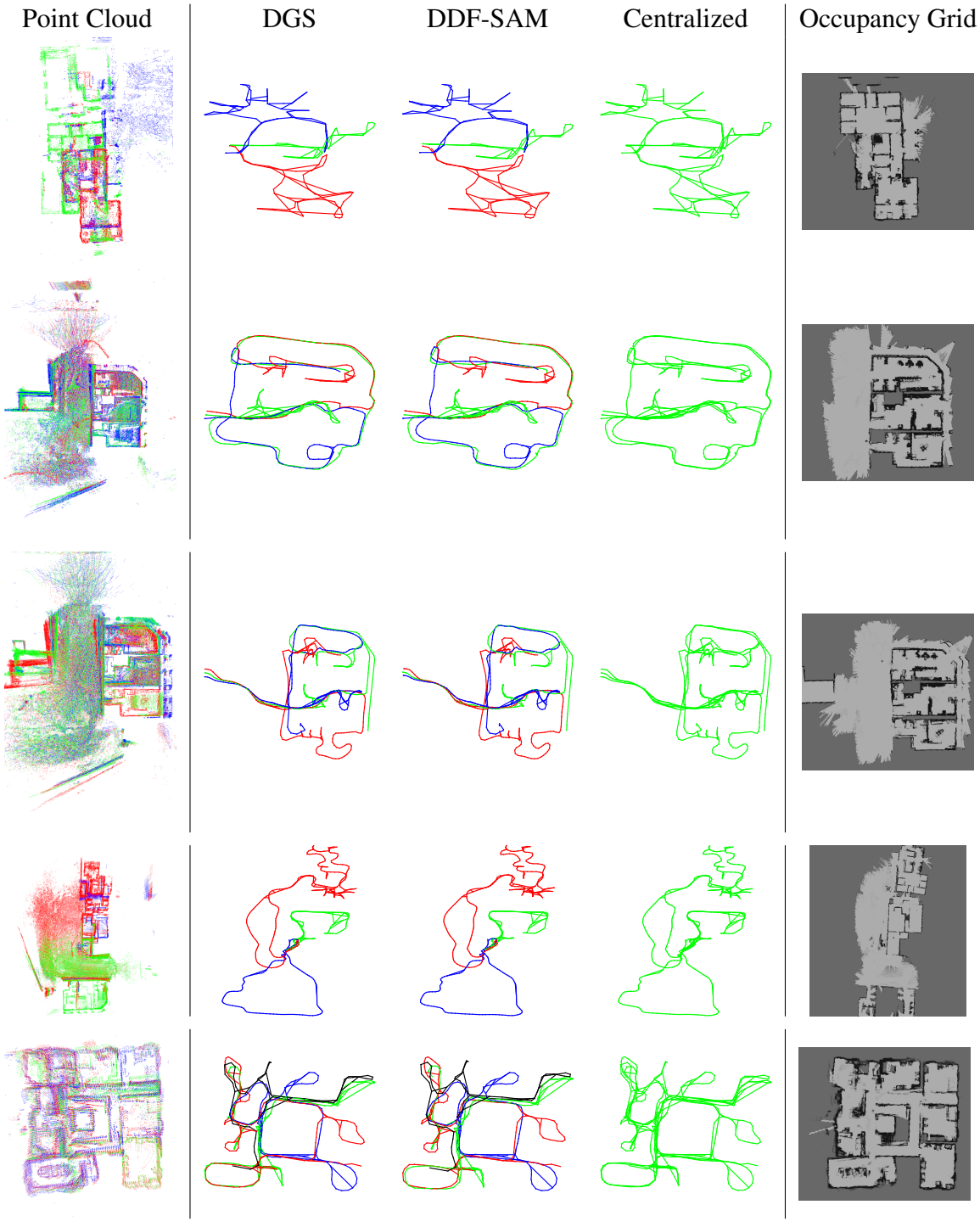


Figure 26: Mixed indoor-outdoor scenarios: (Left) aggregated point cloud obtained from the DGS trajectory estimate. (Center) estimated trajectories for DGS, GN and DDF-SAM (robots shown in red, blue, green and black for the distributed techniques). (Right) overall occupancy grid map obtained from the DGS trajectory estimate.

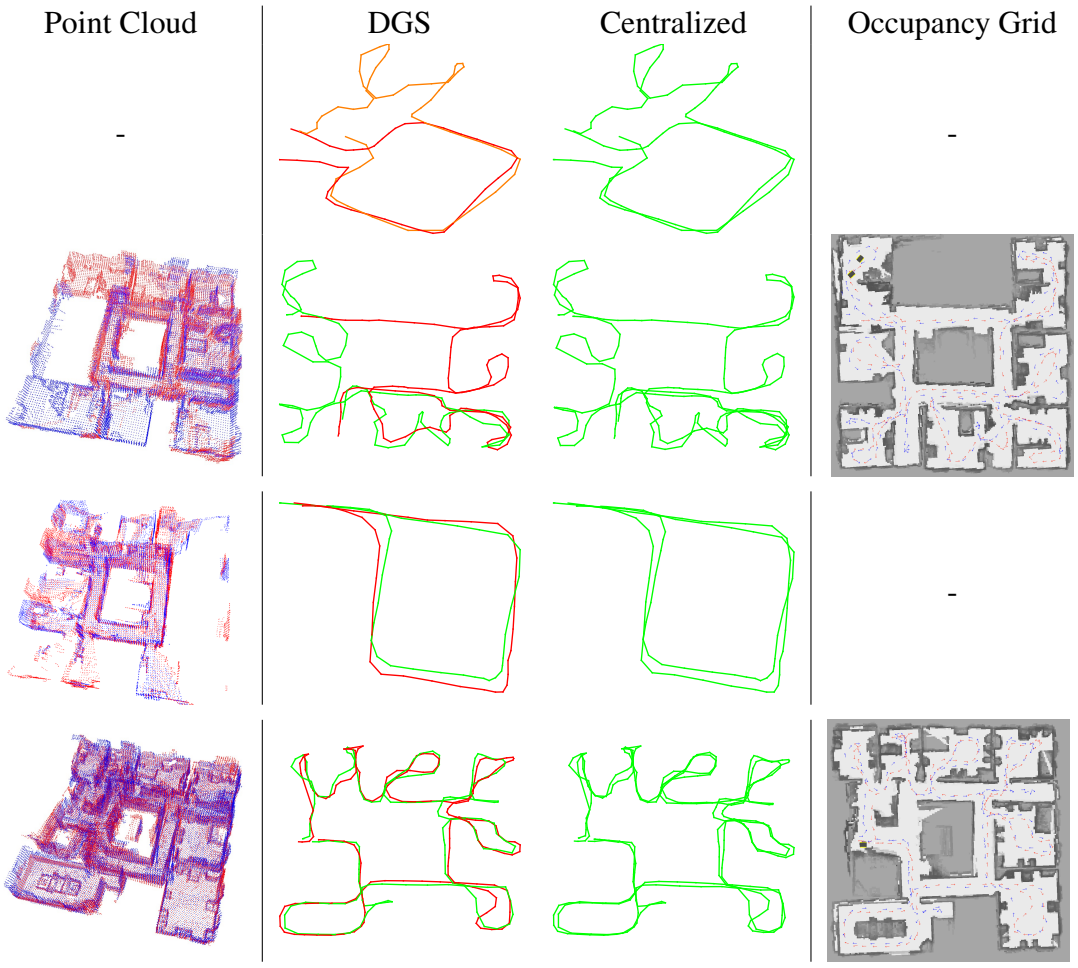
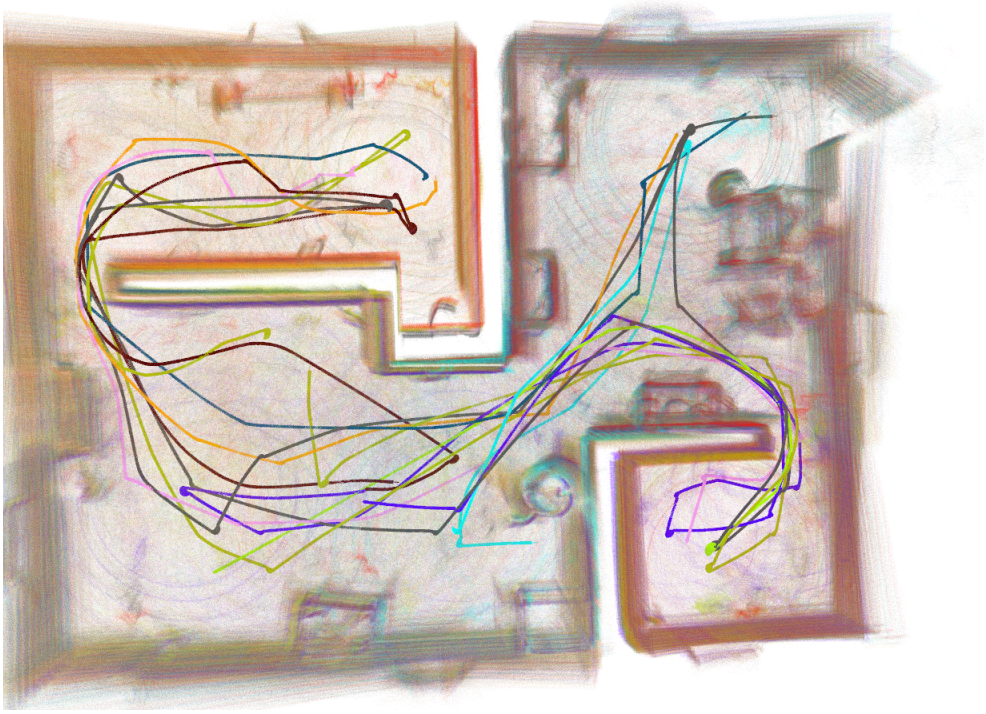


Figure 27: Early tests with 2 robots: (Left) aggregated point cloud obtained from the DGS trajectory estimate. (Center) estimated trajectories for DGS and GN. (Right) overall occupancy grid map obtained from the DGS trajectory estimate.

Table 3: Performance of DGS on field data as compared to the centralized GN method and DDF-SAM. Number of iterations, ATE* and ARE* with respect to centralized Gauss-Newton estimate are also shown.

#Test	#vert.	#edges	#links	Distributed Gauss-Seidel								Centralized		DDF SAM
				$\eta_r = \eta_p = 10^{-1}$				$\eta_r = \eta_p = 10^{-2}$				2-Stage	GN	
				#Iter	Cost	ATE*	ARE*	#Iter	Cost	ATE*	ARE*	Cost	Cost	
0	194	253	16	12	1.42	0.21	1.63	197	1.40	0.07	0.67	1.40	1.40	4.86
1	511	804	134	10	0.95	1.22	6.64	431	0.91	1.18	6.37	0.89	0.89	6.88
2	547	890	155	21	1.99	1.03	4.74	426	1.95	1.08	4.79	1.93	1.93	12.54
3	657	798	47	176	0.32	0.68	2.40	446	0.32	0.69	2.06	0.32	0.32	2.39
4	510	915	179	23	10.89	1.10	6.69	782	10.57	0.71	4.53	10.51	10.50	37.94
5	418	782	151	13	3.02	0.51	5.75	475	2.92	0.39	4.32	2.91	2.90	18.31
6	439	720	108	26	9.28	0.68	5.08	704	9.12	0.31	2.39	9.10	9.07	72.76
7	582	1152	228	10	3.91	0.31	3.40	579	3.78	0.26	2.43	3.78	3.78	16.38
8	404	824	183	11	1.92	0.13	1.78	410	1.89	0.12	1.25	1.89	1.89	6.82
9	496	732	86	41	4.30	0.54	4.20	504	4.29	0.45	3.04	4.28	4.27	21.53
10	525	923	147	15	5.56	0.39	3.93	577	5.43	0.23	2.04	5.43	5.40	19.59
11	103	107	3	71	0.85	1.58	14.44	328	0.84	0.27	2.18	0.84	0.84	-
12	227	325	50	16	0.79	1.11	10.44	511	0.71	0.80	7.02	0.68	0.68	-
13	77	127	26	10	0.33	0.34	2.23	78	0.26	0.21	1.25	0.26	0.26	-
14	322	490	85	28	1.42	0.83	5.05	606	1.07	0.47	2.10	1.04	1.04	-



DGS

Centralized

Figure 28: Test with 10 robots in a military training facility. (Top) Shows the aggregated point cloud and trajectories estimated by DGS method. (Bottom) Shows the aggregated point cloud and trajectories estimated by the centralized GN method. Trajectories and point clouds for different robots are shown using different colors.

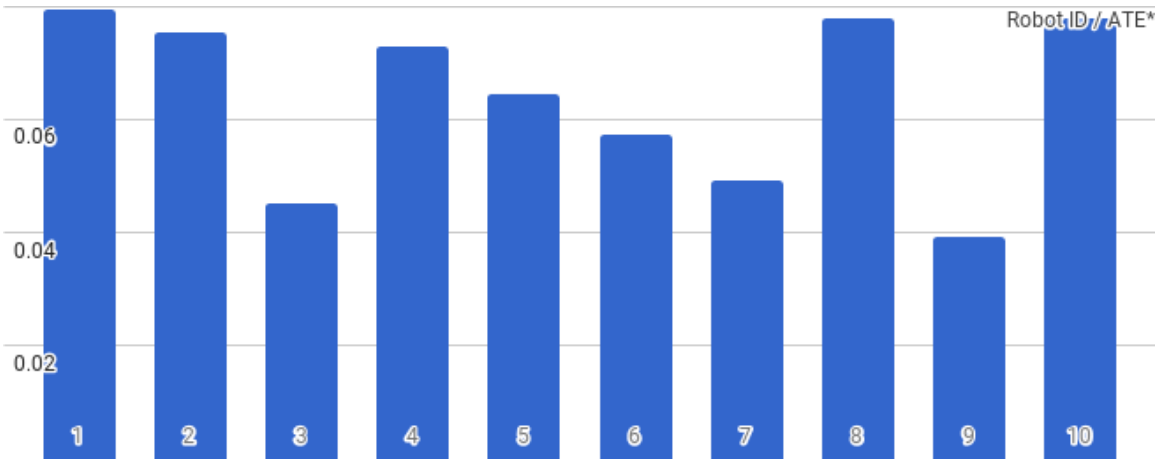
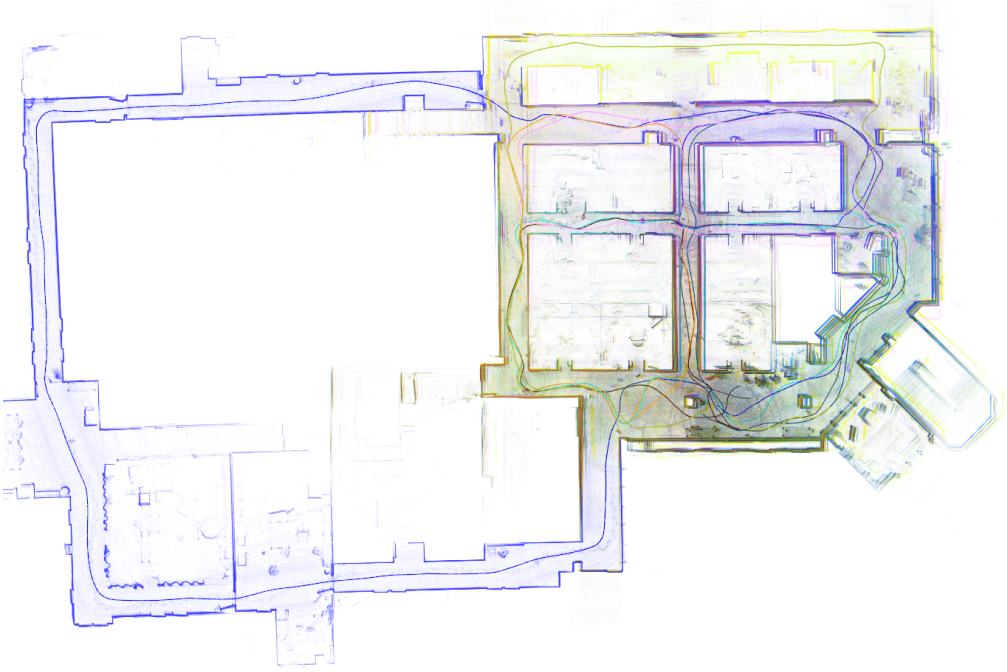
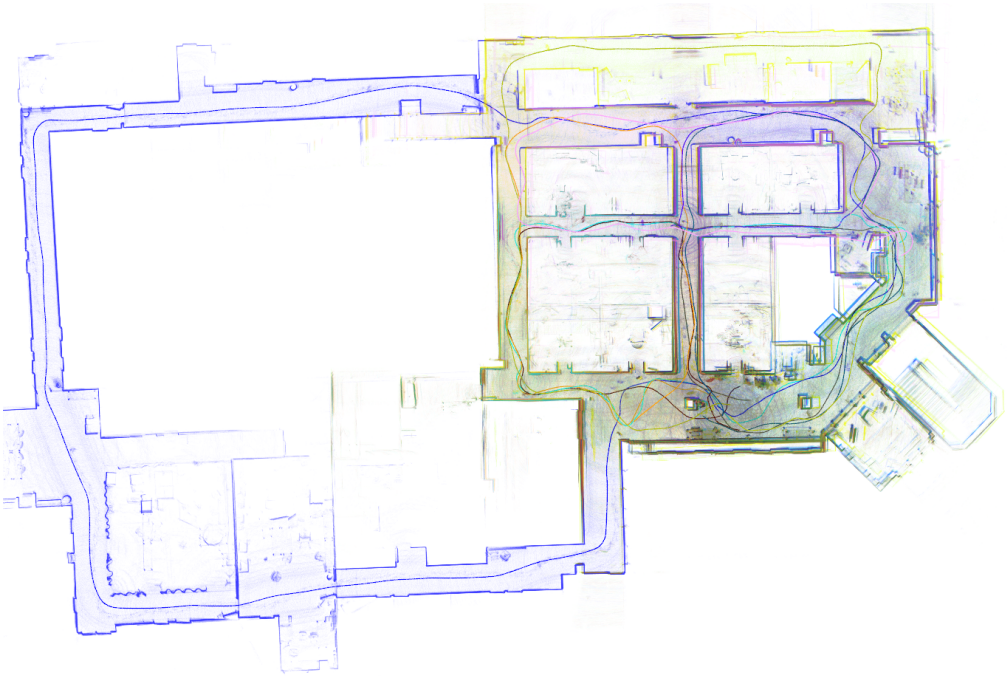


Figure 29: Per-Robot ATE* comparison w.r.t Centralized for data collected in military training facility given in Fig. 28.



DGS



Centralized

Figure 30: Test with 11 robots in the IRIM lab. (Top) Shows the aggregated point cloud and trajectories estimated by DGS method. (Bottom) Shows the aggregated point cloud and trajectories estimated by the centralized GN method. Trajectories and point clouds for different robots are shown using different colors.

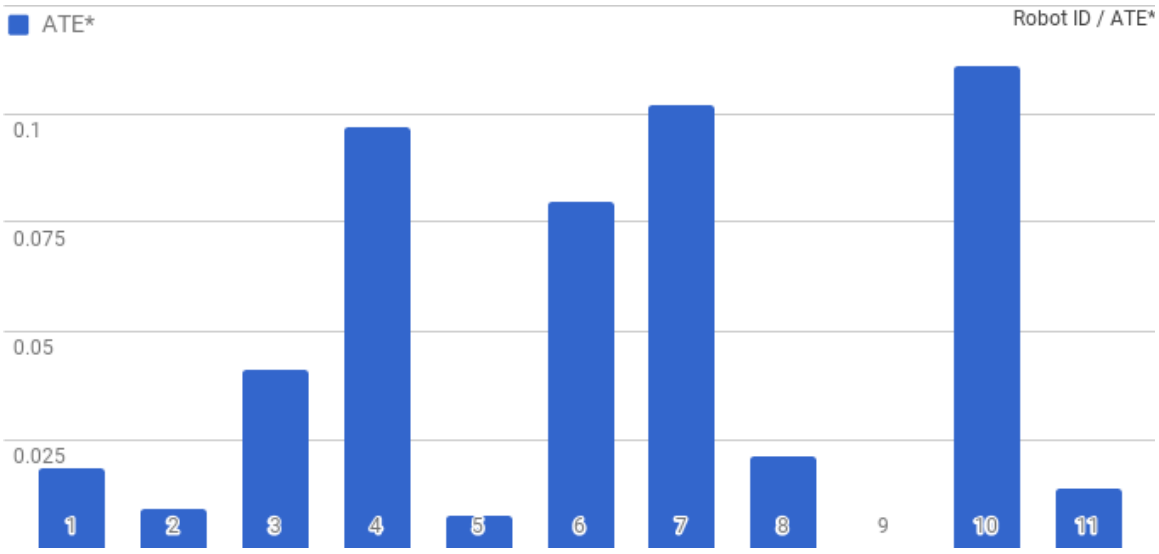
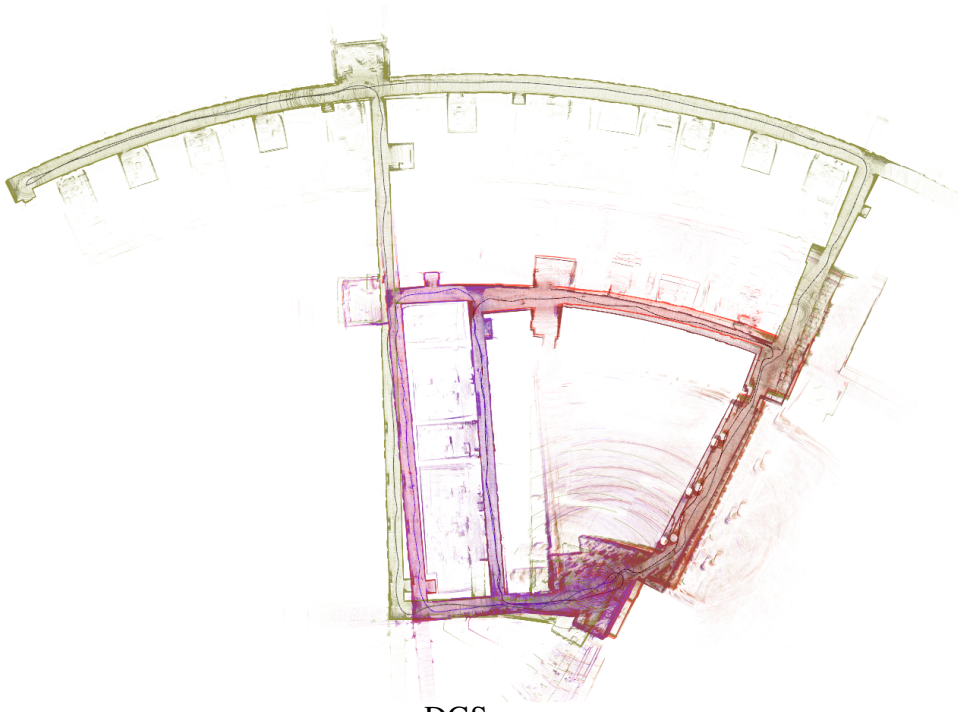
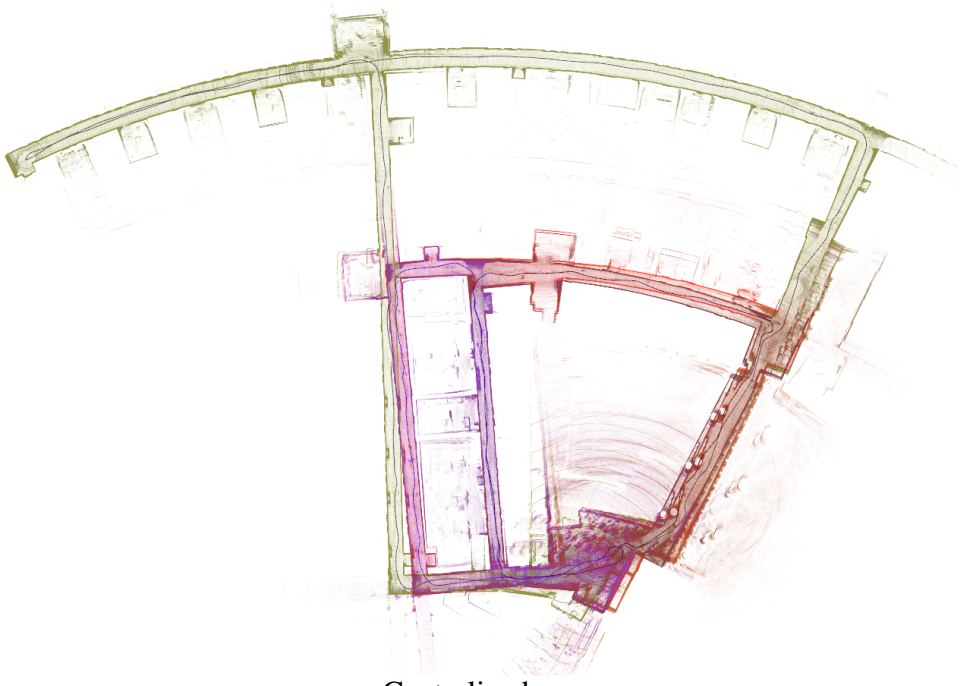


Figure 31: Per-Robot ATE* comparison w.r.t Centralized for data collected in IRIM lab given in Fig. 30.



DGS



Centralized

Figure 32: Test with 5 robots in the Klaus building. (Top) Shows the aggregated point cloud and trajectories estimated by DGS method. (Bottom) Shows the aggregated point cloud and trajectories estimated by the centralized GN method. Trajectories and point clouds for different robots are shown using different colors. The misalignments at the bottom left is due to the lack of inter-robot loop closures in that region.

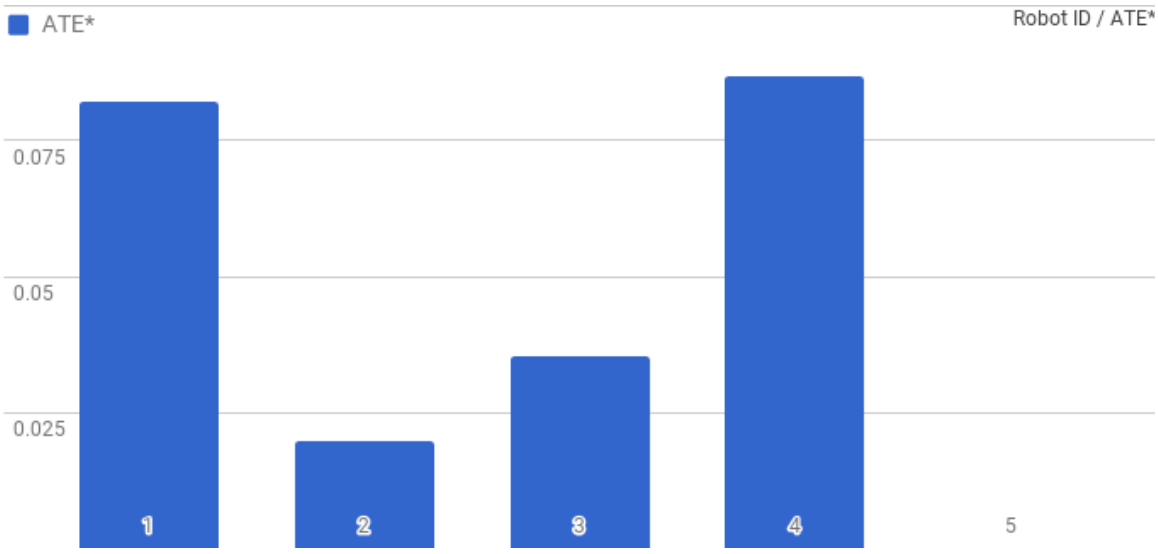


Figure 33: Per-Robot ATE* comparison w.r.t Centralized for data collected in Klaus lab given in Fig. 32.

3.7 Conclusions

We investigated distributed algorithms to estimate the 3D trajectories of multiple cooperative robots from relative pose measurements. Our contribution is the design of a 2-stage approach for distributed pose estimation and propose a number of algorithmic variants.One of these algorithms, the Distributed Gauss-Seidel (DGS) method, is shown to have excellent performance in practice: (i) its communication burden scale linearly in the number of separators and respect agents' privacy, (ii) it is robust to noise and the resulting estimates are sufficiently accurate after few communication rounds, (iii) the approach is simple to implement and scales well to large teams. We demonstrated the effectiveness of the DGS approach in extensive simulations and field tests.

CHAPTER IV

DISTRIBUTED OBJECT BASED SLAM WITH KNOWN OBJECT MODELS

4.1 Introduction

Dealing with bandwidth constraints is challenging for two reasons. First, most approaches for multi robot SLAM imply a communication burden that grows quadratically in the number of locations co-observed by the robots [29]; these approaches are doomed to quickly hit the bandwidth constraints. In the previous chapter (Ch. 3) we alleviated this issue by proposing an approach, based on the distributed Gauss-Seidel method, which requires linear communication. The second issue regards the communication cost of establishing loop closures among robots. When the robots are not able to directly detect each other, loop closures have to be found by comparing raw sensor data; in our setup the robots are equipped with an RGBD camera and exchanging multiple 3D point clouds quickly becomes impractical in presence of communication bounds. In this chapter we address this second issue by using an object-level map representation.

Related Work. Traditional approaches for multi robot mapping use low-level primitives like points and lines to model the geometry of the environment [32]; these maps become memory-intensive in long-term operation, contain very redundant information (e.g., use thousands of points to represent a planar surface), and lack semantic understanding, which is a key element in a wide range of tasks (e.g., human robot interaction or manipulation). For these reasons, *semantic mapping* has attracted a conspicuous interest from the community, starting from early papers [72], to more recent works which use object templates [108], door signs [101], or planes [127] for mapping. A recent survey can be found

in [70]. *Distributed* estimation in multi robot systems is currently an active field of research, with special attention being paid to communication constraints [94], heterogeneous teams [59] and robust data association [37]. The robotic literature offers distributed implementations of different estimation techniques, including Extended Kalman filters [103], information filters [124], and Gaussian elimination [29].

Contribution. In this chapter we advocate the use of higher-level map representations as a tool to enable operation in bandwidth-constrained multi robot scenarios. Maps augmented with objects provide a number of advantages: objects (including planes and other geometric shapes) can be represented in a compact manner and provide a richer and human-understandable description of the environment. Objects are more discriminative, which helps data association and loop-closure detection. Finally, object representations reduce the computational complexity of SLAM by reducing the number of variables (intuitively, we estimate the pose of few objects rather than the position of several 3D points).

We propose an approach for Multi Robot Object-based SLAM with two distinctive features. The first is the front-end, which performs accurate object detection using *deep learning*. Deep learning provides an effective tool to generalize early work on object-based mapping [108] to a large number of object categories. The second is the back-end, which implements distributed pose graph optimization using the distributed Gauss-Seidel method, described in our previous work [23]. We show that the combination of these two techniques reduces the memory requirement and information exchange among robots, allows accurate and parsimonious large-scale mapping, and scales to large teams.

Section 4.2 introduces the additional mathematical notation and formalizes the problem of distributed object-based SLAM. Section 4.3 presents the implementation details of our distributed object-based SLAM system.

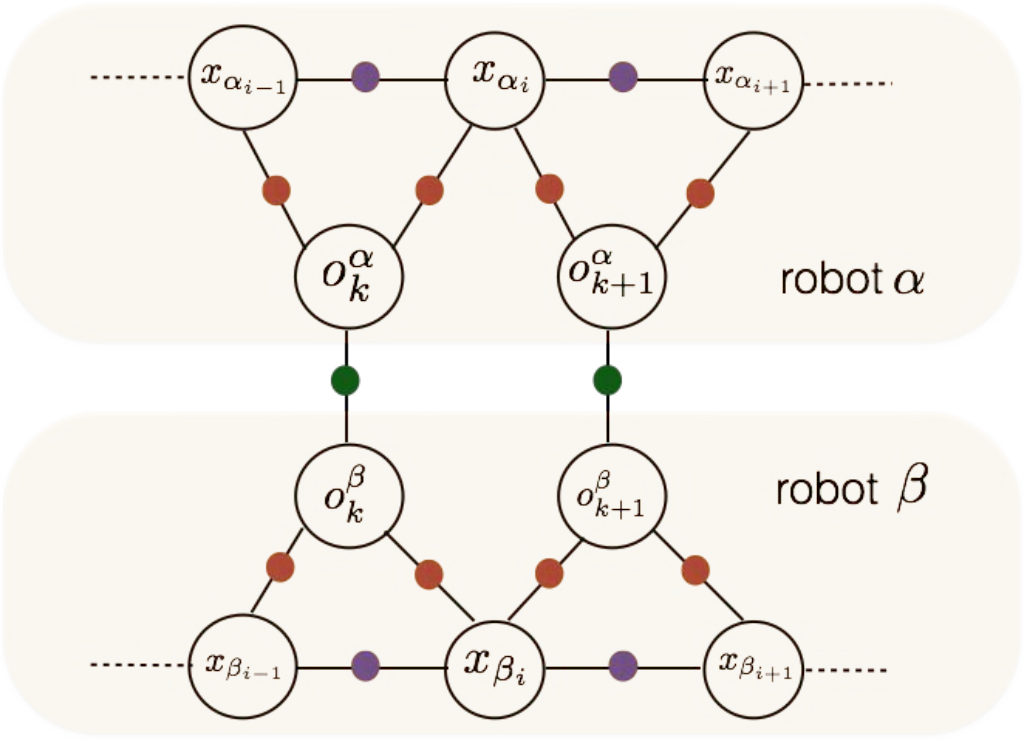


Figure 34: Factor graph representation of Multi-Robot Object based SLAM. $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{x}_{\beta_i}$ denote the poses assumed by robot α and β at time i respectively. The pose of the k^{th} object as estimated by robot α and β is denoted with $\boldsymbol{o}_k^\alpha$ and $\boldsymbol{o}_k^\beta$ respectively. Green dots shows inter-robot factors whereas orange and purple dots shows intra-robot factors.

4.2 Problem Formulation: Distributed Object-based SLAM

We consider a multi robot system as defined in Section 3.2. Each robot, in addition to estimating its own trajectory using local measurements and occasional communication with other robots, also estimates the pose of a set of objects in the environment. We model each trajectory as a finite set of poses; the trajectory of robot α is $\boldsymbol{x}_\alpha = [\boldsymbol{x}_{\alpha_1}, \boldsymbol{x}_{\alpha_2}, \ldots]$. In addition, we denote with $\boldsymbol{o}_k^\alpha \in \mathrm{SE}(3)$ the pose of the k^{th} object as estimated by of robot α (Fig. 34).

Measurements. Similar to distributed pose graph optimization (Section 3.2), we assume that each robot acquires two types of relative pose measurements: intra-robot and inter-robot measurements. The *intra-robot measurements* consist of the odometry measurements, which constrain consecutive robot poses (e.g., $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{x}_{\alpha_{i+1}}$ in Fig. 34), and

object measurements which constrains robot poses with the corresponding visible object landmarks (e.g., $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{o}_k^\alpha$ in Fig. 34). Contrarily to Section 3.2, the *inter-robot measurements* relate the object poses observed by different robots. During a rendezvous between robot α and robot β, each robot shares the label and pose of detected object landmarks with the other robot. Then, for each object observed by both robots, the teammates add an inter-robot measurements, enforcing the object pose estimate to be consistent across the teammates. For instance, if $\boldsymbol{o}_k^\beta$ and $\boldsymbol{o}_k^\alpha$ in Fig. 34 model the pose of the same object, then the two poses should be identical in the global coordinate frame. For this reason, inter-robot measurement between a pair of associated object poses is the identity.

The intra-robot object measurements follow the same measurements model of eq. (1). For instance, if the robot α at time i and at pose $\boldsymbol{x}_{\alpha_i}$ observes an object at pose $\boldsymbol{o}_k^\alpha$, then the corresponding measurement $\bar{\boldsymbol{z}}_{\boldsymbol{o}_k^\alpha}^{\boldsymbol{x}_{\alpha_i}}$ measures the relative pose between $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{o}_k^\alpha$. Similarly we denote inter-robot measurement between object poses $\boldsymbol{o}_k^\alpha$ and $\boldsymbol{o}_k^\beta$ as $\bar{\boldsymbol{z}}_{\boldsymbol{o}_k^\beta}^{\boldsymbol{o}_k^\alpha}$. In order to ensure that the object pose estimate is consistent across teammates, we define the inter-robot measurement model $\bar{\boldsymbol{z}}_{\boldsymbol{o}_k^\beta}^{\boldsymbol{o}_k^\alpha}$ as:

$$\bar{\boldsymbol{z}}_{\boldsymbol{o}_k^\beta}^{\boldsymbol{o}_k^\alpha} \doteq (\mathbf{I}, \mathbf{0}), \quad \text{with:} \begin{cases} \boldsymbol{R}_{\boldsymbol{o}_k^\beta}^{\boldsymbol{o}_k^\alpha} = (\boldsymbol{R}_{\boldsymbol{o}_k^\alpha})^\mathsf{T} \boldsymbol{R}_{\boldsymbol{o}_k^\beta} \boldsymbol{R}_\epsilon = \mathbf{I} \\ \bar{\boldsymbol{t}}_{\boldsymbol{o}_k^\beta}^{\boldsymbol{o}_k^\alpha} = (\boldsymbol{R}_{\boldsymbol{o}_k^\alpha})^\mathsf{T} (\boldsymbol{t}_{\boldsymbol{o}_k^\beta} - \boldsymbol{t}_{\boldsymbol{o}_k^\alpha}) + \boldsymbol{t}_\epsilon = \mathbf{0} \end{cases} \tag{23}$$

where the relative object pose measurement $\bar{\boldsymbol{z}}_{\boldsymbol{o}_k^\beta}^{\boldsymbol{o}_k^\alpha}$ includes the relative rotation measurements $\boldsymbol{R}_{\boldsymbol{o}_k^\beta}^{\boldsymbol{o}_k^\alpha} = \mathbf{I}$, which describes the attitude of the estimated object pose $\boldsymbol{o}_k^\beta$, $\boldsymbol{R}_{\boldsymbol{o}_j^\beta}$ with respect to the reference frame of robot α, "plus" a random rotation $\boldsymbol{R}_\epsilon$ (estimation noise), and the relative position measurement $\bar{\boldsymbol{t}}_{\boldsymbol{o}_k^\beta}^{\boldsymbol{o}_k^\alpha} = \mathbf{0}$, which describes the position $\boldsymbol{t}_{\boldsymbol{o}_k^\beta}$ in the reference frame of robot α, plus random noise $\boldsymbol{t}_\epsilon$.

In the following, we denote with $\mathcal{E}_I^\alpha$ the set of intra-robot odometry for robot α, while we call $\mathcal{E}_I$ the set of intra-robot odometry measurements for all robots in the team, i.e., $\mathcal{E}_I = \cup_{\alpha \in \boldsymbol{\Omega}} \mathcal{E}_I^\alpha$. Similarly the set of intra-robot object measurements for robot α is denoted as $\mathcal{E}_o^\alpha$, whereas the set of all intra-robot object measurements is denoted as $\mathcal{E}_o$. Similar to Section

3.2, the set of inter-robot measurements involving robot α is denoted with $\mathcal{E}_S^\alpha$. The set of all inter-robot measurements is denoted with $\mathcal{E}_S$. The set of all available measurements is then $\mathcal{E} = \mathcal{E}_I \cup \mathcal{E}_O \cup \mathcal{E}_S$. Note that each robot only has access to its own intra and inter-robot measurements $\mathcal{E}_I^\alpha$, $\mathcal{E}_O^\alpha$ and $\mathcal{E}_S^\alpha$.

ML trajectory and objects estimation. Let us collect all robot trajectories and object poses in a (to-be-estimated) set of robot poses $\boldsymbol{x} = [\boldsymbol{x}_\alpha, \boldsymbol{x}_\beta, \boldsymbol{x}_\gamma, \ldots]$ and set of object poses $\boldsymbol{o} = [\boldsymbol{o}^\alpha, \boldsymbol{o}^\beta, \boldsymbol{o}^\gamma, \ldots]$. The ML estimate for $\boldsymbol{x}$ and $\boldsymbol{o}$ is defined as the maximum of the measurement likelihood:

$$\boldsymbol{x}^\star, \boldsymbol{o}^\star = \arg\max_{\boldsymbol{x},\boldsymbol{o}} \prod_{(\boldsymbol{x}_{\alpha_i}, \boldsymbol{x}_{\alpha_{i+1}}) \in \mathcal{E}_I} \underbrace{\mathcal{L}(\bar{\boldsymbol{z}}^{\boldsymbol{x}_{\alpha_i}}_{\boldsymbol{x}_{\alpha_{i+1}}} \mid \boldsymbol{x})}_{\text{odometry factors}} \prod_{(\boldsymbol{x}_{\alpha_i}, \boldsymbol{o}^\alpha_k) \in \mathcal{E}_O} \underbrace{\mathcal{L}(\bar{\boldsymbol{z}}^{\boldsymbol{x}_{\alpha_i}}_{\boldsymbol{o}^\alpha_k} \mid \boldsymbol{x}, \boldsymbol{o})}_{\text{intra-robot object-measurement factors}}$$
$$\prod_{(\boldsymbol{o}^\alpha_i, \boldsymbol{o}^\beta_j) \in \mathcal{E}_S} \underbrace{\mathcal{L}(\bar{\boldsymbol{z}}^{\boldsymbol{o}^\alpha_i}_{\boldsymbol{o}^\beta_j} \mid \boldsymbol{x}, \boldsymbol{o})}_{\text{inter-robot object-object factors}} \tag{24}$$

where we used the same assumptions on measurement noise as in Section 3.2. Defining $\mathcal{X} = \boldsymbol{x} \cup \boldsymbol{o}$, we rewrite eq. (28) as:

$$\mathcal{X}^\star = \arg\max_{\mathcal{X}} \prod_{(\alpha_i, \beta_j) \in \mathcal{E}} \mathcal{L}(\bar{\boldsymbol{z}}^{\alpha_i}_{\beta_j} \mid \mathcal{X}) \tag{25}$$

Since the optimization problem in eq. (29) has the same structure of the one in eq. (2), we follow the same steps to solve it in a distributed manner using the Distributed Gauss-Seidel method.

The next section presents the implementation details of our distributed object-based SLAM system.

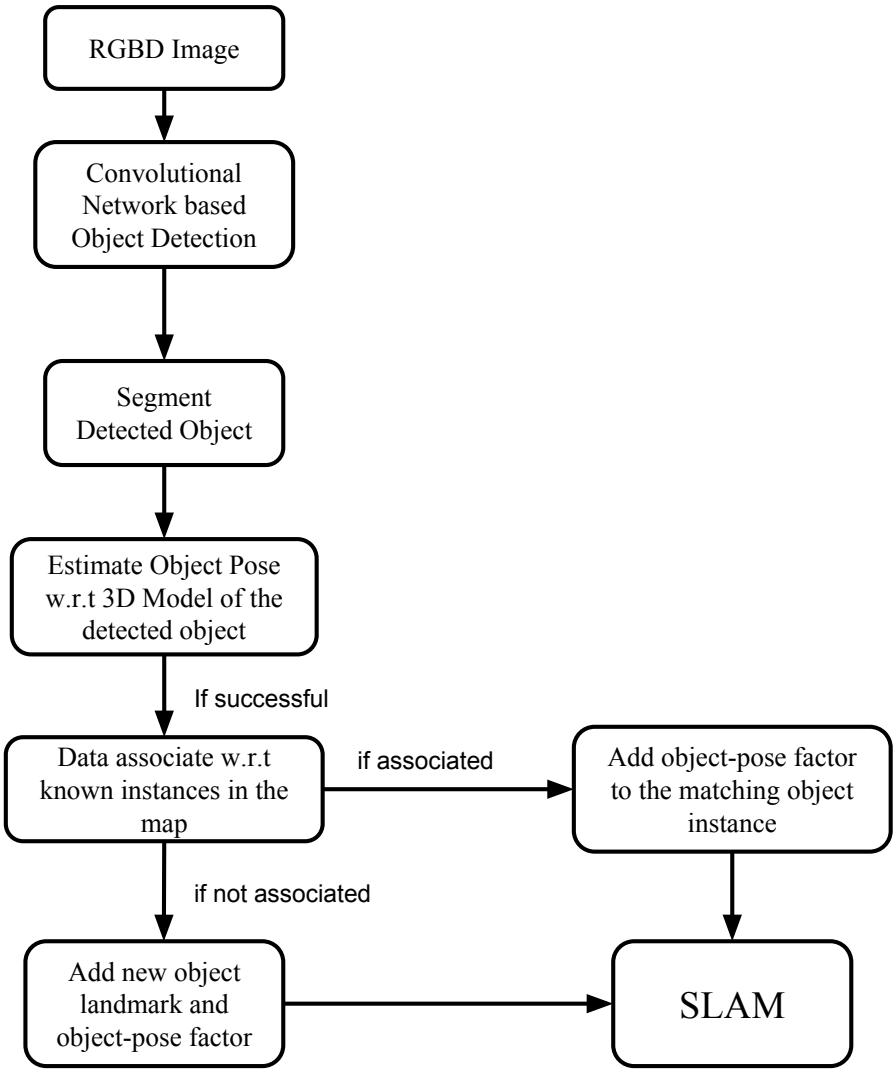


Figure 35: Flowchart of Object based SLAM

4.3 Implementation Details: Distributed Object based SLAM

Ego-Motion Estimation. Each robot collects 3D scans using Velodyne 32E, RGB-D scans using Orbbec Astra sensor, and inertial measurements using IMU and odometry measurements using wheel sensors. Fig. 22 shows the sensor layout on a Jackal robot. Relative pose estimates from all the sensors are fused together using OmniMapper[128] to estimate robot's ego-motion[1]. Fig. 3 shows the overview of ego-motion estimation.

[1] `https://github.com/CognitiveRobotics/omnimapper`

Specifically, 3D scans from Veldoyne 32E are used to compute relative pose with respect to the previous scan using GICP (generalized iterative closest point [110]). Inertial measurements from IMU are fused with Wheel Odometry measurements to generate IMU corrected odometry estimates. Relative pose estimates using RGB-D scans are computed using ORB-SLAM [85]. Relative pose estimates are then fused using OmniMapper.

Object detection and pose estimation. In our approach, each RGB frame (from RGBD) is passed to the YOLO object detector [99, 100], which detects objects at 45 frames per second. Compared to object-proposal-based detectors, YOLO is fast, since it avoids the computation burden of extracting object proposals, and is less likely to produce false positives in the background. We fine-tune the YOLO detector on a subset of objects from the *BigBird* dataset ([112]). The training dataset contains the object images in a clean background taken from different viewpoints and labeled images of the same objects taken by a robot in an indoor environment. During testing, we use a probability threshold of 0.3 to avoid false detections.

Each detected object bounding box is segmented using the *organized point cloud segmentation* [129]. The segmented object is then matched to the 3D template of the detected object class to estimate its pose. We extract PFHRGB features [107] in the source (object segment) and target (object model) point clouds and register the two point clouds in a Sample Consensus Initial Alignment framework [106]. If we have at least 12 inlier correspondences, GICP (generalized iterative closest point [110] is performed to further refine the registration and the final transformation is used as the object pose estimate. If less than 12 inlier correspondences are found, the detection is considered to be a false positive and the corresponding measurement is discarded. In hindsight, this approach verifies the detection both semantically and geometrically.

Data Association. If object pose estimation is successful, it is data-associated with other instances already present in the map by finding the object landmark having the same category label within 2σ distance of the newly detected object. If there are multiple objects

with the same label within that distance, the newly detected object is matched to the nearest object instance. If there exists no object having the same label, a new object landmark is created.

Before the first rendezvous event, each robot performs standard single-robot SLAM using OmniMapper [127]. Both wheel odometry and relative pose measurements to the observed objects are fed to the SLAM back-end. A flowchart of the approach is given in Fig. 35.

Robot Communication. During a rendezvous between robots α and β, robot α communicates the category labels (class) and poses (in robot α's frame) of all the detected objects to robot β. We assume that the initial pose of each robot is known to all the robots, hence, given the initial pose of robot α, robot β is able to transform the communicated object poses from robot α's frame to its own frame.[2] For each object in the list communicated by robot α, robot β finds the nearest object in its map, having the same category label and within 2σ distance. If such an object exists, it is added to the list of *shared* objects: this is the set of objects seen by both robots. The list of shared objects contains pairs $(\boldsymbol{o}_k^\alpha, \boldsymbol{o}_l^\beta)$ and informs the robots that the poses $\boldsymbol{o}_k^\alpha$ and $\boldsymbol{o}_l^\beta$ correspond to the same physical object, observed by the two robots. For this reason, in the optimization we enforce the relative pose between $\boldsymbol{o}_k^\alpha$ and $\boldsymbol{o}_l^\beta$ to be zero.

We remark that, while before the first rendezvous the robots α and β have different reference frames, the object-object factors enforce both robots to have a single shared reference frame, facilitating future data association.

Next we show the experimental evaluation which includes realistic Gazebo simulations and field experiments in a military test facility.

[2]The knowledge of the initial pose is only used to facilitate data association but it is not actually used during pose graph optimization. We believe that this assumption can be easily relaxed but for space reasons we leave this task to future work.

4.4 Experiments

We evaluate our approach in large scale simulations (Section 4.4.1) and field tests (Section 4.4.2). The results demonstrate that the proposed approach is accurate, scalable, robust to noise, and requires less memory and communication bandwidth.

4.4.1 Simulation Results: Distributed Object based SLAM

In this section we characterize the performance of the DGS algorithms associated with our object-based model in a simulated environment. We test the resulting multi robot object-based SLAM approach in terms of scalability in the number of robots and sensitivity to noise.

Simulation setup and performance metrics. We consider two scenarios, the `25 Chairs` and the `House`, which we simulated in Gazebo. In the `25 Chairs` scenario, we placed 25 chairs as objects on a grid, with each chair placed at a random angle. In the `House` scenario, we placed furniture as objects in order to simulate a living room environment. Fig. 36 shows the two scenarios. Unless specified otherwise, we generate measurement noise from a zero-mean Gaussian distribution with standard deviation $\sigma_R = 5^\circ$ for the rotations and $\sigma_t = 0.2$m for the translations. Six robots are used by default. Results are averaged over 10 Monte Carlo runs.

We use the *absolute translation error* (ATE*) and *absolute rotation error* (ARE*) of the robot and landmark poses with respect to the centralized estimate as error metric. These two metrics are formally defined below.

Absolute Translation Error (ATE).* Similar to the formulation by Sturm et al. [115], the average translation error measures the absolute distance between the trajectory and object poses estimated by our approach versus the centralized GN method. The ATE* is defined as:

$$ATE^* = \left(\frac{1}{\sum_{\alpha \in \Omega} n_\alpha} \sum_{\alpha \in \Omega} \sum_{i=1}^{n_\alpha} \| \boldsymbol{t}_{\alpha_i} - \boldsymbol{t}^*_{\alpha_i} \|^2 \right)^{\frac{1}{2}} \tag{26}$$

where $\boldsymbol{t}_{\alpha_i}$ is the position estimate for robot α at time i, $\boldsymbol{t}^*_{\alpha_i}$ is the corresponding estimate

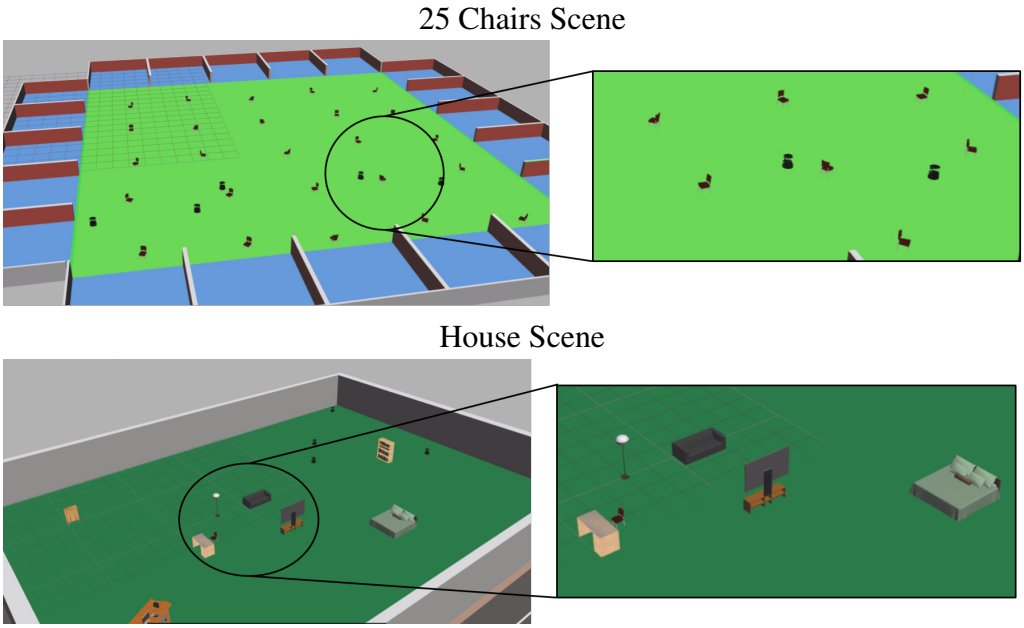


Figure 36: Multi robot object-based SLAM in Gazebo: the `25 Chairs` and `House` scenarios simulated in Gazebo.

from GN, and n_α is the number of poses in the trajectory of α. A similar definition holds for the object positions.

Absolute Rotation Error (ARE).* The average rotation error is computed by evaluating the angular mismatch between the (trajectory and objects) rotations produced by the proposed approach versus a centralized GN method:

$$ARE^* = \left(\frac{1}{\sum_{\alpha \in \Omega} n_\alpha} \sum_{\alpha \in \Omega} \sum_{i=1}^{n_\alpha} \| \text{Log} \left((\boldsymbol{R}^*_{\alpha_i})^\mathsf{T} \boldsymbol{R}_{\alpha_i} \right) \|^2 \right)^{\frac{1}{2}} \tag{27}$$

where $\boldsymbol{R}_{\alpha_i}$ is the rotation estimate for robot α at time i, $\boldsymbol{R}^*_{\alpha_i}$ is the corresponding estimate from GN. A similar definition holds for the object rotations.

Accuracy in the number of robots. Fig. 37 compares the object locations and trajectories estimated using multi-robot mapping and centralized mapping for the two scenarios. Videos showing the map building for the two scenarios are available at: `https://youtu.be/nXJamypPvVY` and `https://youtu.be/nYm2sSHuGjo`.

Table 4 reports the number of iterations and our accuracy metrics (cost, ATE*, ARE*) for increasing number of robots. The table confirms that the distributed approach is nearly

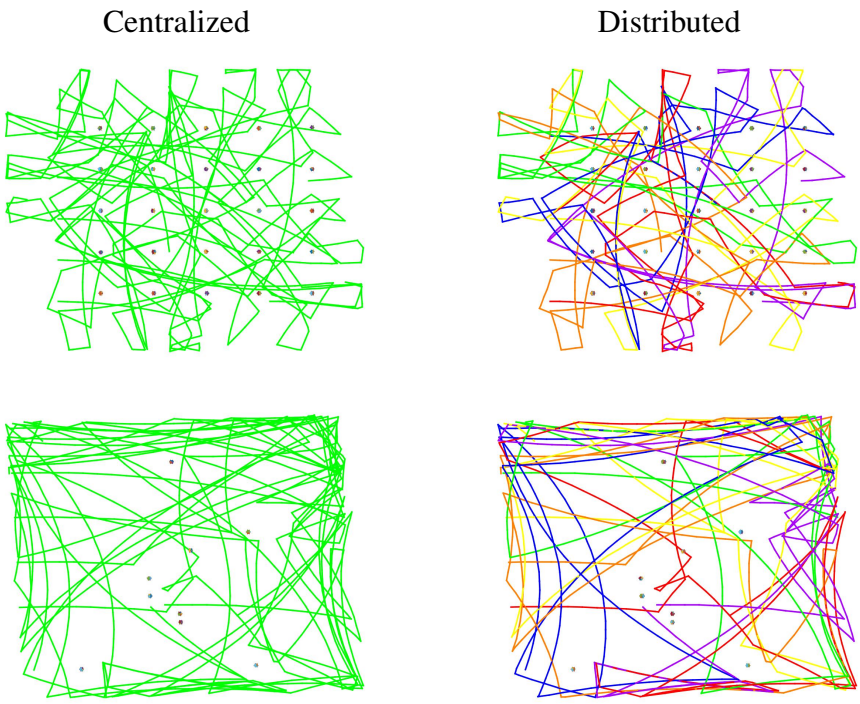


Figure 37: Shows the trajectories of the six robots and object locations (shows as dots) estimated using centralized mapping and multi-robot mapping for `25 Chairs` (top) and `House` scenario (bottom).

Table 4: Number of iterations, cost, ATE* and ARE* of our approach compared to the centralized Gauss-Newton method for *increasing number of robots*. ATE* and ARE* are measured using $\eta = 10^{-1}$ as stopping condition. Measurement noise is generated from a Gaussian distribution with standard deviation $\sigma_R = 5^\circ$ for the rotations and $\sigma_t = 0.2$m for the translations. Results are averaged over 10 Monte Carlo runs.

#Robots	Distributed Gauss-Seidel				Cent.	ATE* (m)		ARE* (deg)	
	$\eta=10^{-1}$		$\eta=10^{-2}$		GN Cost	Poses	Lmrks.	Poses	Lmrks.
	#Iter	Cost	#Iter	Cost					
2	5.0	56.1	9.0	56.0	54.7	1.5e-03	8.0e-04	2.1e-01	2.8e-01
4	5.0	118.0	8.0	117.9	113.8	9.7e-04	7.5e-04	2.0e-01	2.8e-01
6	5.0	166.6	7.0	166.5	160.9	3.1e-03	2.1e-03	3.3e-01	4.0e-01

as accurate as the centralized Gauss-Newton method and the number of iterations does not increase with increasing number of robots, making our approach scalable to large teams. Usually, few tens of iterations suffice to reach an accurate estimate. Note that even when the cost of the DGS method is slightly higher than GN, the actual mismatch in the pose

Table 5: Number of iterations, cost, ATE* and ARE* of our approach compared to a centralized Gauss-Newton method for *increasing measurement noise* in `25 Chairs` scenario with 6 robots. ATE* and ARE* are measured using $\eta = 10^{-1}$ as stopping condition.

Measurement noise		Distributed Gauss-Seidel				Cent.	ATE* (m)		ARE* (deg)	
		$\eta = 10^{-1}$		$\eta = 10^{-2}$		GN	Poses	Lmrks.	Poses	Lmrks.
$\sigma_r(^\circ)$	σ_t(m)	#Iter	Cost	#Iter	Cost	Cost				
1	0.05	5.0	12.7	6.0	12.7	12.5	1.8e-04	1.3e-04	7.5e-02	9.0e-02
5	0.1	5.0	166.6	7.0	166.5	160.9	3.1e-03	2.1e-03	3.3e-01	4.0e-01
10	0.2	5.0	666.2	8.0	665.9	643.4	1.3e-02	8.8e-03	6.7e-01	8.1e-01
15	0.3	6.0	1498.3	10.0	1497.8	1447.2	3.0e-02	2.1e-02	1.0e+00	1.2e+00

Figure 38: Objects from BigBird dataset used in Field Experiments

estimates is negligible (in the order of millimeters for positions and less than a degree for rotations).

Sensitivity to measurement noise. We further test the accuracy of our approach by evaluating the number of iterations, the cost, the ATE* and ARE* for increasing levels of noise in `25 Chairs` scenario with 6 robots. Table 5 shows that our approach is able to replicate the accuracy of the centralized Gauss-Newton method, regardless of the noise level.

4.4.2 Field Experiments: Distributed Object based SLAM

We tested our approach on field data collected by two Jackal robots (Fig. 39) moving in a military training facility. We scattered the environment with a set of objects from the BigBird dataset [112], shown in Fig. 38. Each robot is equipped with an Asus Xtion sensor

Figure 39: (Left) Clearpath Jackal robot used for the field tests: platform and sensor layout; (right) snapshot of the test facility and the Jackal robots.

Figure 40: Shows YOLO object detection snapshots in three difference scenes, (l to r) stadium, house, UW scene 2.

and uses wheel odometry to measure its ego-motion.

We evaluated our approach in two different scenarios, the `stadium` and the `house`. We did two runs inside the `stadium` (`Stadium-1` & `Stadium-2`) and one run in the `house` with objects randomly spread along the robot trajectories. `Stadium` scenario datasets were collected in an indoor basketball stadium with the robot trajectories bounded in a roughly rectangular area. `House` scenario dataset was collected around the living room and kitchen area of a house.

Object Detection. We used 12 objects from the BigBird dataset in all three runs. The two-stage process of object detection (semantic verification) followed by pose estimation (geometric verification) ensured that we do not add false positive detections. Our current distributed optimization technique (DGS) is not robust to outliers. The detection thresholds can be further relaxed when using robust pose graph optimization techniques.

In the first run (`Stadium-1`), 6 objects were added to the map out of 12 objects kept in the environment. Similarly 5 objects were detected in the other two runs. Fig. 40 shows the bounding box snapshots of the detected object in three different scenes. Videos showing YOLO object detection results on UW Scenes2 dataset [75] is available at `https://`

youtu.be/urZiIJK2IYk and https://youtu.be/-F6JpVmOrc0.

Table 6: Memory and communication requirements for our object based approach (Obj) as compared to Point cloud based approach (PCD) on field data.

Scenario	Avg. Per-Robot Memory Req. (MB)		Avg. Comm. Bandwidth Req. (MB)	
	PCD	Obj	PCD	Obj
Stadium-1	1.2e+03	1.9e+00	1.9e+01	1.5e-05
Stadium-2	1.4e+03	1.9e+00	1.4e+01	1.1e-05
House	2.1e+03	1.9e+00	1.6e+01	1.3e-05

Memory Requirements. Table 6 compares the average memory requirement per robot to store a dense point cloud map (PCD) with respect to storing a object-based map (Obj). The table also compares the average communication requirements in the case of dense point cloud map and object-based map.

Per-robot memory requirement in the case of dense point cloud is computed as $n_f KC$ where n_f is the number of frames, K is the number of points per frame and C is the memory required to store each point. In the case of object level map, it is computed as $n_o PC$ where n_o is the number of object models and P is the average number of points in each object model. Table 6 shows that, as expected, the per-robot memory requirement is orders of magnitude smaller with our object-based map as compared to point-cloud-based maps.

When using point clouds, the robots are required to send at least one frame at every rendezvous to estimate their relative pose. So the average communication for dense point cloud map is computed as $n_c KC$ where n_c is the number of rendezvous, K is the number of points per frame and C is the memory required to send each point. Communication in the case of our object-based map requires sending object category and object pose; a upper bound can be computed as $n_o L$ where n_o is the number of objects and L is the memory required to store category label and pose of an object. Table 6 confirms that our approach provides a remarkable advantage in terms of communication burden as it requires transmitting 6 orders of magnitude less than a point-cloud-based approach.

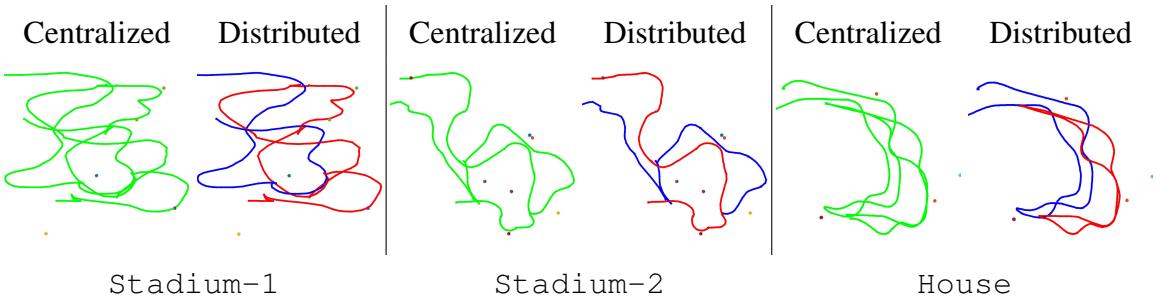


Figure 41: Field tests: estimated trajectories for the our algorithm (distributed Gauss-Seidel) and for the centralized Gauss-Newton method [35]. Trajectories of the two robots are shown in red and blue.

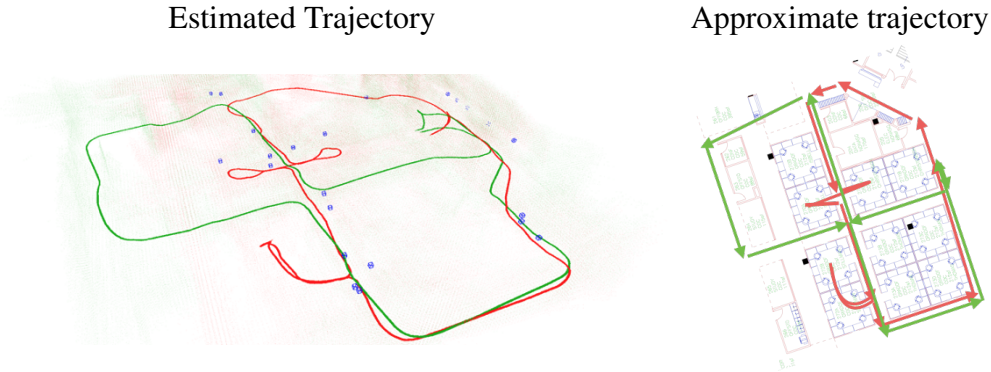


Figure 42: Lab test: estimated trajectory for our algorithm (distributed Gauss-Seidel) and approximate trajectory marked on the blue-print. Trajectories of the two robots are shown in red and green. Object landmarks are shown in blue.

Accuracy. Fig. 41 shows the trajectories of the two robots in three runs. The figure compares our approach and the corresponding centralized estimate. Fig. 42 shows the trajectories of the two robots for a run in Georgia Tech IRIM lab. Quantitative results are given in Table 6 and Table 7, which reports the cost attained by the our approach, the number of iterations, ATE*, ARE* as compared to the centralized approach. The table confirms that the distributed approach is nearly as accurate as the centralized Gauss-Newton method and requires very few iterations to compute a good estimate.

Table 7: Number of iterations, cost, ATE* and ARE* of our approach as compared to centralized Gauss-Newton method for Field data.

Scenario	Initial	Distributed Gauss-Seidel				Centralized	ATE (m)		ARE (deg)	
		$\eta_r = \eta_p = 10^{-1}$		$\eta_r = \eta_p = 10^{-2}$		GN	Poses	Lmrks.	Poses	Lmrks.
	Cost	#Iter	Cost	#Iter	Cost	Cost				
Stadium-1	120.73	5.0	1.1e-09	5.0	1.1e-09	1.6e-10	1.9e-10	1.9e-10	1.4e-03	1.2e-04
Stadium-2	310.24	5.0	4.5e-12	8.0	4.4e-12	3.5e-13	2.1e-03	2.2e-03	1.2e-02	1.4e-02
House	43.59	5.0	1.1e-03	6.0	1.0e-03	8.4e-04	4.4e-02	6.2e-02	4.3e-01	4.9e-01

4.5 Main Experimental Insights

In the previous chapter (Chap 3), we proposed a distributed Gauss-Seidel method, which reduces the communication burden of distributed SLAM from quadratic to linear in the number of locations co-observed by the robots. However, the work [23], as most related works, requires the exchange of point clouds among the robots, to estimate relative poses during rendezvous. This communication burden is unnecessary, as it leads to exchanging a large amount of uninformative points, and quickly becomes impractical in presence of bandwidth constraints.

In this chapter we address this issue by using an object-based representation. Objects provide a suitable abstraction level, and provide a natural tool to compress large point clouds into a semantically meaningful compact representation. In our system, the robots perform local computation to detect objects and compute their pose. We leverage recent progress in object detection using deep learning: this allows us to reliably detect objects in RGB images at high frame rate. Then, during rendezvous the robots only need to exchange the observed object instances and the measured object poses. This allows the robots to greatly minimize the amount of data exchanged with the teammates.

Experimental evidence shows that our approach leads to a remarkable reduction in the memory footprint (3 orders of magnitude less) and in the communication requirements (6 orders of magnitude less communication), enabling operation in severely bandwidth-constrained scenarios. The experiments also show that our object-based distributed SLAM

approach is as accurate as a standard centralized solver and is able to tolerate a large amount of measurement noise.

4.6 Conclusions

We proposed a Distributed Object-based SLAM approach that uses object landmarks in a multi robot mapping framework. We showed that this approach (i) reduces the memory requirement and information exchange among robots, (ii) is as accurate as the centralized estimate, (iii) scales well to large number of robots and (iv) is resistant to noise.

However it can be challenging to store a model of all the object instances due to large intra-class variation. Searching through all the object models for object pose estimation can be computationally demanding. It won't generalize to new unseen instances of the same object category as well. Therefore, in the next two chapters, we extend this work to the case where object models are previously unknown and are modeled jointly with Distributed Object based SLAM.

CHAPTER V

OBJECT BASED SLAM WITH JOINT OBJECT MODELING AND MAPPING

5.1 Introduction

Service robots operating in human environments require a variety of perceptual capabilities, including localization, mapping, object recognition, and modeling. Each environment may have a unique set of objects, and a set of object models may not be available a-priori. Even if the object models are available, it can be challenging to store a model of all the object instances due to large intra-class variation.

However, recently the performance of category level object detectors have improved considerably, providing real-time bounding box object detections for a large number of object categories commonly found in indoor environments [100]. Simultaneous Localization and Mapping (SLAM) is also required for service robot to be able to map new environments and navigate within them.

In this chapter, we leverage the improvement in object detectors and propose an approach to online object modeling, and show how to combine this with a SLAM system. The benefits are twofold: an object model database is produced in addition to the map, and the modeled objects are used as landmarks for SLAM, producing improved mapping results.

Related Work. Localization and navigation are two basic problems in the area of mobile robotics. Map based navigation is a commonly used method to navigate from one point to another, where maps are commonly obtained using simultaneous localization and mapping (SLAM). Durrant-Whyte and Bailey provide a survey of the SLAM literature and the state of art in [38, 10]. Recent SLAM systems use graph optimization technique to jointly

estimate the entire robot trajectory and landmarks using sparse optimization techniques [42, 34, 123].

However, traditional feature based maps are composed of low level primitives like points and lines which are mostly used to model space based on its geometric shape [32]. These maps lack the semantic information necessary for performing wider range of tasks, which may require higher level representation such as object models. Dense metric maps also have a lot of redundant information like 3D points in floor and uninformative textured surfaces which do not facilitate robot localization. In addition, dense maps built using depth cameras can be memory intensive as well.

In contrast, maps augmented with objects confer a number of advantages for mapping the environment. Features, e.g., objects and planes, can be represented using a compact representation and provide a richer description of the environment. It is simpler to include prior constraints at the object level than at the lower feature level. The semantic information available for an object provides better cues for data association as compared to a 3D point cloud. An object would not be represented by a 3D point but rather by a 3D point cloud. Joint optimization over all the camera poses and objects is computationally cheaper than the joint optimization over all the 3D points and cameras since, as there are many fewer objects compared to the number of 3D points in a map.

Hence, *semantic mapping* has gathered a lot of interest from the robotics community. Kuipers [72] modeled the environment as a spatial semantic hierarchy, where each level expresses states of partial knowledge corresponding to different level of representations. Ranganathan and Dellaert [98] presented a 3D generative model for representing places using objects. The object models are learned in a supervised manner. Nuchter and Hertzberg [92] described an approach to semantic mapping by creating a 3D point cloud map of the environment and labeling points using the different semantic categories like floor, wall, ceiling or door. Pronobis et al. [97] proposed a complete and efficient representation of indoor spaces including semantic information. They use a multi-layered semantic mapping

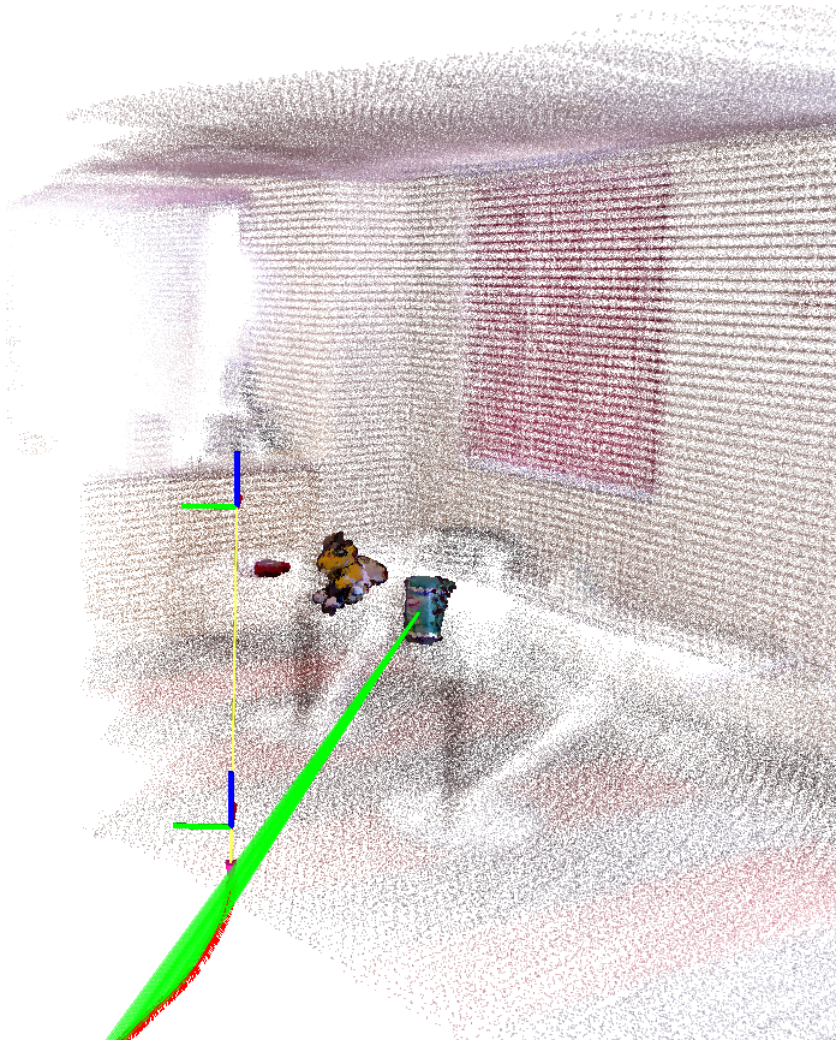


Figure 43: Snapshot of the process at one instant. The robot trajectory is shown in red. Green lines add constraints between the modeled object landmarks and the robot poses they are seen from. Light background shows the aggregated map cloud generated using the current SLAM solution.

representation to combine information about the existence of objects in the environment with knowledge about the topology and semantic properties of space such as room size, shape and general appearance.

Some recent semantic mapping work has focused on using higher level landmarks such as objects. Rogers et al. recognize door signs and read their text labels such as room numbers, which are used as landmarks in SLAM [101]. Trevor et al. used planar surfaces corresponding to walls and tables as landmarks in a mapping system [127]. More recently, the SLAM++ system proposed by Salas-Moreno et al. [108] trained domain specific object detectors corresponding to repeated objects like tables and chairs. The learned detectors are integrated inside the SLAM framework to recognize and track those objects resulting in semantic map. Similarly Kim et al. [67] uses learned object models to reconstruct dense 3D models from single scan of the indoor scene.

Object discovery and scene understanding are also related to our approach. Karpathy et al. [62] decompose a scene into candidate segments and ranks them according to their objectness properties. Collet et al. used domain knowledge in the form of metadata and use it as constraints to generate object candidates [27]. Using RGBD sensor, Koppula et al. [69] used graphical models capturing various image feature and contextual relationship to semantically label the point cloud with object classes and used that on a mobile robot for finding objects in a large cluttered room. Hoiem and Savarese [54] did a survey of additional recent work in the area of 3D scene understanding and 3D object recognition.

Contribution. All of the above approaches either learn object models in advance, or discover them in a batch process after the robot has scanned the environment and generated a 3D model or a video sequence. In contrast, our main contribution is to integrate object modeling and mapping in a SLAM framework, following an online learning paradigm. Considering the advantages of object augmented maps, we too use objects as landmarks in addition to other features. We compare the performance of our algorithm with the state of art RGB-D mapping approaches and show that our approach is nearly as accurate as the

state of art RGB-D mapping approach while using less memory than those approaches.

Section 5.2 introduces the additional mathematical notation and formalizes the problem of Object-SLAM with joint object modeling and mapping. Section 5.3 presents the implementation details of our system.

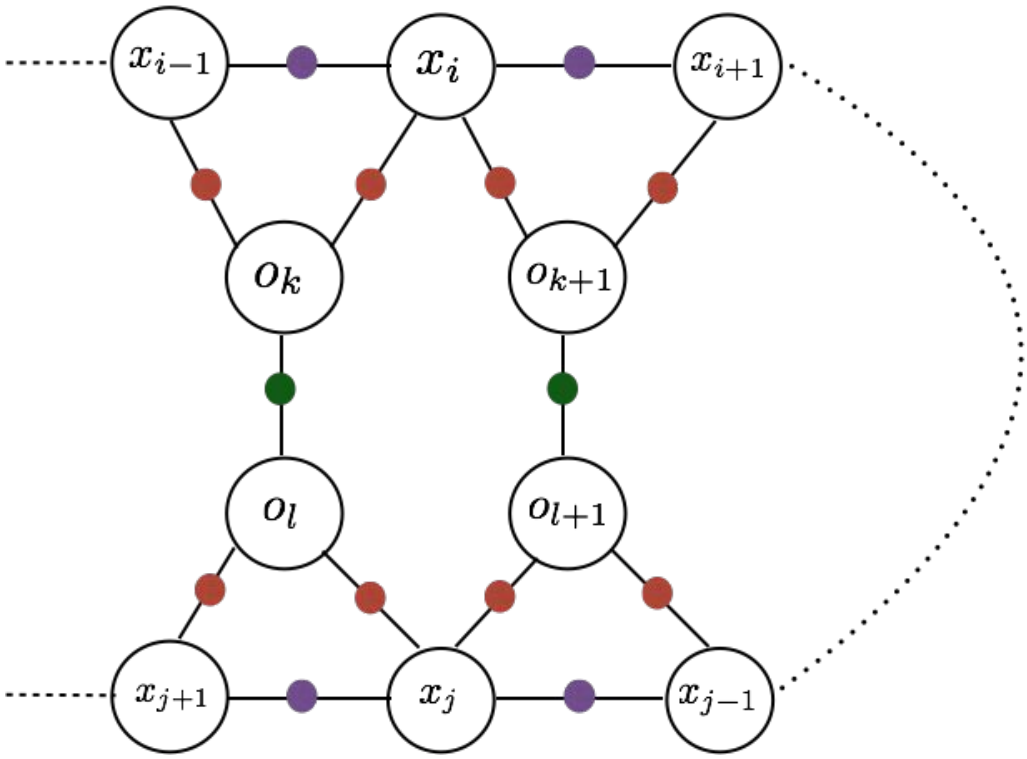


Figure 44: Factor graph representation of Object based SLAM with Joing Object Modeling and Mapping. $\boldsymbol{x}_{i-1}$,$\boldsymbol{x}_i$,$\boldsymbol{x}_{i+1}$ denotes the trajectory poses assumed by a robot at time $i-1$,i,$i+1$ respectively. The pose of the k^{th} object as estimated by the robot is denoted with $\boldsymbol{o}_k$. Green dots shows object-object loop closure factor whereas orange and purple dots show object-pose factors and odometry factors respectively.

5.2 Problem Formulation: Object-SLAM with Joint Object Modeling and Mapping

We consider a system, where a robot estimates its own trajectory using local measurements and also estimates the pose of a set of objects in the environment. We model the trajectory as a finite set of poses; the trajectory of robot α is $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots]$. In addition, we denote with $\boldsymbol{o}_k \in \mathrm{SE}(3)$ the pose of the k^{th} object as estimated by the same robot.

Measurements. We assume that a robot acquires three types of relative pose measurements: odometry measurement, object-pose measurement and object-object loop closure measurements. The *odometry measurements* consist of the fused robot motion measurements, which constrain consecutive robot poses (e.g., $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i+1}$ in Fig. 44). The *object-pose measurement* constrains robot poses with the corresponding visible object landmarks (e.g., $\boldsymbol{x}_i$ and $\boldsymbol{o}_k$ in Fig. 44). The *object-object loop closure measurement* relate the object landmarks corresponding to the same physical object but added twice due to the odometry drift. We use sparse feature matching followed by ICP based refinement to estimate the object-object loop closure measurement.

The odometry, object-pose and object-object measurements follows the same measurement model of eq. (1). Specifically, for the object-pose measurements, if a robot at time i and at pose $\boldsymbol{x}_i$ observes an object at pose $\boldsymbol{o}_k$, then the corresponding measurement $\bar{\boldsymbol{z}}^{\boldsymbol{x}_i}_{\boldsymbol{o}_k}$ measures the relative pose between $\boldsymbol{x}_i$ and $\boldsymbol{o}_k$. Similarly we denote the loop closure measurement between object poses $\boldsymbol{o}_k$ and $\boldsymbol{o}_l$ as $\bar{\boldsymbol{z}}^{\boldsymbol{o}_k}_{\boldsymbol{o}_l}$.

In the following, we denote with $\mathcal{M}_I$ the set of odometry measurements for a robot. Similarly the set of object-pose measurements for a robot is denoted as $\mathcal{M}_O$. The set of object-object loop closure measurements is denoted with $\mathcal{M}_S$. The set of all available measurements is then $\mathcal{M} = \mathcal{M}_I \cup \mathcal{M}_O \cup \mathcal{M}_S$.

ML trajectory and objects estimation. Let us collect the trajectories and object poses in a (to-be-estimated) set of robot poses $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots]$ and set of object poses $\boldsymbol{o} = [\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots]$. The ML estimate for $\boldsymbol{x}$ and $\boldsymbol{o}$ is defined as the maximum of the measurement likelihood:

$$\boldsymbol{x}^\star, \boldsymbol{o}^\star = \arg\max_{\boldsymbol{x},\boldsymbol{o}} \prod_{(\boldsymbol{x}_i,\boldsymbol{x}_{i+1})\in\mathcal{M}_I} \underbrace{\mathcal{L}(\bar{\boldsymbol{z}}^{\boldsymbol{x}_i}_{\boldsymbol{x}_{i+1}} \mid \boldsymbol{x})}_{\text{odometry factors}} \prod_{(\boldsymbol{x}_i,\boldsymbol{o}_k)\in\mathcal{M}_O} \underbrace{\mathcal{L}(\bar{\boldsymbol{z}}^{\boldsymbol{x}_i}_{\boldsymbol{o}_k} \mid \boldsymbol{x}, \boldsymbol{o})}_{\text{object-measurement factors}} \prod_{(\boldsymbol{o}_i,\boldsymbol{o}_j)\in\mathcal{M}_S} \underbrace{\mathcal{L}(\bar{\boldsymbol{z}}^{\boldsymbol{o}_i}_{\boldsymbol{o}_j} \mid \boldsymbol{x}, \boldsymbol{o})}_{\text{object-object loop closure factors}} \tag{28}$$

where we used the same assumptions on measurement noise as in Section 3.2. Defining

$\mathcal{X} = \boldsymbol{x} \cup \boldsymbol{o}$, we rewrite eq. (28) as:

$$\mathcal{X}^{\star} = \arg\max_{\mathcal{X}} \prod_{(\alpha_i, \beta_j) \in \mathcal{M}} \mathcal{L}(\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i} \mid \mathcal{X}) \tag{29}$$

The given optimization problem is optimized using iSAM2 [61].

The next section presents the implementation details of our Object-SLAM with joint object modeling and mapping system.

5.3 Implementation Details: Object based SLAM with Joint Object Modeling and Mapping

Ego-Motion Estimation. As the robot moves around in an indoor environment, we continuously map the environment using the SLAM system presented by Trevor et al. [127, 128]. A robot collects 3D scans using Velodyne 32E, RGB-D scans using an Orbbec Astra sensor, and inertial measurements using IMU and odometry measurements using wheel sensors. Fig. 22 shows the sensor layout on a Jackal robot. Relative pose estimates from all the sensors are fused together using OmniMapper[128] to estimate robot's ego-motion[1]. Fig. 3 shows the overview of the egomotion estimation pipeline.

Specifically, 3D scans from the Veldoyne 32E are used to compute relative pose with respect to the previous scan using GICP (generalized iterative closest point [110]). Inertial measurements from IMU are fused with Wheel Odometry measurements to generate IMU corrected odometry estimates. Relative pose estimates using RGB-D scans are computed using ORB-SLAM [85]. Relative pose estimates are then fused using OmniMapper.

Per Frame Segmentation. We assume that in indoor environments, objects are supported by homogeneous planes such as walls, floors, table tops. We consider all the homogeneous planes as planar landmarks and the remaining non-planar regions as the regions corresponding to potential objects of interest. To segment such scenes, we employ the organized connected component segmentation approach of Trevor et. al [129]. Given a point

[1] `https://github.com/CognitiveRobotics/omnimapper`

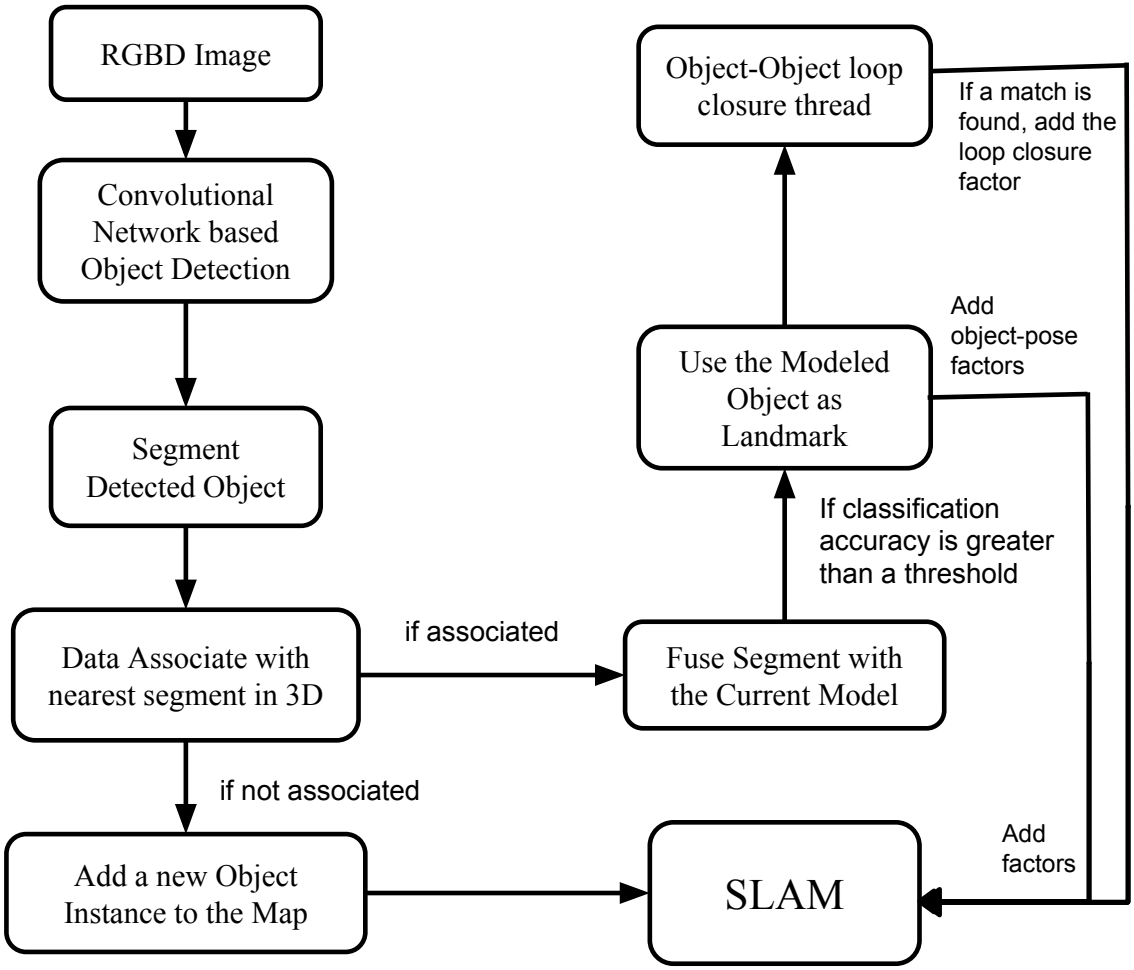


Figure 45: Flowchart of Object based SLAM with Joint Object Modeling and Mapping

cloud, surface normals are computed for each point in each frame using the technique of Holzer et. al[55]. Using these normals, connected component corresponding to surfaces with smoothly varying normals are found. Least squares plane fit is used to recover planar segments having more than minimum number of inliers and low curvature.

Points corresponding to non-planar regions are further segmented using different segmentation techniques. We experiment with connected component segmentation [129] or graph based segmentation [40] to segment out objects. We use connected component segmentation considering its better runtime performance and comparable accuracy to graph based segmentation. The remaining clustered regions are further filtered to exclude clusters that are either too small or too large, to exclude noise and large furniture. In this work,

we consider only clusters that include at least 1000 points, and have a bounding box volume of less than 1 cubic meter. These values were determined empirically.

Object Detection. Each RGB frame (from RGBD) is passed to the YOLO v2 object detector [99, 100], which detects objects at 45 frames per second. We fine-tune the YOLO detector on a subset of objects from the *BigBird* dataset ([112]). The training dataset contains the object images in a clean background taken from different viewpoints and labeled images of the same objects taken by a robot in an indoor environment. During testing, we use a probability threshold of 0.3 to avoid false detections.

Object Modeling. The non-planar clusters from per-frame segmentation which are inside the detection bounding boxes are then associated with other modeled objects visible in the co-visible keyframes [85]. To data associate the current object segment with the other objects in the covisible keyframes, we use ICP-like matching score as proposed by Sunderhauf et al. [119]. Firstly, we find the candidate object landmarks by sorting the modeled objects in an ascending order of centroid distance from the object segment centroid. Then we perform a nearest neighbor search between the 3D points in the candidate object landmark and in the detection, and calculate the Euclidean distance between the associated point pairs. A detection is associated to an existing landmark if at least 50% of its 3D points have a distance of 2 cm or less. If none of the covisible object landmarks match, and distance to the nearest landmark bounding box is greater than 5 cm, then a new object landmark is added. Otherwise the new object segment is associated to the matched object landmark. If the IoU of the new object segment bounding box with the nearest object landmark is greater than zero but less than 50% of its 3D points have a distance of 2 cm or less with any of the candidate object landmarks, then the new object segment is neither associated nor added as a new landmark and the segment is thrown away.

Object Representation. Each object model is represented semantically using the per-class probabilities provided by the YOLO v2 detector. Each new detection consists of a vector of per-class probabilities. When a new detection is associated with one of the object

landmarks, the per-class probabilities of the object landmark is updated using running average technique. If the object landmark is visible from n poses, $\mathbf{S}$ is the current accumulated per-class probabilities and s is the new detection per-class probabilities, then the updated probabilities $\mathbf{S}$ and n is given as:

$$\mathbf{S} = \frac{n \times \mathbf{S} + s}{n+1}, n = n + 1 \tag{30}$$

If the given object landmark is visible from more than 5 frames and the largest class probability of the object landmark is greater than 0.5, then we add the object landmark into the SLAM factor graph. The category of that object landmark is set to the object class having the largest class probability.

The object landmark $\boldsymbol{o}_k$ is initialized at the pose of the first frame x_k in which it is visible from. The corresponding object pose measurement for the n^{th} frame, $\bar{\boldsymbol{z}}_{\boldsymbol{o}_k}^{\boldsymbol{x}_n}$ is estimated using ICP initialized using the transformation between the n^{th} frame and the first frame x_k in which the object is visible from. The factor corresponding to the object pose measurement is then added to the factor graph.

If an object is added to the factor graph, we pad the object bounding box by 20 cm in all direction and included all the points in this extended bounding box. The additional padded points serve the purpose of context which helps when estimating the transformation between two object landmarks in the case of symmetrical objects and it also helps in disambiguating the same objects in different locations in the environment (eg. chair of the same model placed in different areas). However the value of padding is determined empirically and can be further explored in future research. The points in extended bounding box are converted to TDF representation for establishing correspondence between different object landmarks using 3DMatch [139].

Object-Object Loop Closure. In a parallel thread, each object landmark which is added to the factor graph is matched to all the other landmarks having the same object category name. Correspondences are estimated using 3DMatch [139]. The estimated relative

pose given the sparse correspondences is then further refined using ICP. If the average euclidean distance of the transformed points in one object is less than 10^{-3} to the points in the matching object, we consider the matching to be successful. The resulting transformation is then used as the object-object loop closure measurement $\bar{z}_{o_l}^{o_k}$. The corresponding factor is added to the factor graph and optimized using iSAM2 [61]. Fig. 45 shows the complete flowchart.

Next we show the experimental evaluation which includes tests on standard and large-scale datasets.

5.4 Experiments

We evaluate our approach on small-scale standard SLAM datasets (UW RGB-D Scenes v2[74] and TUM[115]). We also evaluate our approach on large scale robot datasets captured using Jackal robot in different indoor environments. We compare the performance of our approach against other state of art RGB-D mapping approaches like ORB-SLAM2[85], ElasticFusion[138] and Kintinuous[137].

We use standard SLAM metrics like absolute trajectory error (ATE) and relative pose error (RPE) proposed by Sturm et al.[115] to compare the accuracy of the estimated trajectory. ATE directly measures the difference between points of the true and the estimated trajectory. ATE is evaluated using the root mean squared error over all time indices of the translation different between the estimated and groundtruth pose. RPE measures the local accuracy or the drift of the estimated trajectory over a fixed time interval.

We also compare the memory requirements of our approach as compared to ORB-SLAM2[85]. As compared to other RGB-D mapping approaches, ORB-SLAM2 maintains a sparse map of the environment and therefore requires the least memory when compared to other RGB-D mapping approaches. ElasticFusion maintains a surfel based map of the environment which is limited to a room sized environment since its complexity scales with the number of surfels in the map. Kintinuous maintains the RGB-D frames for the purpose

of loop closure and therefore does not scale well with time.

Table 8: ATE (in meters) comparison of Object-SLAM with joint object modeling and mapping (our approach), ORB-SLAM2, ElasticFusion and Kintinuous on UW RGB-D Scenes v2 dataset.

Scene ID	Object-SLAM	ORB-SLAM2	ElasticFusion	Kintinuous
1	0.009181	0.010568	0.015422	0.054868
2	0.008175	0.007374	0.010236	0.033052
3	0.005754	0.006336	0.015407	0.039809
4	0.008145	0.008141	0.019869	0.032991
5	0.028222	0.026408	0.039670	0.051828
6	0.024401	0.018720	0.024715	0.047619
7	0.011808	0.013378	0.035173	0.045526
8	0.017886	0.016270	0.037738	0.048659
9	0.010901	0.011902	0.007678	0.058484
10	0.009667	0.010983	0.005266	0.057244
11	0.013161	0.012608	0.009350	0.039200
12	0.008220	0.007880	0.008023	0.022030
13	0.006299	0.007455	0.006003	0.012381
14	0.018972	0.017364	0.009994	0.021740

5.4.1 UW RGB-D Scenes v2 dataset

We compare the performance of Object-SLAM with joint object modeling and mapping with ORB-SLAM2, ElasticFusion and Kintinuous on all 14 scenes in UW RGB-D Scenes v2 dataset. **Setup.** Since it is a handheld sequence, we use the local mapper in ORB-SLAM2 for ego-motion estimation for Object-SLAM. For object detection, we fine-tune the YOLO v2 detector on a subset of object classes (bowls, caps, cereal boxes, coffee

Table 9: RPE comparison of Object-SLAM with joint object modeling and mapping (our approach), ORB-SLAM2, ElasticFusion and Kintinuous on UW RGB-D Scenes v2 dataset.

Scene ID	Object-SLAM	ORB-SLAM2	ElasticFusion	Kintinuous
1	0.014328	0.015887	0.017999	0.113334
2	0.013608	0.012488	0.014283	0.077422
3	0.009940	0.009685	0.020132	0.095086
4	0.013148	0.013754	0.022681	0.084665
5	0.044958	0.040229	0.046880	0.117374
6	0.039250	0.031916	0.030698	0.116774
7	0.018957	0.020173	0.040176	0.096612
8	0.028336	0.025227	0.037073	0.116530
9	0.019602	0.020110	0.012689	0.110331
10	0.016941	0.018201	0.007244	0.122100
11	0.021976	0.020820	0.014971	0.088745
12	0.015514	0.015002	0.011368	0.079399
13	0.010886	0.012286	0.008649	0.033556
14	0.024059	0.021932	0.018983	0.055346

mugs, and soda cans) from UW RGB-D object dataset and UW RGB-D Scenes v1 dataset.

Results. Table 8 and 9 compare the ATE and RPE of Object-SLAM as compared to other RGB-D mapping approaches. Since most of the sequences are captured in room sized environment, there is no loop closure and therefore the accuracy of Object-SLAM is nearly the same as the accuracy of ORB-SLAM2. ElasticFusion and Kintinuous perform worse than ORB-SLAM2 on most of the sequences. Table 10 compares the memory footprint

Table 10: Memory footprint comparison of Object-SLAM with joint object modeling and mapping (our approach) and ORB-SLAM2 on UW RGB-D Scenes v2 dataset.

Scene ID	Object-SLAM	ORB-SLAM2
1	7.6 MB	28 MB
2	6.6 MB	28 MB
3	8.8 MB	28 MB
4	14 MB	26 MB
5	7.4 MB	38 MB
6	8.4 MB	59 MB
7	11 MB	44 MB
8	12 MB	54 MB
9	5.0 MB	21 MB
10	7.0 MB	19 MB
11	5.2 MB	21 MB
12	7.1 MB	22 MB
13	5.2 MB	6.4 MB
14	9.7 MB	16 MB

requirement of Object-SLAM as compared to ORB-SLAM2. We can see that Object-SLAM requires less memory than ORB-SLAM2 for all the sequences. Therefore for loopy small sequences like UW RGB-D Scenes v2 dataset, we showed that Object-SLAM reduces the memory requirement while maintaining the trajectory accuracy of the state of art RGB-D mapping approaches. Figures 46, 47 and 48 provide the qualitative comparison of the results produced by Object-SLAM, ORB-SLAM2 and ElasticFusion.

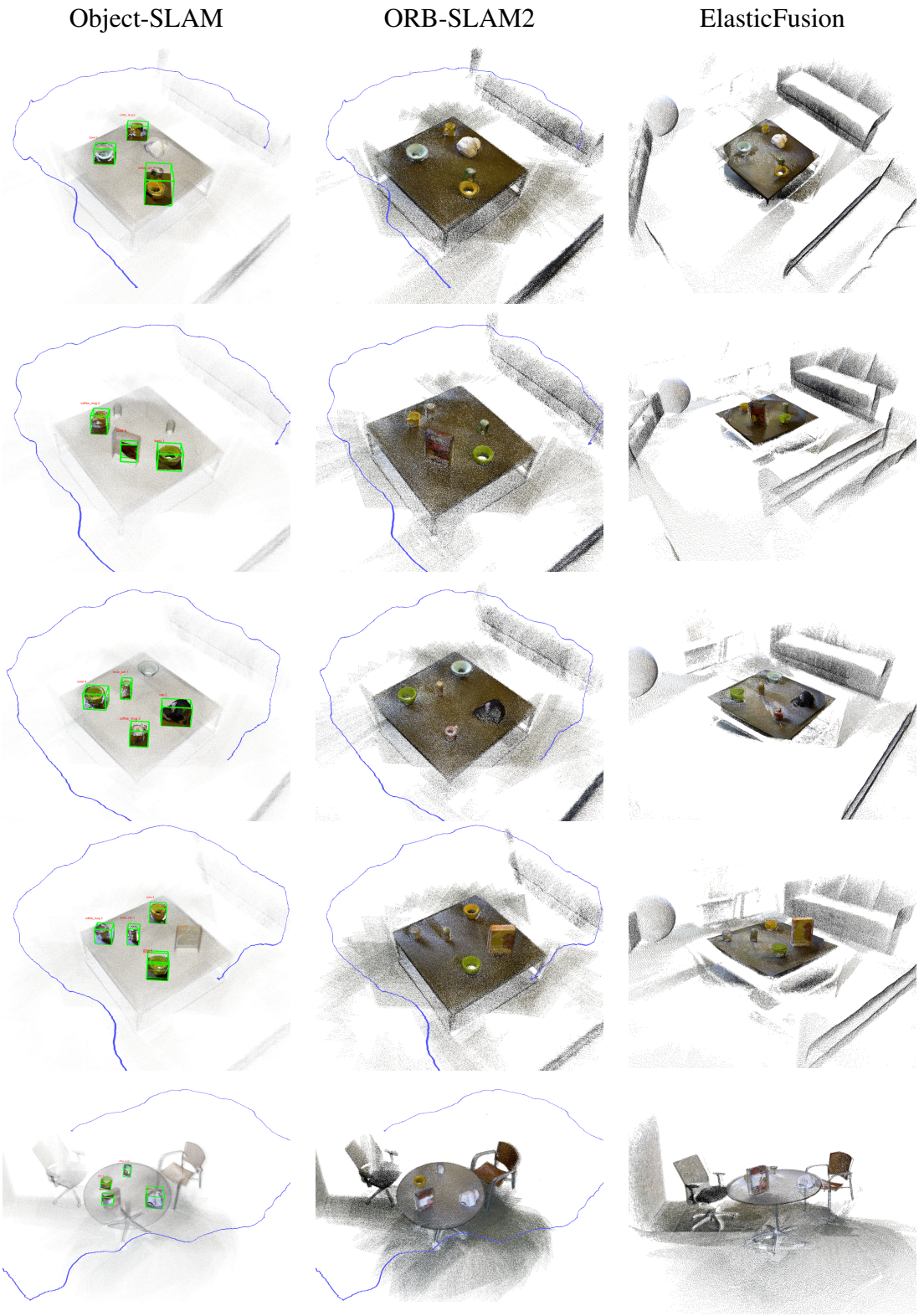


Figure 46: Qualitative comparison of the performance of Object-SLAM with joint object modeling and mapping (our approach), ORB-SLAM2 and ElasticFusion on UW RGB-D Scenes Dataset v2, scenes 1-5. For Object-SLAM and ORB-SLAM2 the trajectory is shown in blue color. For object SLAM, object bounding boxes are shown in green and category labels shown in red. Transparent point cloud background is just shown for visualization and is not used in the actual algorithm.

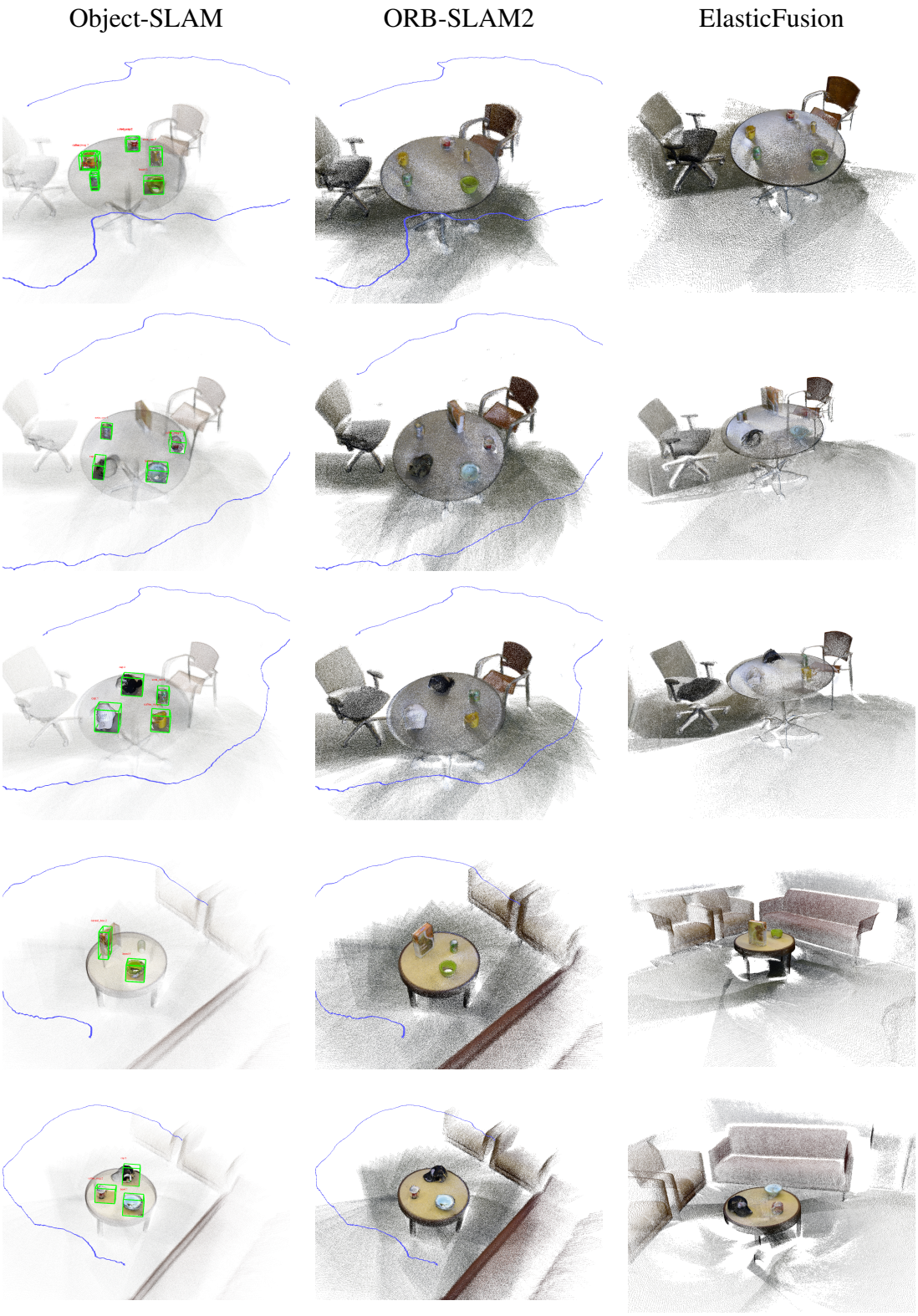


Figure 47: Qualitative comparison of the performance of Object-SLAM with joint object modeling and mapping (our approach), ORB-SLAM2 and ElasticFusion on UW RGB-D Scenes Dataset v2, scenes 6-12. For Object-SLAM and ORB-SLAM2 the trajectory is shown in blue color. For object SLAM, object bounding boxes are shown in green and category labels shown in red. Transparent point cloud background is just shown for visualization and is not used in the actual algorithm.

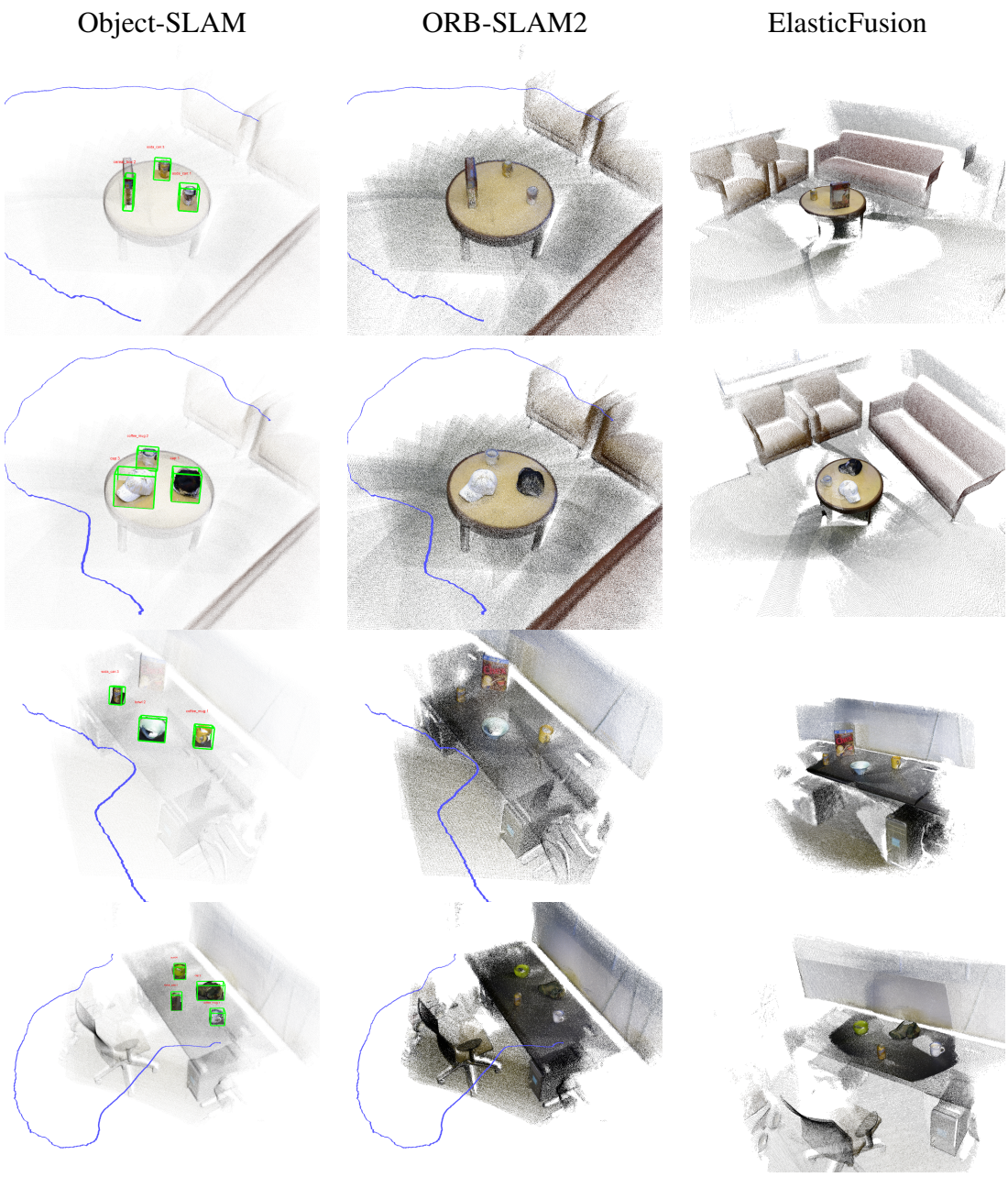


Figure 48: Qualitative comparison of the performance of Object-SLAM with joint object modeling and mapping (our approach), ORB-SLAM2 and ElasticFusion on UW RGB-D Scenes Dataset v2, scenes 11-14. For Object-SLAM and ORB-SLAM2 the trajectory is shown in blue color. For object SLAM, object bounding boxes are shown in green and category labels shown in red. Transparent point cloud background is just shown for visualization and is not used in the actual algorithm.

Table 11: ATE (in meters) comparison of Object-SLAM with joint object modeling and mapping (our approach), ORB-SLAM2, ElasticFusion and Kintinuous on TUM RGB-D dataset.

	Object-SLAM	ORB-SLAM2	ElasticFusion	Kintinuous
fr1/desk	0.015048	0.016059	0.022808	0.070849
fr1/desk2	0.023610	0.019193	0.065344	0.071
fr1/room	0.116864	0.043273	0.226645	0.313867
fr2/desk	0.015463	0.008687	0.095495	0.196952
fr2/xyz	0.004151	0.003751	0.012830	0.029
fr3/office	0.012157	0.011722	0.018876	0.097613

Table 12: RPE comparison of Object-SLAM with joint object modeling and mapping (our approach), ORB-SLAM2, ElasticFusion and Kintinuous on TUM RGB-D dataset.

	Object-SLAM	ORB-SLAM2	ElasticFusion	Kintinuous
fr1/desk	0.022871	0.022918	0.040841	0.127574
fr1/desk2	0.043759	0.043759	0.106668	0.150
fr1/room	0.156864	0.156864	0.318108	0.415113
fr2/desk	0.035788	0.035788	0.148241	0.292563
fr2/xyz	0.010794	0.010794	0.021372	0.031
fr3/office	0.024367	0.024367	0.037320	0.166917

5.4.2 TUM RGB-D dataset

We compare the performance of Object-SLAM with joint object modeling and mapping with ORB-SLAM2, ElasticFusion and Kintinuous on a subset of sequences in TUM RGB-D dataset.

Table 13: Memory footprint comparison of Object-SLAM with joint object modeling and mapping (our approach) and ORB-SLAM2 on TUM RGB-D dataset.

	Object-SLAM	ORB-SLAM2
fr1/desk	49 MB	12 MB
fr1/desk2	4 MB	27 MB
fr1/roo MB	41 MB	47 MB
fr2/desk	2.1 MB	20 MB
fr2/xyz	2 MB	15 MB
fr3/office	10 MB	67 MB

Setup. Similar to UW-RGBD dataset, we use the local mapper in ORB-SLAM2 for ego-motion estimation for Object-SLAM. For object detection, instead of finetuning the YOLO v2 detector, we use a pretrained COCO detector [80]. COCO detector is trained on 80 object classes commonly found in indoor and outdoor environments.

Results. Table 11 and 12 compares the ATE and RPE of Object-SLAM as compared to other RGB-D mapping approaches. Similar to UW-RGBD dataset, since most of the sequences are captured in room sized environment, there is no loop closure and therefore the accuracy of Object-SLAM is nearly the same as the accuracy of ORB-SLAM2. Elastic-Fusion and Kintinuous perform worse than ORB-SLAM2 on most of the sequences. Table 13 compares the memory requirement of Object-SLAM as compared to ORB-SLAM2. We can see that Object-SLAM requires less memory than ORB-SLAM2 for most of the sequences except for fr1/desk since the scene is very cluttered and therefore lot of object landmarks are added to the map.

5.4.3 Handheld Experiments

We also created some handheld sequences using Orbbec Astra RGB-D sensor in larger than room sized environment in CPL and IRIM labs.

Table 14: ATE (in meters) and RPE comparison of Object-SLAM with joint object modeling and mapping (our approach), Kintinuous and ORB-SLAM2 without loop closures w.r.t ORB-SLAM2 with loop closures. We compare against ORB-SLAM2 output since we don't have groundtruth trajectory estimates.

	Object-SLAM		Kintinuous		ORB-SLAM2 (no loop closure)	
	ATE	RPE	ATE	RPE	ATE	RPE
Handheld IRIM	0.552987	0.7876	0.499105	1.0437	1.8364	2.2739
Handheld CPL	0.215320	0.49313	0.62345	0.8	1.127	1.224

Table 15: Memory requirement comparison of Object-SLAM with joint object modeling and mapping (our approach) and ORB-SLAM2 for Handheld datasets collected using Orbbec Astra RGBD sensor in IRIM lab and CPL lab.

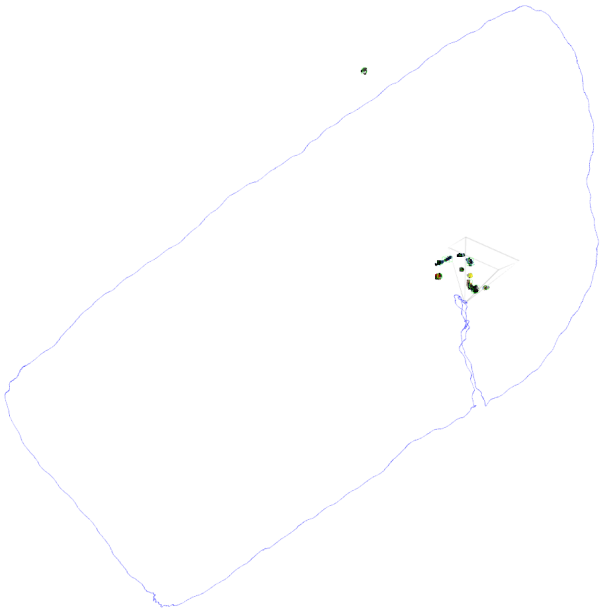
	Object-SLAM	ORB-SLAM2
Handheld IRIM	41 MB	596 MB
Handheld CPL	38 MB	737 MB

Setup. Similar to standard benchmark datasets, we use the local mapper in ORB-SLAM2 for ego-motion estimation. For object detection, we use the pretrained COCO detector as used in TUM RGB-D dataset.

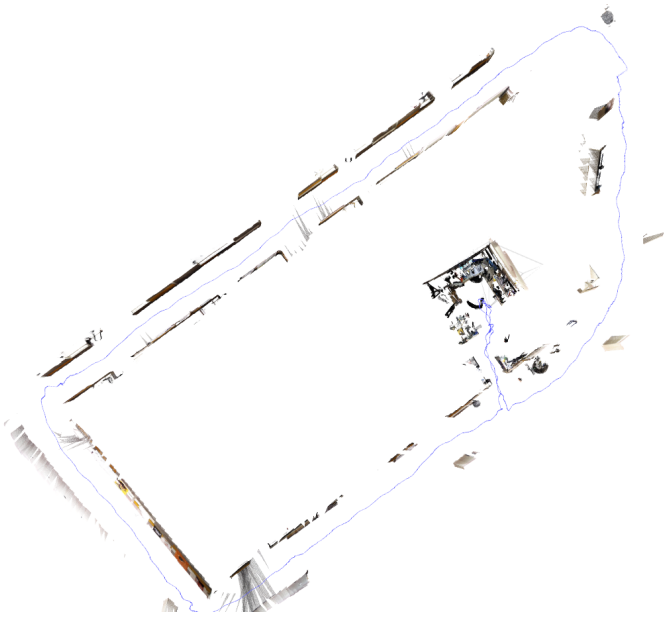
Results. Figure 49 and 50 provide the qualitative comparison of the results produced by Object-SLAM and ORB-SLAM2. Table 14 compares the ATE and RPE of estimates computed by our approach, Kintinuous and ORB-SLAM2 (w/o loop closure) with respect to the estimates computed by ORB-SLAM2. We compare against ORB-SLAM2 output since we don't have groundtruth trajectory estimates. ORB-SLAM2 (w/o loop closure) acts as a baseline and shows that using objects as landmarks during loop closure improves the trajectory estimate (better ATE, RPE as compared to ORB-SLAM2 w/o loop closure). The estimates returned by Kintinous for Handheld IRIM and CPL dataset are closer to Object-SLAM's estimate as well given that the ATE/RPE with respect to ORB-SLAM2 are similar. This shows that our approach performs similar to other state of art RGB-D

mapping approaches.

Table 15 compares the memory requirement of Object-SLAM as compared to ORB-SLAM2. We can see that Object-SLAM requires less memory than ORB-SLAM2 for both the sequences.

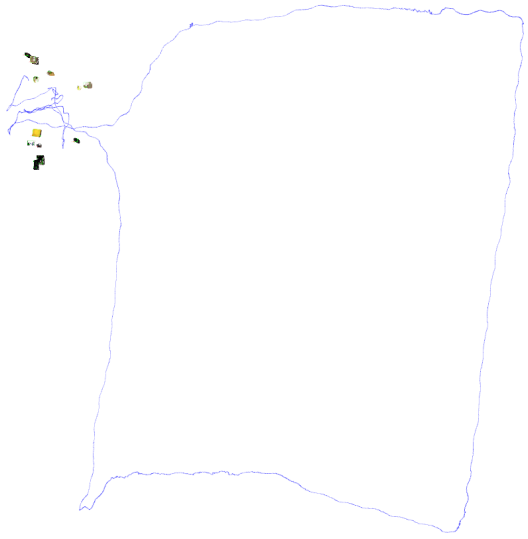


Object-SLAM with Joint Object Modeling and Mapping

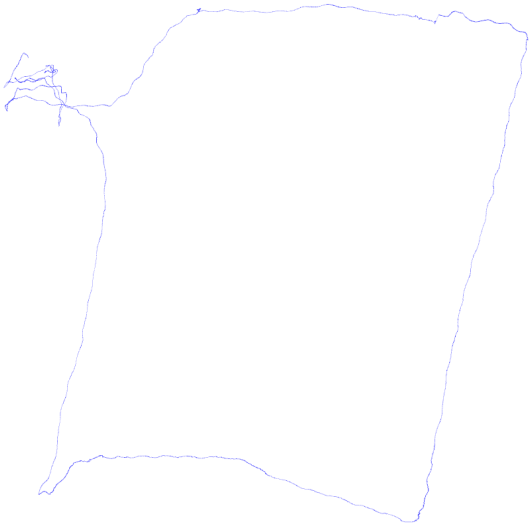


ORB-SLAM2

Figure 49: Qualitative comparison of the performance of Object-SLAM with joint object modeling and mapping (object-object loop closure, our approach) and ORB-SLAM2 (keyframe based loop-closure) method on handheld IRIM dataset. (Top) Shows the modeled objects and trajectory estimated by our approach. Zoom in along the trajectory in the electronic version to see the objects. (Bottom) Shows the aggregated point cloud and trajectory estimated by ORB-SLAM2 method. For object SLAM, object bounding boxes are shown in green and category labels shown in red.



Object-SLAM with Joint Object Modeling and Mapping



ORB-SLAM2

Figure 50: Qualitative comparison of the performance of Object-SLAM with joint object modeling and mapping (object-object loop closure, our approach) and ORB-SLAM2 (keyframe based loop-closure) method on handheld CPL dataset. (Top) Shows the modeled objects and trajectory estimated by our approach. Zoom in along the trajectory in the electronic version to see the objects. (Bottom) Shows the trajectory estimated by ORB-SLAM2 method. For object SLAM, object bounding boxes are shown in green and category labels shown in red.

5.4.4 Robot Experiments

In addition to the handheld datasets, we created new large scale datasets with a team of Jackal robots in different indoor scenarios.

Setup. In all the scenarios, the objects from the BigBird dataset [112] were randomly spread along the robot trajectory. We fuse the estimates from RGB-D sensor, laser scanner, IMU and wheel odometry for ego motion estimation as explained in Section 5.3. For object detection, we finetune the YOLO v2 detector on a subset of objects from BigBird dataset.

We compare the performance of Object-SLAM with joint object modeling and mapping with ORB-SLAM2. In both the case, we use the same algorithm for ego-motion estimation. However for loop closure, in the case of object-slam we use the modeled objects and estimate object-object loop closures, whereas in the case of ORB-SLAM2 we use the keyframes to estimate the loop closure.

The experiments were performed in the IRIM lab, CPL lab and in the 3rd floor of Klaus building at Georgia Tech. We also collected data inside a building at a military facility.

Table 16: ATE-O (in meters) and RPE-O comparison of Object-SLAM with joint object modeling and mapping (our approach) w.r.t ORB-SLAM2. We compare against ORB-SLAM2 output since we don't have groundtruth trajectory estimates.

	ATE-O	RPE-O
IRIM	0.000004	2.40e-06
CPL	0.000004	1.5e-06
Klaus	0.28	0.37
Military Facility	0.000451	0.0002

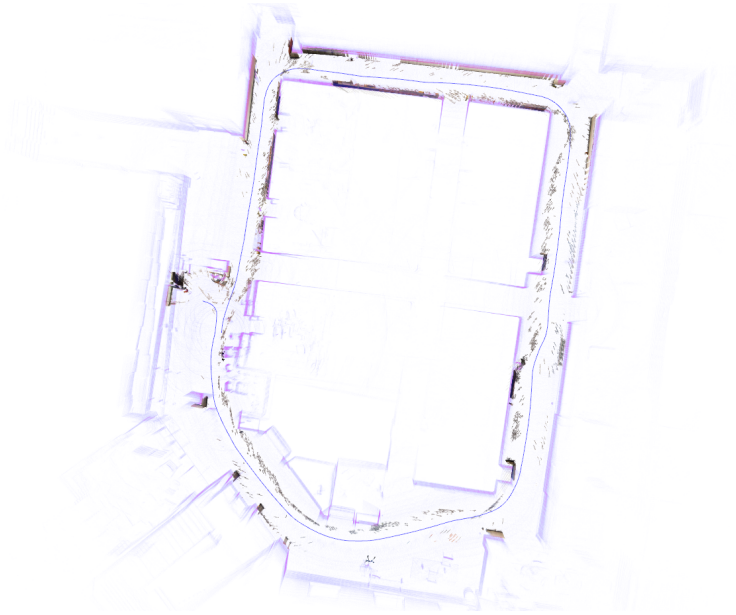
Results. Figures 51, 52, 53 and 54 show the qualitative comparison of the trajectories and point cloud estimated by Object-SLAM with joint object modeling and mapping as compared to ORB-SLAM2 for IRIM, CPL, Klaus and Military facility dataset respectively. Since we don't have groundtruth trajectories for these experiments, we use ORB-SLAM's

Table 17: Memory requirement comparison of Object-SLAM with joint object modeling and mapping (our approach) and ORB-SLAM2 for Robot datasets collected with Jackal robot in IRIM lab, CPL lab, Klaus building and a military training facility.

	Object-SLAM	ORB-SLAM2
IRIM	5.5 MB	524 MB
CPL	14 MB	1005 MB
Klaus	14 MB	739 MB
Military Facility	38 MB	184 MB

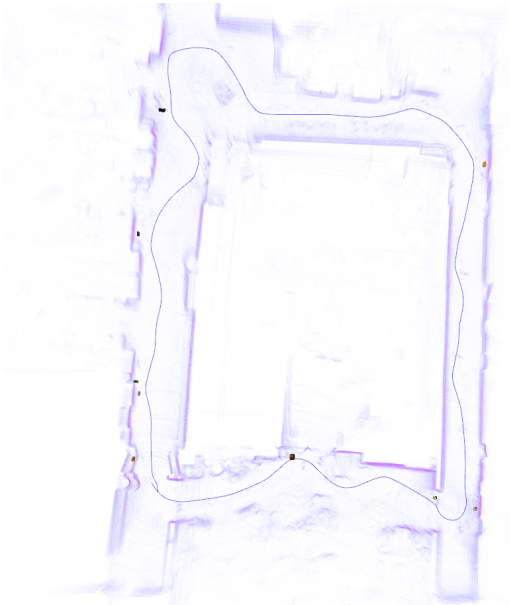
estimate as a groundtruth to measure the accuracy of the estimate given by our approach. We define ATE with respect to ORB-SLAM2 as *ATE-O* and RPE with respect to ORB-SLAM2 as *RPE-O*. Table 16 shows ATE-O and RPE-O on the four datasets. We can see that the estimate returned by our approach is close to the estimate returned by the state of art RGB-D mapping approach (ORB-SLAM2). Table 17 compares the memory requirement of Object-SLAM as compared to ORB-SLAM2. We can see that Object-SLAM requires less memory than ORB-SLAM2 for all the sequences. We see that the memory savings for large scale exploratory scenes is even higher as compared to standard benchmark datasets like TUM and UW RGB-D Scenes v2. This shows that our approach scales better with the size of the map as compared to state of art RGB-D mapping approaches.

Object-SLAM with Joint Object Modeling and Mapping



ORB-SLAM2

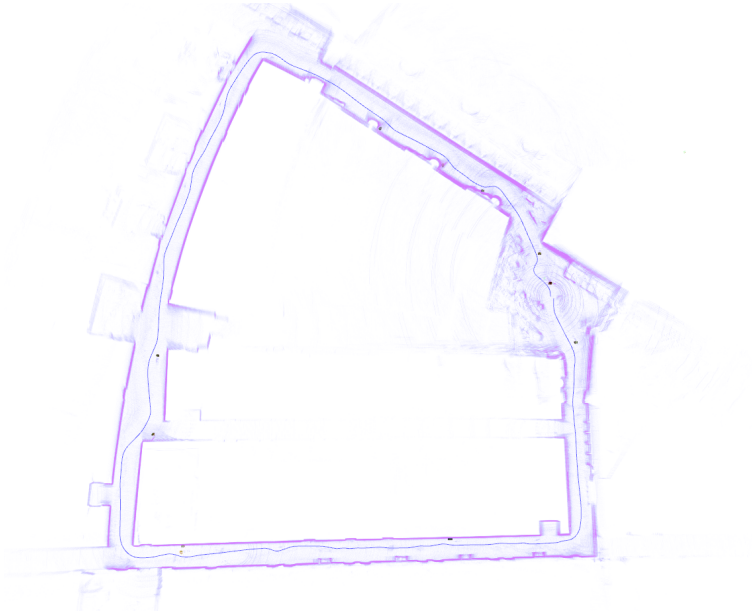
Figure 51: Qualitative comparison of the performance of Object-SLAM with joint object modeling and mapping (object-object loop closure, our approach) and ORB-SLAM2 (keyframe based loop-closure) method on IRIM dataset. (Top) Shows the aggregated point cloud and trajectory estimated by our approach. Zoom in along the trajectory in the electronic version to see the objects. (Bottom) Shows the aggregated point cloud and trajectory estimated by ORB-SLAM2 method. For object SLAM, object bounding boxes are shown in green and category labels shown in red. The transparent point cloud background is just shown for visualization.



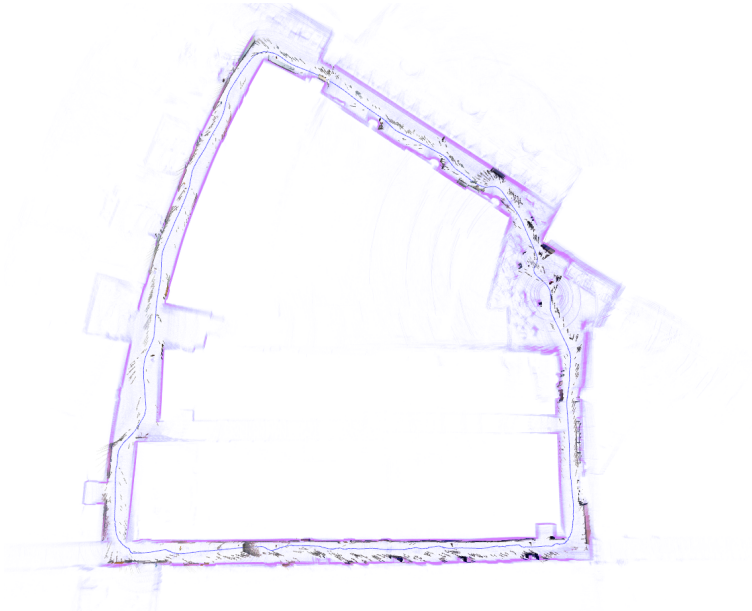
Object-SLAM with Joint Object Modeling and Mapping

ORB-SLAM2

Figure 52: Qualitative comparison of the performance of Object-SLAM with joint object modeling and mapping (object-object loop closure, our approach) and ORB-SLAM2 (keyframe based loop-closure) method on CPL dataset. (Top) Shows the aggregated point cloud and trajectory estimated by our approach. Zoom in along the trajectory in the electronic version to see the objects. (Bottom) Shows the aggregated point cloud and trajectory estimated by ORB-SLAM2 method. For object SLAM, object bounding boxes are shown in green and category labels shown in red. The transparent point cloud background is just shown for visualization.

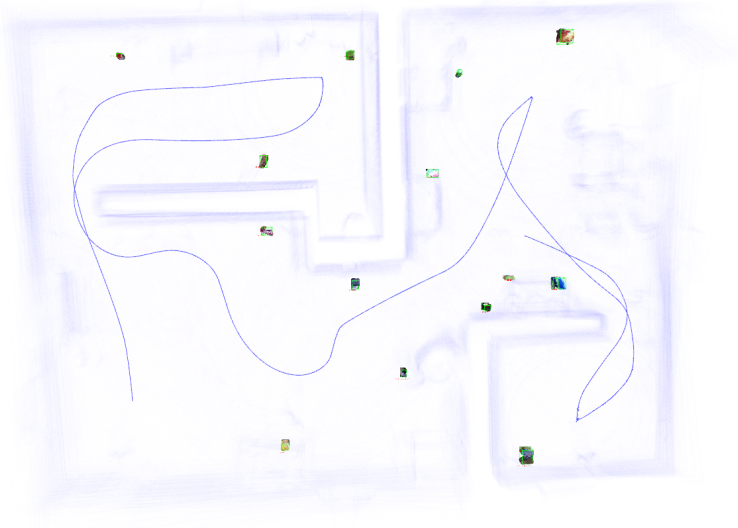


Object-SLAM with Joint Object Modeling and Mapping



ORB-SLAM2

Figure 53: Qualitative comparison of the performance of Object-SLAM with joint object modeling and mapping (object-object loop closure, our approach) and ORB-SLAM2 (keyframe based loop-closure) method on Klaus dataset. (Top) Shows the aggregated point cloud and trajectory estimated by our approach. Zoom in along the trajectory in the electronic version to see the objects. (Bottom) Shows the aggregated point cloud and trajectory estimated by ORB-SLAM2 method. For object SLAM, object bounding boxes are shown in green and category labels shown in red. The transparent point cloud background is just shown for visualization.



Object-SLAM with Joint Object Modeling and Mapping

ORB-SLAM2

Figure 54: Qualitative comparison of the performance of Object-SLAM with joint object modeling and mapping (object-object loop closure, our approach) and ORB-SLAM2 (keyframe based loop-closure) method on a dataset collected at a military facility. (Top) Shows the aggregated point cloud and trajectory estimated by our approach. (Bottom) Shows the aggregated point cloud and trajectory estimated by ORB-SLAM2 method. For object SLAM, object bounding boxes are shown in green and category labels shown in red. The transparent point cloud background is just shown for visualization.

5.5 Conclusions

In this chapter, we proposed an approach to SLAM with simultaneous object modeling and mapping. Objects provide a richer description of the environment and can be more effective for data association. We use the modeled objects as landmarks and the data association among them assist with the loop closure in large environments. The pipeline follows an online learning framework to avoid any pre-training on the object models.

We show that our approach requires less memory as compared to the state of art RGB-D mapping approach like ORB-SLAM2 (Table 10, 13 and 17) whereas the accuracy of the trajectory estimated by our approach is similar to other RGB-D mapping approaches (Table 8, 11, 16). We also show that our approach scales better with the size of the map as compared to other RGB-D mapping approaches (Table 17, 15).

The take-home message from this chapter is that dense RGB-D mapping approaches like ElasticFusion or Kintinuous or sparse feature based methods like ORB-SLAM2 can be used for local and accurate visual tracking but we don't have to store all the dense map information (for ElasticFusion or Kintinuous) or sparse keyframes (for ORB-SLAM2) for large-scale trajectory estimation. Storing just the salient information like objects (or planes) is good enough for closing the loop resulting in the same level of accuracy as other state of art RGB-D mapping approaches.

CHAPTER VI

DISTRIBUTED OBJECT BASED SLAM WITH JOINT OBJECT MODELING AND MAPPING

6.1 Introduction

The distributed object SLAM approach introduced in Chapter 4 assumes that the object model of each instance that each mapped in an environment is known in advance. However it can be challenging to store a model of all the object instances due to large intra-class variation. Searching through all the object models for object pose estimation can be computationally demanding. It will not generalize to new unseen instances of the same object category as well.

In the previous chapter, we introduce a framework for object based SLAM with joint object modeling and mapping and show that the approach is nearly as accurate as the state of art RGB-D mapping approach while using less memory than those approaches.

Contribution. In this chapter, we propose to integrate the joint object modeling and mapping framework (Chapter 5) with the distributed object SLAM approach (Chapter 4) resulting in distributed object based SLAM with Joint Object Modeling and Mapping. In other words, we extend the distributed object SLAM approach introduced in Chapter 4 to the case where object models are previously unknown and are modeled jointly with Distributed Object based SLAM. We use an off-the-shelf lightweight convolutional network based object detectors to detect object at categorical level which are then modeled at instance level by integrating detection across frames. The modeled object instance can then be data associated against other instances seen by the same robot or other robots to generate object-object constraint. We show that this approach generalizes multi robot object-based SLAM to unseen object models while further reducing the memory required to store the

object models.

Section 6.2 introduces the additional mathematical notation and formalizes the problem of distributed object SLAM with joint object modeling and mapping. Section 6.3 presents the implementation details of our system.

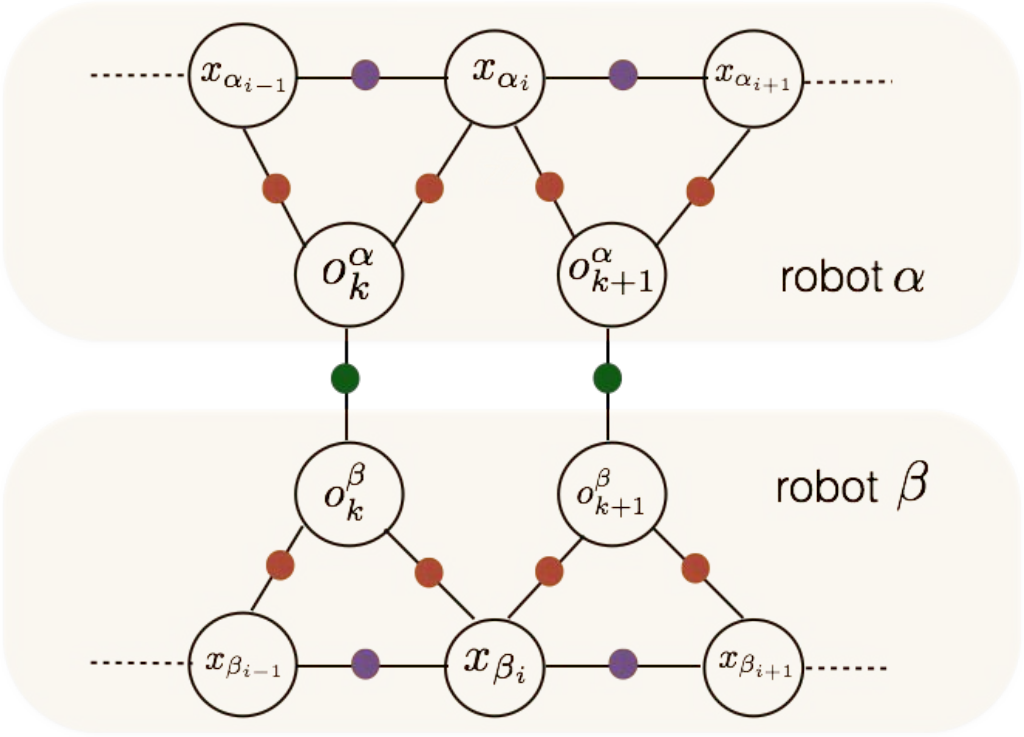


Figure 55: Factor graph representation of Distributed Object based SLAM with Joint Modeling and Mapping. $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{x}_{\beta_i}$ denote the poses assumed by robot α and β at time i respectively. The pose of the k^{th} object as estimated by robot α and β is denoted with $\boldsymbol{o}_k^\alpha$ and $\boldsymbol{o}_k^\beta$ respectively. Green dots shows inter-robot factors whereas orange and purple dots shows intra-robot factors.

6.2 Problem Formulation: Distributed Object-based SLAM with Joint Object Modeling And Mapping

We consider a distributed object based SLAM system as defined in Section 4.2. Each robot, in addition to estimating its own trajectory using local measurements and occasional communication with other robots, also models a set of objects in the environment. We model each trajectory as a finite set of poses; the trajectory of robot α is $\boldsymbol{x}_\alpha = [\boldsymbol{x}_{\alpha_1}, \boldsymbol{x}_{\alpha_2}, \ldots]$. We denote with $\boldsymbol{o}_k^\alpha \in \mathrm{SE}(3)$ the pose of the k^{th} object as modeled by of robot α (Fig. 55).

Measurements. Similar to distributed object based SLAM (Section 4.2), we assume that each robot acquires two types of relative pose measurements: intra-robot and inter-robot measurements. The *intra-robot measurements* consist of the odometry measurements, which constrain consecutive robot poses (e.g., $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{x}_{\alpha_{i+1}}$ in Fig. 55), and object measurements which constrains robot poses with the corresponding visible object landmarks (e.g., $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{o}_k^\alpha$ in Fig. 55). The *inter-robot measurements* relate the object poses observed by different robots. During a rendezvous between robot α and robot β, each robot shares the label and object TDF representation of detected object landmarks with the other robot. If there exists a matching object landmark, the teammates add inter-robot measurements, enforcing the object pose estimate to be consistent across the teammates. For instance, if $\boldsymbol{o}_k^\beta$ and $\boldsymbol{o}_k^\alpha$ in Fig. 55 model the the same object, then the two models should align in the global coordinate frame. We use sparse matching followed by ICP to find the relative transformation between the modeled objects which is used as the inter-robot measurement.

The intra-robot and inter-robot object measurements follow the same measurements model of eq. (1). For instance, if the robot α at time i and at pose $\boldsymbol{x}_{\alpha_i}$ observes an object at pose $\boldsymbol{o}_k^\alpha$, then the corresponding measurement $\bar{\boldsymbol{z}}_{\boldsymbol{o}_k^\alpha}^{\boldsymbol{x}_{\alpha_i}}$ measures the relative pose between $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{o}_k^\alpha$. Similarly we denote inter-robot measurement between object poses $\boldsymbol{o}_k^\alpha$ and $\boldsymbol{o}_k^\beta$ as $\bar{\boldsymbol{z}}_{\boldsymbol{o}_k^\beta}^{\boldsymbol{o}_k^\alpha}$.

In the following, we denote with $\mathcal{E}_I^\alpha$ the set of intra-robot odometry for robot α, while we call $\mathcal{E}_I$ the set of intra-robot odometry measurements for all robots in the team, i.e., $\mathcal{E}_I = \cup_{\alpha\in\boldsymbol{\Omega}}\mathcal{E}_I^\alpha$. Similarly the set of intra-robot object measurements for robot α is denoted as $\mathcal{E}_o^\alpha$, whereas the set of all intra-robot object measurements is denoted as $\mathcal{E}_o$. Similar to Section 3.2, the set of inter-robot measurements involving robot α is denoted with $\mathcal{E}_S^\alpha$. The set of all inter-robot measurements is denoted with $\mathcal{E}_S$. The set of all available measurements is then $\mathcal{E} = \mathcal{E}_I \cup \mathcal{E}_O \cup \mathcal{E}_S$. Note that each robot only has access to its own intra and inter-robot measurements $\mathcal{E}_I^\alpha$, $\mathcal{E}_O^\alpha$ and $\mathcal{E}_S^\alpha$.

ML trajectory and objects estimation. Let us collect all robot trajectories and object poses in a (to-be-estimated) set of robot poses $\boldsymbol{x} = [\boldsymbol{x}_\alpha, \boldsymbol{x}_\beta, \boldsymbol{x}_\gamma, \ldots]$ and set of object poses $\boldsymbol{o} = [\boldsymbol{o}^\alpha, \boldsymbol{o}^\beta, \boldsymbol{o}^\gamma, \ldots]$. The ML estimate for $\boldsymbol{x}$ and $\boldsymbol{o}$ is defined as the maximum of the measurement likelihood:

$$\boldsymbol{x}^\star, \boldsymbol{o}^\star = \arg\max_{\boldsymbol{x},\boldsymbol{o}} \prod_{(\boldsymbol{x}_{\alpha_i}, \boldsymbol{x}_{\alpha_{i+1}}) \in \mathcal{E}_I} \underbrace{\mathcal{L}(\bar{\boldsymbol{z}}^{\boldsymbol{x}_{\alpha_i}}_{\boldsymbol{x}_{\alpha_{i+1}}} \mid \boldsymbol{x})}_{\text{odometry factors}} \prod_{(\boldsymbol{x}_{\alpha_i}, \boldsymbol{o}^\alpha_k) \in \mathcal{E}_O} \underbrace{\mathcal{L}(\bar{\boldsymbol{z}}^{\boldsymbol{x}_{\alpha_i}}_{\boldsymbol{o}^\alpha_k} \mid \boldsymbol{x}, \boldsymbol{o})}_{\text{intra-robot object-measurement factors}}$$
$$\prod_{(\boldsymbol{o}^\alpha_i, \boldsymbol{o}^\beta_j) \in \mathcal{E}_S} \underbrace{\mathcal{L}(\bar{\boldsymbol{z}}^{\boldsymbol{o}^\alpha_i}_{\boldsymbol{o}^\beta_j} \mid \boldsymbol{x}, \boldsymbol{o})}_{\text{inter-robot object-object factors}} \tag{31}$$

where we used the same assumptions on measurement noise as in Section 3.2. Defining $\mathcal{X} = \boldsymbol{x} \cup \boldsymbol{o}$, we rewrite eq. (28) as:

$$\mathcal{X}^\star = \arg\max_{\mathcal{X}} \prod_{(\alpha_i, \beta_j) \in \mathcal{E}} \mathcal{L}(\bar{\boldsymbol{z}}^{\alpha_i}_{\beta_j} \mid \mathcal{X}) \tag{32}$$

Since the optimization problem in eq. (29) has the same structure of the one in eq. (2), we follow the same steps to solve it in a distributed manner using the Distributed Gauss-Seidel method.

The next section presents the implementation details of our distributed object-based SLAM system.

6.3 Implementation Details: Distributed Object based SLAM with Joint Object Modeling and Mapping

Ego-Motion Estimation. Each robot collects 3D scans using Velodyne 32E, RGB-D scans using Orbbec Astra sensor, and inertial measurements using IMU and odometry measurements using wheel sensors. Fig. 22 shows the sensor layout on a Jackal robot. Relative pose estimates from all the sensors are fused together using OmniMapper[128] to estimate robot's ego-motion[1]. Fig. 3 shows the overview of ego-motion estimation.

[1] `https://github.com/CognitiveRobotics/omnimapper`

Specifically, 3D scans from Veldoyne 32E are used to compute relative pose with respect to the previous scan using GICP (generalized iterative closest point [110]). Inertial measurements from IMU are fused with Wheel Odometry measurements to generate IMU corrected odometry estimates. Relative pose estimates using RGB-D scans are computed using ORB-SLAM [85]. Relative pose estimates are then fused using OmniMapper.

Object Modeling and Mapping. The implementation details for object modeling and mapping is explained in detail in the previous chapter (Ch. 5). In summary, each frame is segmented into planar and non-planar segments. At the same time, each frame is passed through YOLO object detector which generates bounding box detections. All the non planar segments which are inside a detection bounding box are considered as object segment and are associated with other co-visible object landmarks. Object models which are sufficiently confident about their category label and are visible in sufficient number of frames are then added as a variable in the SLAM factor graph. Corresponding object-pose factors are added to the factor graph as well. Object-object loop closure runs in a parallel thread, where it tries to match the object landmarks which are added to the factor graph. If successful, the loop closure thread adds object-object loop closure factor to the factor graph. Figure 45 shows the complete flowchart.

Robot Communication. During a rendezvous between robots α and β, robot α communicates the category labels (class) and object TDF representation of all the added objects landmarks to robot β. The communicated object is matched to all the other landmarks having the same object category label. Correspondences are estimated using 3DMatch [139]. The estimated relative pose given the sparse correspondences is then further refined using ICP. If the average euclidean distance of the transformed points in one object is less than 10^{-3} to the points in the matching object, we consider the matching to be successful. The resulting transformation is then used as the object-object intra-robot measurement and the corresponding factor is added to the factor graph.

The list of shared objects contains pairs $(\boldsymbol{o}_k^\alpha, \boldsymbol{o}_l^\beta)$ along with relative transformation

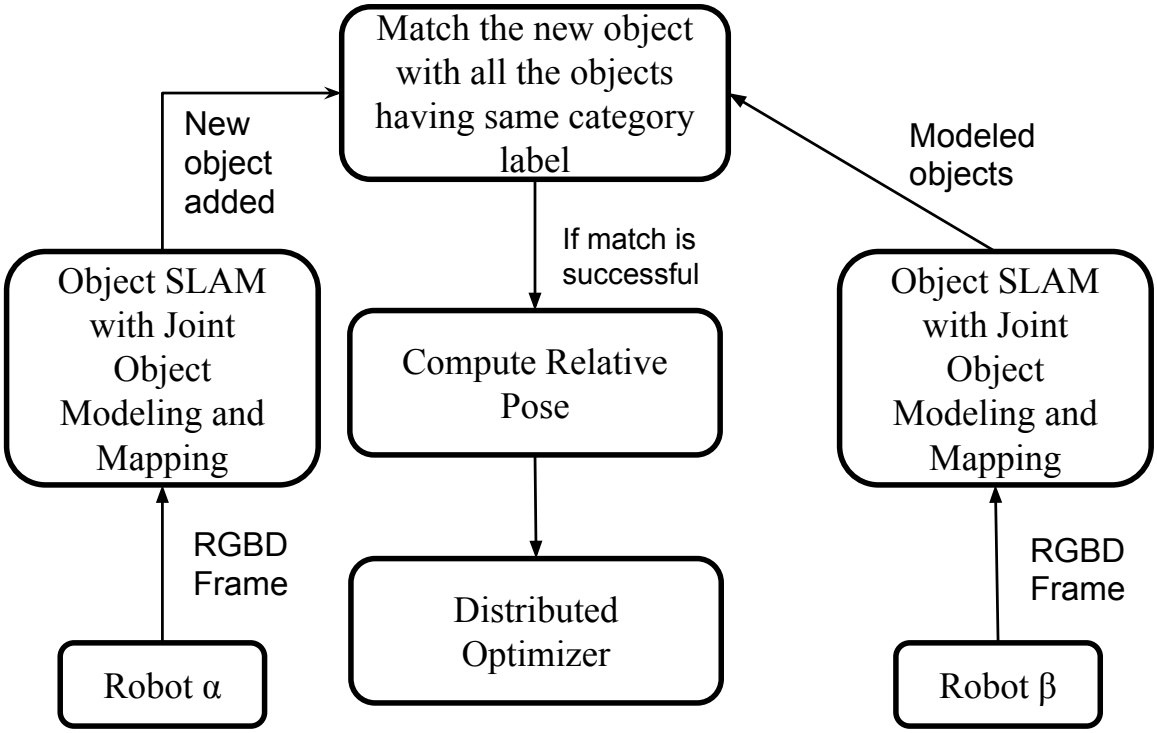


Figure 56: Overview of Distributed Object SLAM Communication

required to align the object models in the global coordinate frame. An overview of the robot communication pipeline is shown in Fig. 56.

Next we show the experimental evaluation which includes field experiments in different indoor environments.

6.4 Experiments

6.4.1 Experimental Setup

In order to evaluate the performance of our algorithm, we created new large scale datasets with a team of Jackal robots in different indoor scenarios. In all the scenarios, the objects from the BigBird dataset [112] were randomly spread along the robot trajectory. We fuse the estimates from RGB-D sensor, laser scanner, IMU and wheel odometry for ego motion estimation as explained in Section 6.3. For object detection, we finetune the YOLO v2 detector on a subset of objects from BigBird dataset.

We compare the performance of distributed object-SLAM with joint object modeling and mapping with distributed ORB-SLAM2 (as explained in Chapter 3). For both the algorithms, we use the same algorithm for ego-motion estimation. However for loop closure and communication, in the case of distributed object-slam we use the modeled objects and estimate object-object loop closures, whereas in the case of distributed ORB-SLAM2 we use the keyframes to estimate the loop closure.

The experiments were performed in the IRIM lab and in the 3rd floor of Klaus building at Georgia Tech. We also collected data inside a building at a military facility.

Table 18: Per-robot memory and communication requirement comparison of Distributed Object-SLAM with joint object modeling and mapping (our approach) and Distributed ORB-SLAM2 for dataset collected in IRIM lab given in Figure 57.

Robot ID	Memory Requirements		Communication Requirements	
	Dist. Object-SLAM	Dist. ORB-SLAM2	Dist. Object-SLAM	Dist. ORB-SLAM2
1	5.6 MB	528 MB	61.6 MB	5808 MB
2	12 MB	333 MB	132 MB	3663 MB
3	12 MB	732 MB	132 MB	8052 MB
4	8.0 MB	237 MB	88 MB	2607 MB
5	8.6 MB	294 MB	94.6 MB	3234 MB
6	17 MB	954 MB	187 MB	10494 MB
7	11 MB	463 MB	121 MB	5093 MB
8	14 MB	631 MB	154 MB	6941 MB
9	18 MB	984 MB	198 MB	10824 MB
10	19 MB	834 MB	209 MB	9174 MB
11	3.2 MB	555 MB	35.2 MB	6105 MB

Table 19: Per-robot memory and communication requirement comparison of Distributed Object-SLAM with joint object modeling and mapping (our approach) and Distributed ORB-SLAM2 for dataset collected in Klaus building given in Figure 62.

Robot ID	Memory Requirements		Communication Requirements	
	Dist. Object-SLAM	Dist. ORB-SLAM2	Dist. Object-SLAM	Dist. ORB-SLAM2
1	12 MB	762 MB	60 MB	3810 MB
2	3.5 MB	133 MB	17.5 MB	665 MB
3	8.0 MB	611 MB	40 MB	3055 MB
4	8.4 MB	941 MB	42 MB	4705 MB
5	6.2 MB	225 MB	31 MB	1125 MB

Table 20: Per-robot memory and communication requirement comparison of Distributed Object-SLAM with joint object modeling and mapping (our approach) and Distributed ORB-SLAM2 for dataset collected in a military training facility given in Figure 67 .

Robot ID	Memory Requirements		Communication Requirements	
	Dist. Object-SLAM	Dist. ORB-SLAM2	Dist. Object-SLAM	Dist. ORB-SLAM2
1	38 MB	193 MB	380 MB	1930 MB
2	34 MB	294 MB	340 MB	2940 MB
3	23 MB	109 MB	230 MB	1090 MB
4	18 MB	124 MB	180 MB	1240 MB
5	15 MB	62 MB	150 MB	620 MB
6	15 MB	114 MB	150 MB	1140 MB
7	16 MB	87 MB	160 MB	870 MB
8	15 MB	129 MB	150 MB	1290 MB
9	4.0K	86 MB	40 MB	860 MB
10	41 MB	171 MB	410 MB	1710 MB

6.4.2 Results

Fig. 57, 62 and 67 show the qualitative comparison of the trajectories and point cloud estimated by distributed Object SLAM with joint object modeling and mapping as compared to distributed ORB-SLAM2 for IRIM, Klaus and Military facility dataset respectively.

6.4.2.1 Accuracy

Since we don't have groundtruth trajectories for these experiments, we use ORB-SLAM's estimate as a groundtruth to measure the accuracy of the estimate given by our approach. We define ATE w.r.t ORB-SLAM2 as *ATE-O* and RPE w.r.t ORB-SLAM2 as *RPE-O*. Figure 58 and 59 shows ATE-O and RPE-O for the IRIM dataset. We also computed ATE* and RPE* with respect to the centralized estimate. Figure 60 and 61 shows ATE* and RPE* for IRIM dataset. Similarly, Fig. 63, 64, 65 and 66 shows ATE-O, RPE-O, ATE* and RPE* for Klaus dataset respectively. Figure 68 and 69 shows ATE-O and RPE-O for the dataset collected in a military facility. Given the plots, we can see that the estimate returned by our approach is close to the estimate returned by the distributed RGB-D mapping approach (ORB-SLAM2).

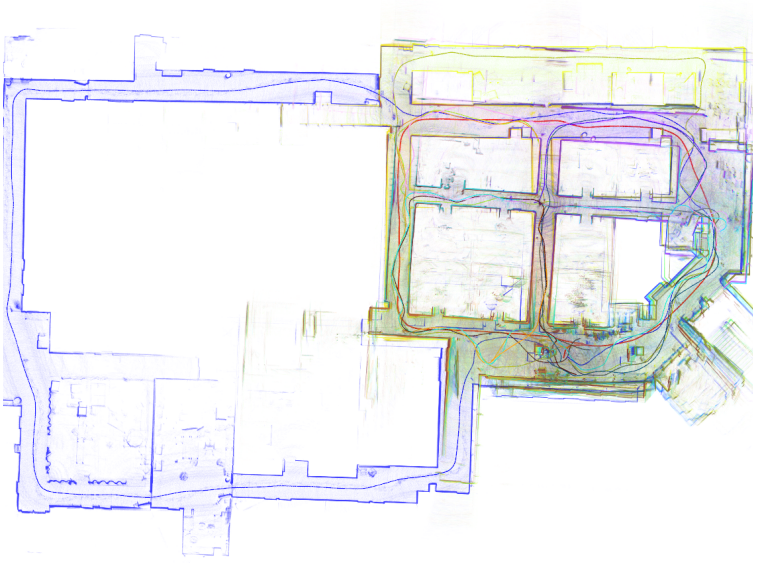
6.4.2.2 Memory

Tables 18, 19, 20 compares per-robot memory requirement of Distributed Object-SLAM as compared to Distributed ORB-SLAM2. The per-robot memory requirement in the case of ORB-SLAM2 is the amount of memory required to store the keyframes. In the case of object level map, it is the amount of memory required to store the object models. We can see that Distributed Object-SLAM with joint object modeling and mapping requires less memory than Distributed ORB-SLAM2 for all the sequences. This shows that our approach reduces the memory requirements while maintaining the final accuracy as distributed ORB-SLAM2. The amount of memory required to store all the models in BigBird data is 203 MB. Tables 18, 19, 20 confirms that the memory requirement of joint object modeling and

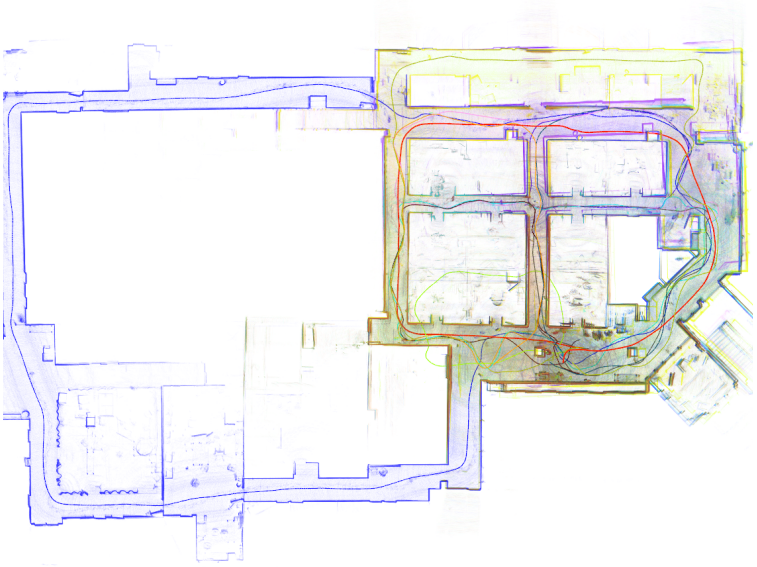
mapping is less than object-slam with known object models.

6.4.2.3 Communication Requirement

Tables 18, 19, 20 show that, as expected, the per-robot memory requirement is orders of magnitude smaller with our object-based map as compared to key-frame based maps (ORB-SLAM2). When using ORB-SLAM2, the robots are required to send keyframes at every rendezvous to estimate their relative pose. Assuming in the worst case scenario, each robot has to send all the keyframes to all the robots, then the average communication requirement for each robot in the case of ORB-SLAM2 is computed as nM_c where n is the number of robots and M_c is the memory required to store the keyframes. Communication in the case of our object-based map requires sending object models; a upper bound in the worst case scenario can be computed as nM_o where n is the number of robots and M_o is the memory required to store object models. Tables 18, 19, 20 confirms that our approach provides a remarkable advantage in terms of communication burden as it requires transmitting orders of magnitude less than keyframe-based approach. However, the communication requirement for distributed object based SLAM with joint object modeling and mapping is more than the memory requirement for distributed object based SLAM with known object models since in the latter case, we only have to communicate object category labels and pose where as in the case of joint object modeling and mapping, modeled objects have to be communicated.



Distributed Object-SLAM with Joint Object Modeling and Mapping



Distributed ORB-SLAM2

Figure 57: Test with 11 robots in the IRIM lab. Shows the qualitative comparison of the performance of distributed object-SLAM with joint object modeling and mapping (object-object loop closure, our approach) and distributed ORB-SLAM2 (keyframe based loop-closure) method. (Top) Shows the aggregated point cloud and trajectories estimated by our approach. Zoom in along the trajectory in the electronic version to see the objects. (Bottom) Shows the aggregated point cloud and trajectories estimated by distributed ORB-SLAM2 method (Chpater 3). For object SLAM, object bounding boxes are shown in green and category labels shown in red. Transparent point cloud background is just shown for the visualization of final estimates.

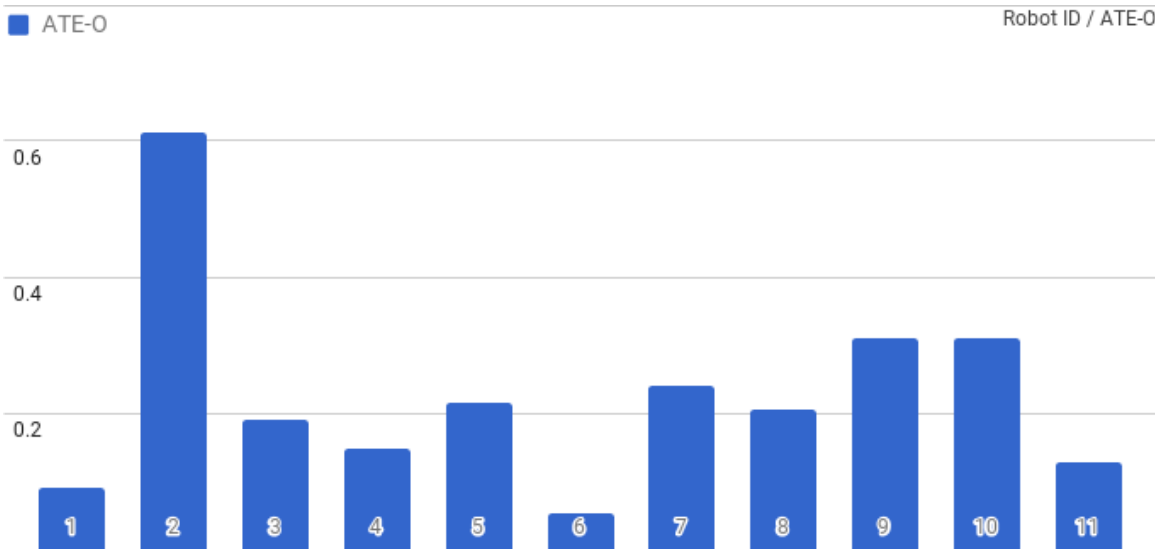


Figure 58: Per-Robot ATE-O comparison (in meters) with respect to ORB-SLAM2 estimate for data collected in IRIM lab given in Fig. 57.

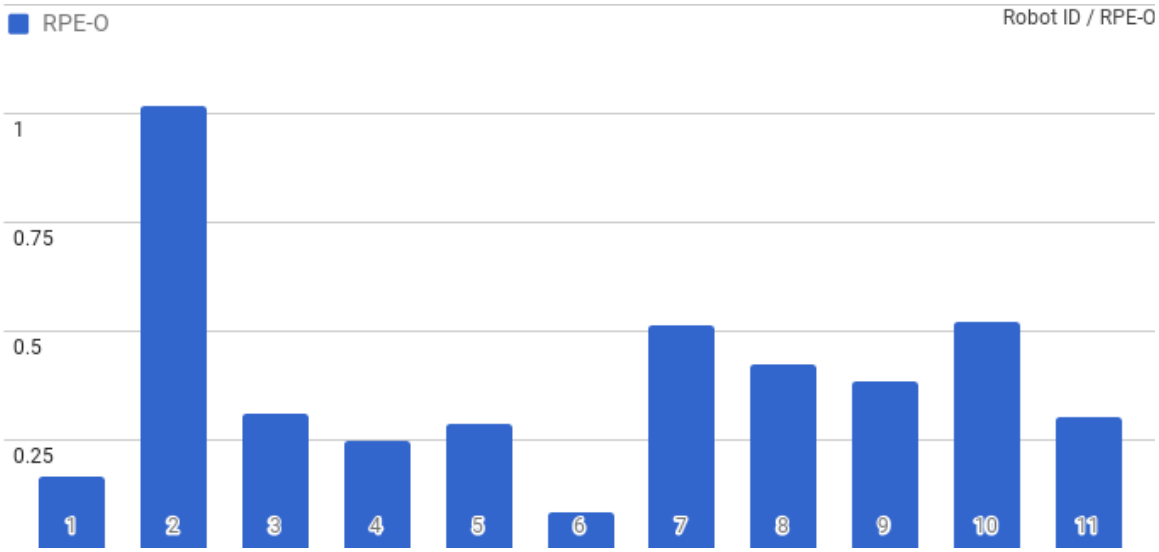


Figure 59: Per-Robot RPE-O comparison with respect to ORB-SLAM2 estimate for data collected in IRIM lab given in Fig. 57.

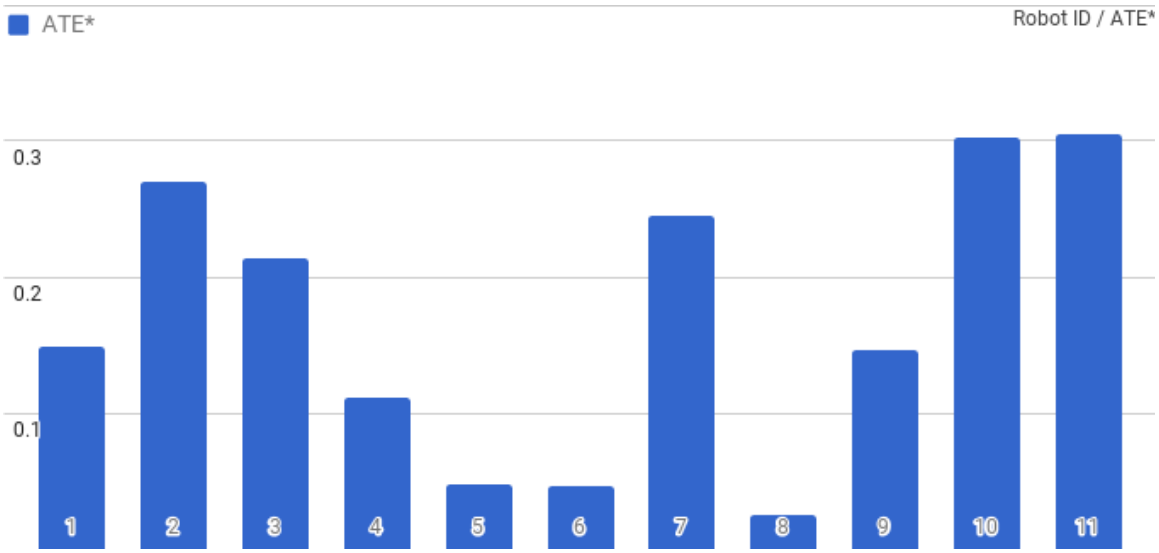


Figure 60: Per-Robot ATE* comparison (in meters) with respect to Centralized estimate for data collected in IRIM lab given in Fig. 57.

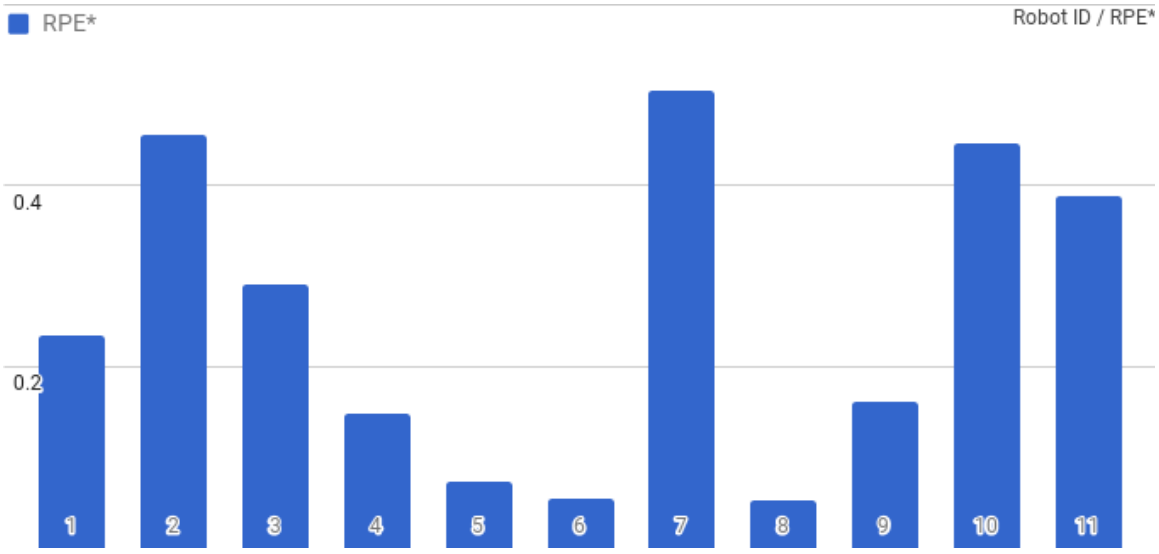
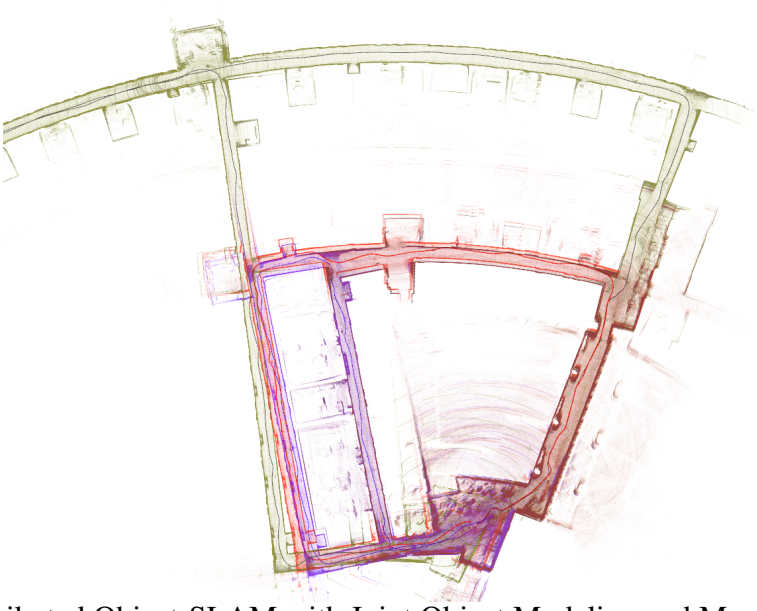
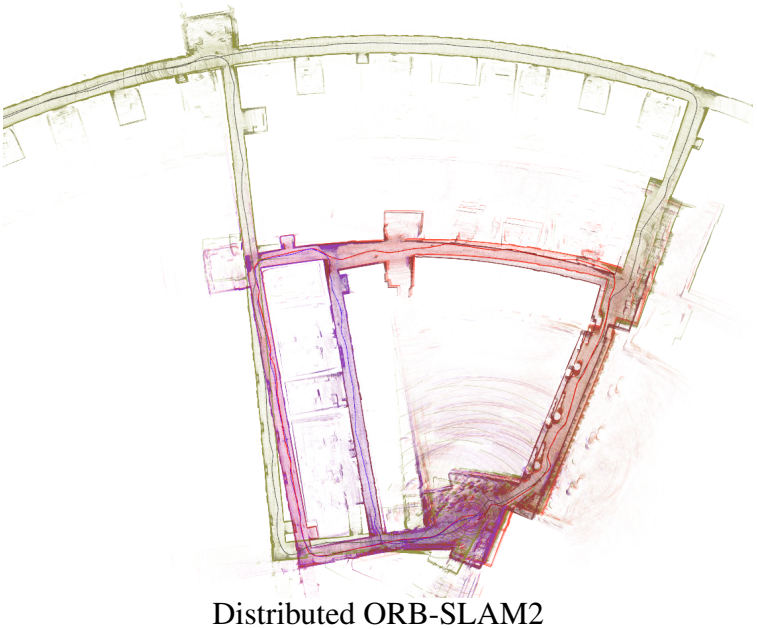


Figure 61: Per-Robot RPE* comparison with respect to Centralized estimate for data collected in IRIM lab given in Fig. 57.



Distributed Object-SLAM with Joint Object Modeling and Mapping



Distributed ORB-SLAM2

Figure 62: Test with 5 robots in Klaus building. Shows the qualitative comparison of the performance of distributed object-SLAM with joint object modeling and mapping (object-object loop closure, our approach) and distributed ORB-SLAM2 (keyframe based loop-closure) method. (Top) Shows the aggregated point cloud and trajectories estimated by our approach. Zoom in along the trajectory in the electronic version to see the objects. (Bottom) Shows the aggregated point cloud and trajectories estimated by ORB-SLAM2 method (Chpater 3). For object SLAM, object bounding boxes are shown in green and category labels shown in red. Transparent point cloud background is just shown for visualization.

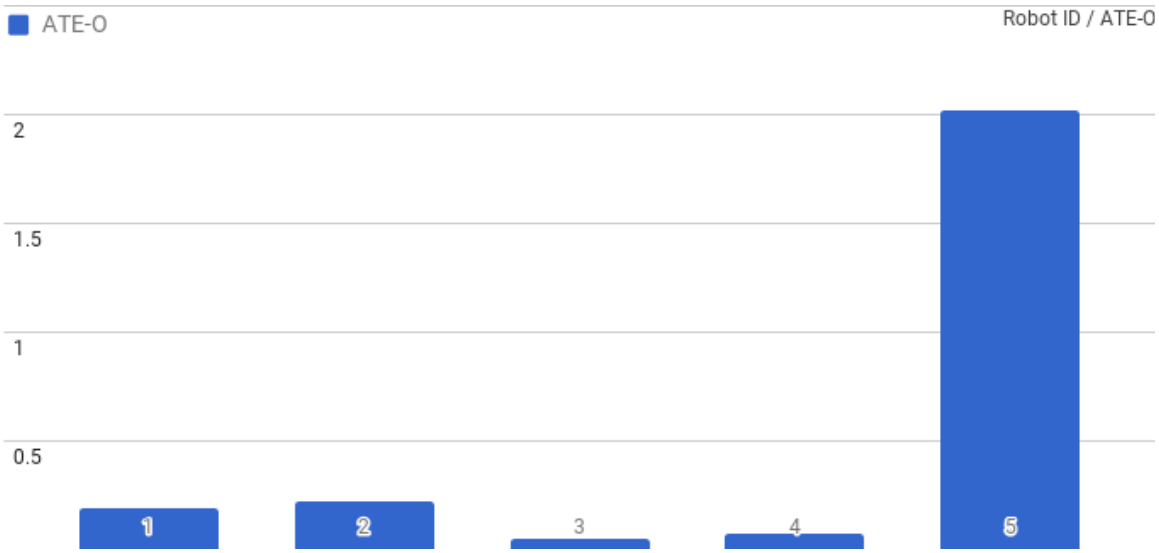


Figure 63: Per-Robot ATE-O comparison (in meters) w.r.t ORB-SLAM2 estimate for data collected in Klaus building given in Fig. 62.

Figure 64: Per-Robot RPE-O comparison w.r.t ORB-SLAM2 estimate for data collected in Klaus building given in Fig. 62.

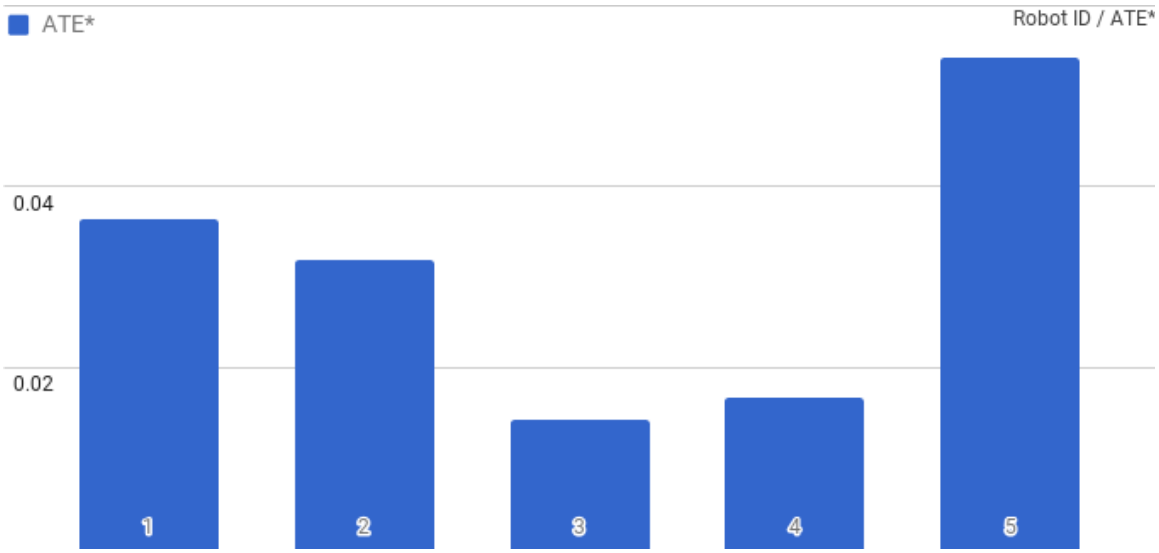


Figure 65: Per-Robot ATE* comparison (in meters) with respect to Centralized estimate for data collected in Klaus building given in Fig. 62.

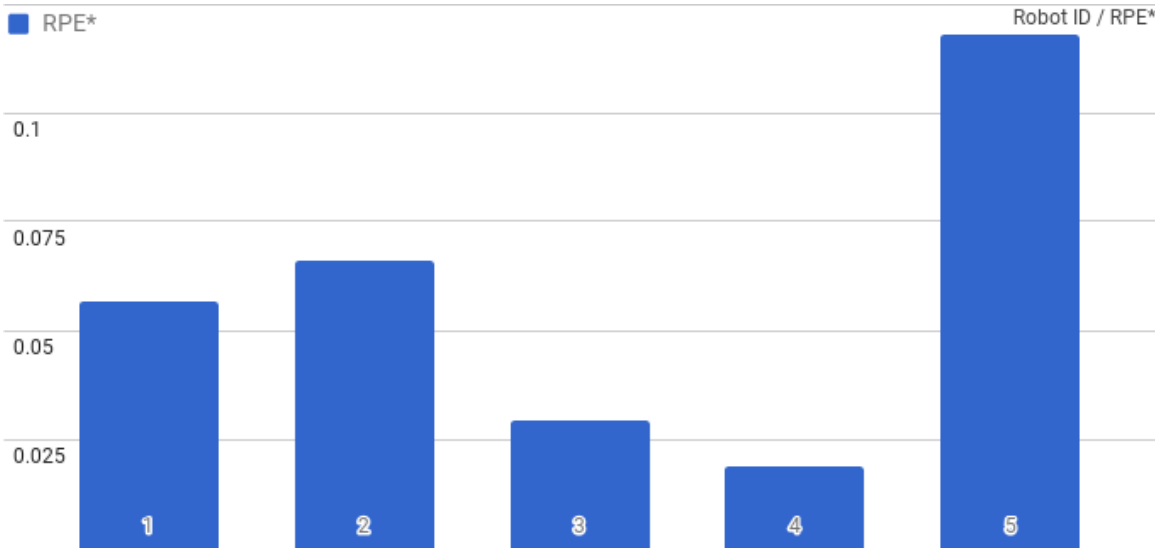
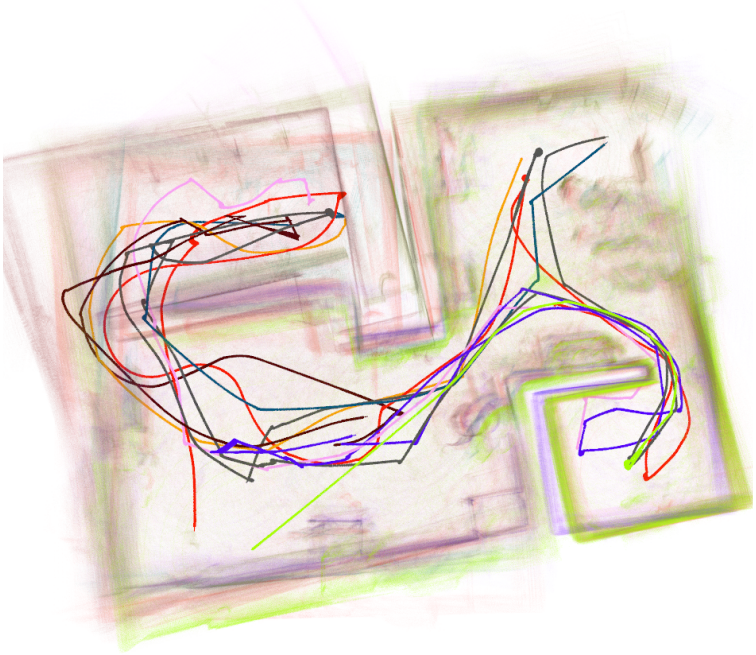
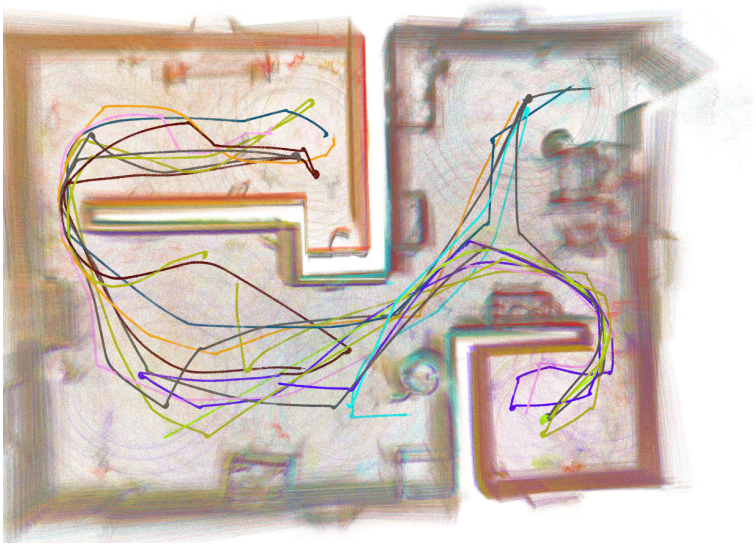


Figure 66: Per-Robot RPE* comparison with respect to Centralized estimate for data collected in Klaus building given in Fig. 62.



Distributed Object-SLAM with Joint Object Modeling and Mapping



Distributed ORB-SLAM2

Figure 67: Test with 10 robots in a military training facility. Shows the qualitative comparison of the performance of distributed object-SLAM with joint object modeling and mapping (object-object loop closure, our approach) and distributed ORB-SLAM2 (keyframe based loop-closure) method. (Top) Shows the aggregated point cloud and trajectories estimated by our approach. (Bottom) Shows the aggregated point cloud and trajectories estimated by ORB-SLAM2 method (Chpater 3). For object SLAM, object bounding boxes are shown in green and category labels shown in red. Transparent point cloud background is just shown for visualization.

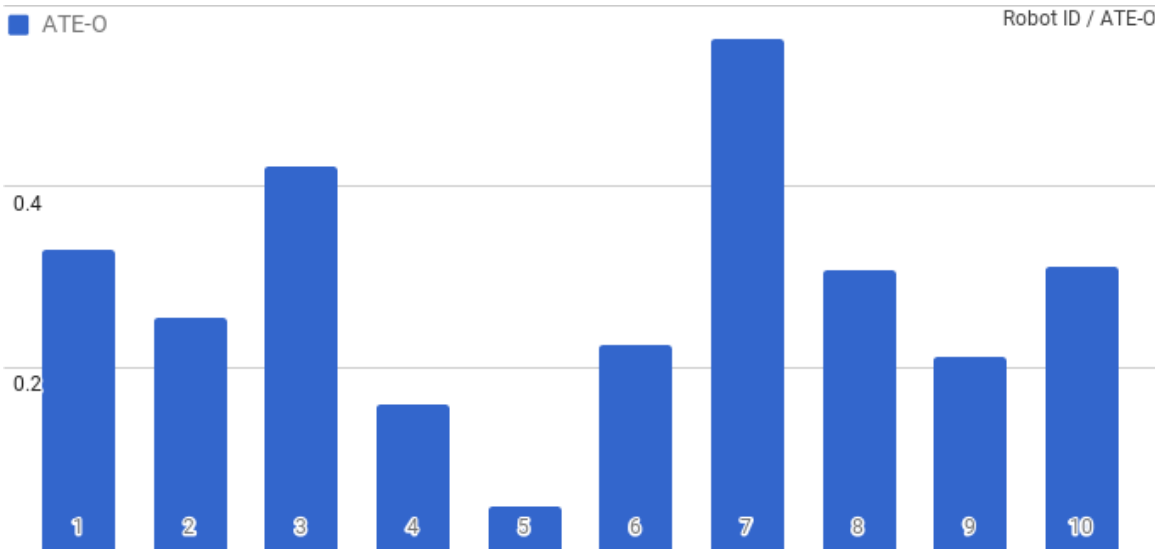


Figure 68: Per-Robot ATE-O comparison (in meters) with respect to ORB-SLAM2 estimate for data collected in a military training facility given in Fig. 67.

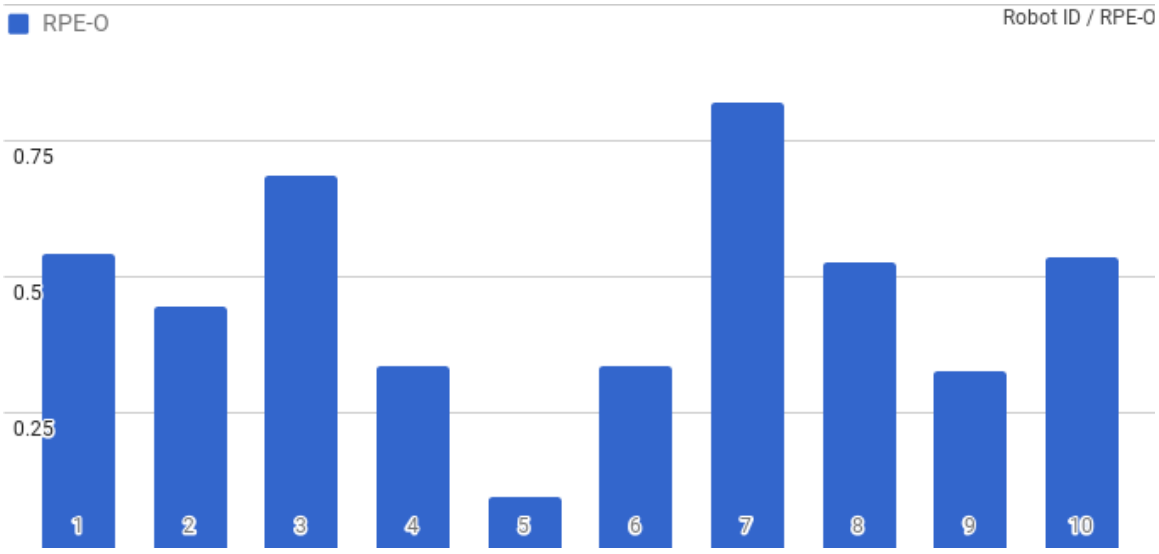


Figure 69: Per-Robot RPE-O comparison with respect to ORB-SLAM2 estimate for data collected in a military training facility lab given in Fig. 67.

6.5 Conclusions

In this chapter, we proposed an approach to distributed object based SLAM with simultaneous object modeling and mapping. We extend the distributed object SLAM approach introduced in Chapter 4 to the case where object models are previously unknown and are modeled jointly with Distributed Object based SLAM. We use the off-the-shelf lightweight convolutional network based object detectors to detect object at categorical level which are then modeled at instance level by integrating detection across frames. The modeled object instance can then be data associated against other instances seen by the same robot or other robots to generate object-object constraint.

We show that this approach generalizes distributed object-based SLAM to unseen object models while further reducing the memory required to store the object models (Tables 18, 19, 20). When compared to the state of art RGB-D mapping approach like ORB-SLAM2, this approach has less memory and communication requirement since we don't have to store or communicate keyframes or dense point clouds. At the same time, the estimate returned by our approach is close to the estimate returned by the distributed RGB-D mapping approach (distributed ORB-SLAM2).

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

In this thesis, we aim at designing a technique that allows each robot in a multi-robot setting to build its own object level map while asking for minimal knowledge of the map of the teammates. We show that using objects as landmarks in a distributed SLAM framework optimized using the state of art distributed optimizer both reduces the communication bandwidth and the memory used by each robot, outputs a human understandable map and improves the robustness and scalability of distributed SLAM.

Our first contribution is the design of a state of art distributed optimizer for distributed pose estimation. Our approach leverages recent results [20] which show that the maximum likelihood trajectory is well approximated by a sequence of two quadratic subproblems. The main contribution of this work (Chapter 3) is to show that these subproblems can be solved in a distributed manner, using the *distributed Gauss-Seidel* (DGS) algorithm. We show that DGS has excellent performance in practice: (i) its communication burden scales linearly in the number of separators and respect agents' privacy, (ii) it is robust to noise and the resulting estimates are sufficiently accurate after few communication rounds, (iii) the approach is simple to implement and scales well to large teams. We demonstrated the effectiveness of the DGS approach in extensive simulations and field tests.

Our second contribution is to extend the DGS approach to using known object models as landmarks for multi robot mapping (Chapter 4). We show that using object-based abstractions in a distributed setup further reduces the memory requirement and the information exchange among teammates. We demonstrate our multi robot object-based mapping approach in Gazebo simulations and in field tests performed in a military training facility.

The approach for object-based mapping introduced in Chapter 4 assumes that a model

of each observed objects is known in advance. However it can be challenging to store a large number of object models, and to account for intra-class variations. Therefore, our third contribution (chapter 5) is to extend our approach to the case where object models are not previously known (at an instance level) and instead object models are jointly optimized within our SLAM framework. We show that this approach requires less memory and scales better with the size of the map as compared to the state of art RGB-D mapping approaches.

Finally, our final contribution (chapter 6) is to extend the distributed object SLAM approach introduced in Chapter 4 to the case where object models are previously unknown and are modeled jointly with Distributed Object based SLAM.

We are currently extending the approach proposed in this paper in several directions. First, our current approach is based on a nonlinear least squares formulation which is not robust to gross outliers. Therefore future work will focus on designing more general algorithms that are robust to spurious measurements. We will in particular investigate distributed implementation of outlier rejection method proposed by Carlone et al. [17] so that they can be applied to distributed systems without requiring all robots to exchange all measurements. Second, our assumption of representing objects using truncated distance function (TDF) can be further explored and estimating an optimal object representation which reduces communication requirement without compromising accuracy is a possible future direction. Third, we currently represent the contextual information around the objects by including additional voxel information around the 3D bounding boxes. We also use the keyframes viewing the objects to further verify the object-object matches. Simpler and elegant ways to encode the contextual information around objects can be studied further in the future works. Fourth, we are working on an incremental version of the Distributed Gauss-Seidel algorithm which can be useful for real-time distributed mapping. Finally, we plan to extend our experimental evaluation to flying robots. While we demonstrated the effectiveness of our approach in large teams of ground robots, we believe that the next grand challenge is to enable coordination and distributed mapping in swarms of agile micro

aerial vehicles with limited communication and computation resources.

REFERENCES

[1] "Deformation-based loop closure for large scale dense rgb-d slam," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, (Tokyo,Japan), 2013.

[2] "Robust real-time visual odometry for dense RGB-D mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2013.

[3] ANDERSON, B., SHAMES, I., MAO, G., and FIDAN, B., "Formal theory of noisy sensor network localization," *SIAM Journal on Discrete Mathematics*, vol. 24, no. 2, pp. 684–698, 2010.

[4] ANDERSSON, L. and NYGARDS, J., "C-SAM : Multi-robot SLAM using square root information smoothing," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2008.

[5] ARAGUES, R., CARLONE, L., CALAFIORE, G., and SAGUES, C., "Multi-agent localization from noisy relative pose measurements," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 364–369, 2011.

[6] ARAGUES, R., CARLONE, L., CALAFIORE, G., and SAGUES, C., "Distributed centroid estimation from noisy relative measurements," *Systems & Control Letters*, vol. 61, no. 7, pp. 773–779, 2012.

[7] ARAGUES, R., CORTES, J., and SAGUES, C., "Distributed consensus on robot networks for dynamically merging feature-based maps," *IEEE Transactions on Robotics*, 2012.

[8] BAHR, A., WALTER, M., and LEONARD, J., "Consistent cooperative localization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 3415–3422, May 2009.

[9] BAILEY, T., BRYSON, M., MU, H., VIAL, J., MCCALMAN, L., and DURRANT-WHYTE, H., "Decentralised cooperative localisation for heterogeneous teams of mobile robots," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[10] BAILEY, T. and DURRANT-WHYTE, H., "Simultaneous localization and mapping (SLAM): part II," *IEEE Robotics Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[11] BAO, S. Y.-Z., BAGRA, M., CHAO, Y.-W., and SAVARESE, S., "Semantic structure from motion with points, regions, and objects," in *CVPR*, pp. 2703–2710, 2012.

[12] BAROOAH, P. and HESPANHA, J., "Semantic structure from motion," in *Intl. Conf. on Intelligent Sensing and Information Processing*, pp. 226–231, 2005.

[13] BAROOAH, P. and HESPANHA, J., "Estimation on graphs from relative measurements," *Control System Magazine*, vol. 27, no. 4, pp. 57–74, 2007.

[14] BENGIO, Y., GOODFELLOW, I. J., and COURVILLE, A., "Deep learning." Book in preparation for MIT Press, 2015.

[15] BERTSEKAS, D. and TSITSIKLIS, J., *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[16] CALAFIORE, G., CARLONE, L., and WEI, M., "A distributed technique for localization of agent formations from relative range measurements," *IEEE Trans. on Systems, Man, and Cybernetics, Part A*, vol. 42, no. 5, pp. 1083–4427, 2012.

[17] CARLONE, L., CENSI, A., and DELLAERT, F., "Selecting good measurements via ℓ_1 relaxation: a convex approach for robust estimation over graphs," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[18] CARLONE, L., NG, M. K., DU, J., BONA, B., and INDRI, M., "Simultaneous localization and mapping using Rao-Blackwellized particle filters in multi robot systems," *J. of Intelligent and Robotic Systems*, vol. 63, no. 2, pp. 283–307, 2011.

[19] CARLONE, L., ROSEN, D., CALAFIORE, G., LEONARD, J., and DELLAERT, F., "Lagrangian duality in 3D SLAM: Verification techniques and optimal solutions," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[20] CARLONE, L., TRON, R., DANIILIDIS, K., and DELLAERT, F., "Initialization techniques for 3D SLAM: a survey on rotation estimation and its use in pose graph optimization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 4597–4604, 2015.

[21] CARRON, A., TODESCATO, M., CARLI, R., and SCHENATO, L., "An asynchronous consensus-based algorithm for estimation from noisy relative measurements," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 3, pp. 2325–5870, 2014.

[22] CHOI, S., ZHOU, Q.-Y., and KOLTUN, V., "Robust reconstruction of indoor scenes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[23] CHOUDHARY, S., CARLONE, L., NIETO, C., ROGERS, J., CHRISTENSEN, H., and DELLAERT, F., "Distributed trajectory estimation with privacy and communication constraints: a two-stage distributed Gauss-Seidel approach," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2016.

[24] CHOUDHARY, S., CARLONE, L., NIETO, C., ROGERS, J., CHRISTENSEN, H., and DELLAERT, F., "Multi robot object-based SLAM," in *Intl. Sym. on Experimental Robotics (ISER)*, 2016.

[25] CHOUDHARY, S., TREVOR, A. J. B., CHRISTENSEN, H. I., and DELLAERT, F., "SLAM with object discovery, modeling and mapping," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, pp. 1018–1025, 2014.

[26] CIVERA, J., GÁLVEZ-LÓPEZ, D., RIAZUELO, L., TARDÓS, J. D., and MONTIEL, J. M. M., "Towards semantic SLAM using a monocular camera," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, CA, USA, September 25-30, 2011*, pp. 1277–1284, 2011.

[27] COLLET ROMEA, A., XIONG, B., GURAU, C., HEBERT, M., and SRINIVASA, S., "Exploiting domain knowledge for object discovery," in *IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, ed.), May 2013.

[28] CUI, Y., SCHUON, S., CHAN, D., THRUN, S., and THEOBALT, C., "3d shape scanning with a time-of-flight camera," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1173–1180, June 2010.

[29] CUNNINGHAM, A., INDELMAN, V., and DELLAERT, F., "DDF-SAM 2.0: Consistent distributed smoothing and mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Karlsruhe, Germany), May 2013.

[30] CUNNINGHAM, A., PALURI, M., and DELLAERT, F., "DDF-SAM: Fully distributed slam using constrained factor graphs," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[31] DAI, A., NIESSNER, M., ZOLLHÖFER, M., IZADI, S., and THEOBALT, C., "Bundlefusion: Real-time globally consistent 3d reconstruction using online surface re-integration," *arXiv preprint arXiv:1604.01093*, 2016.

[32] DAVISON, A. J., REID, I. D., MOLTON, N. D., and STASSE, O., "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[33] DELLAERT, F., "Square Root SAM: Simultaneous location and mapping via square root information smoothing," in *Robotics: Science and Systems (RSS)*, 2005.

[34] DELLAERT, F. and KAESS, M., "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *Intl. J. of Robotics Research*, vol. 25, pp. 1181–1203, Dec 2006.

[35] DELLAERT, F., "Factor graphs and GTSAM: A hands-on introduction," Tech. Rep. GT-RIM-CP&R-2012-002, Georgia Institute of Technology, September 2012.

[36] DENG, J., DONG, W., SOCHER, R., LI, L.-J., LI, K., and FEI-FEI, L., "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[37] DONG, J., NELSON, E., INDELMAN, V., MICHAEL, N., and DELLAERT, F., "Distributed real-time cooperative localization and mapping using an uncertainty-aware expectation maximization approach," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015.

[38] DURRANT-WHYTE, H. and BAILEY, T., "Simultaneous localization and mapping: Part I," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, 2006.

[39] ESTRADA, C., NEIRA, J., and TARDOS, J., "Hierarchical SLAM: Real-time accurate mapping of large environments," *IEEE Trans. Robotics*, vol. 21, pp. 588–596, Aug 2005.

[40] FELZENSZWALB, P. and HUTTENLOCHER, D., "Efficient graph-based image segmentation," *Intl. J. of Computer Vision*, vol. 59, pp. 167–181, 2004.

[41] FINMAN, R., WHELAN, T., KAESS, M., and LEONARD, J. J., "Toward lifelong object segmentation from change detection in dense RGB-D maps," in *2013 European Conference on Mobile Robots, Barcelona, Catalonia, Spain, September 25-27, 2013*, pp. 178–185, 2013.

[42] FOLKESSON, J. and CHRISTENSEN, H., "Graphical SLAM – a self-correcting map," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 1, pp. 383–390, 2004.

[43] FRANCESCHELLI, M. and GASPARRI, A., "On agreement problems with Gossip algorithms in absence of common reference frames," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 337, pp. 4481–4486, 2010.

[44] FRERIS, N. and ZOUZIAS, A., "Fast distributed smoothing of relative measurements," in *IEEE Conf. on Decision and Control*, pp. 1411–1416, 2015.

[45] FRESE, U., "Treemap: An $O(\log n)$ algorithm for indoor simultaneous localization and mapping," *Autonomous Robots*, vol. 21, no. 2, pp. 103–122, 2006.

[46] FRESE, U., LARSSON, P., and DUCKETT, T., "A multilevel relaxation algorithm for simultaneous localisation and mapping," *IEEE Trans. Robotics*, vol. 21, pp. 196–207, April 2005.

[47] GÁLVEZ-LÓPEZ, D., SALAS, M., TARDÓS, J. D., and MONTIEL, J. M. M., "Real-time monocular object SLAM," *Robotics and Autonomous Systems*, vol. 75, pp. 435–449, 2016.

[48] GIRSHICK, R. B., "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015.

[49] GIRSHICK, R. B., DONAHUE, J., DARRELL, T., and MALIK, J., "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pp. 580–587, 2014.

[50] GRISETTI, G., KUEMMERLE, R., STACHNISS, C., FRESE, U., and HERTZBERG, C., "Hierarchical optimization on manifolds for online 2D and 3D mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Anchorage, Alaska), May 2010.

[51] GRISETTI, G., KÜMMERLE, R., and NI, K., "Robust optimization of factor graphs by using condensed measurements," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[52] HATANAKA, T., FUJITA, M., and BULLO, F., "Vision-based cooperative estimation via multi-agent optimization," in *IEEE Conf. on Decision and Control*, 2010.

[53] HENRY, P., KRAININ, M., HERBST, E., REN, X., and FOX, D., "Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments," in *Intl. Sym. on Experimental Robotics (ISER)*, 2010.

[54] HOIEM, D. and SAVARESE, S., *Representations and Techniques for 3D Object Recognition and Scene Interpretation*. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2011.

[55] HOLZER, S., RUSU, R. B., DIXON, M., GEDIKLI, S., and NAVAB, N., "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2684–2689, 2012.

[56] HORNUNG, A., WURM, K. M., BENNEWITZ, M., STACHNISS, C., and BURGARD, W., "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013. Software available at `http://octomap.github.com`.

[57] HOWARD, A., "Multi-robot simultaneous localization and mapping using particle filters," *Intl. J. of Robotics Research*, vol. 25, no. 12, pp. 1243–1256, 2006.

[58] INDELMAN, V., CARLONE, L., and DELLAERT, F., "Planning under uncertainty in the continuous domain: a generalized belief space approach," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014.

[59] INDELMAN, V., GURFIL, P., RIVLIN, E., and ROTSTEIN, H., "Graph-based distributed cooperative navigation for a general multi-robot measurement model," *Intl. J. of Robotics Research*, vol. 31, August 2012.

[60] JIA, Y., SHELHAMER, E., DONAHUE, J., KARAYEV, S., LONG, J., GIRSHICK, R., GUADARRAMA, S., and DARRELL, T., "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[61] KAESS, M., JOHANNSSON, H., ROBERTS, R., ILA, V., LEONARD, J., and DELLAERT, F., "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Intl. J. of Robotics Research*, vol. 31, pp. 217–236, Feb 2012.

[62] KARPATHY, A., MILLER, S., and FEI-FEI, L., "Object discovery in 3d scenes via shape analysis," in *IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, ed.), May 2013.

[63] KENDALL, A., GRIMES, M., and CIPOLLA, R., "Convolutional networks for real-time 6-dof camera relocalization," *CoRR*, vol. abs/1505.07427, 2015.

[64] KERL, C., STURM, J., and CREMERS, D., "Dense visual slam for rgb-d cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2100–2106, Nov 2013.

[65] KERL, C., STURM, J., and CREMERS, D., "Robust odometry estimation for rgb-d cameras," *2013 IEEE International Conference on Robotics and Automation*, pp. 3748–3754, 2013.

[66] KIM, B., KAESS, M., FLETCHER, L., LEONARD, J., BACHRACH, A., ROY, N., and TELLER, S., "Multiple relative pose graphs for robust cooperative mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Anchorage, Alaska), pp. 3185–3192, May 2010.

[67] KIM, Y. M., MITRA, N. J., YAN, D.-M., and GUIBAS, L. J., "Acquiring 3d indoor environments with variability and repetition," *ACM Trans. Graph.*, vol. 31, no. 6, p. 138, 2012.

[68] KNUTH, J. and BAROOAH, P., "Collaborative localization with heterogeneous inter-robot measurements by Riemannian optimization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 1534–1539, 2013.

[69] KOPPULA, H., ANAND, A., JOACHIMS, T., and SAXENA, A., "Semantic labeling of 3d point clouds for indoor scenes," in *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[70] KOSTAVELIS, I. and GASTERATOS, A., "Semantic mapping for mobile robotics tasks: A survey," *Robotics and Autonomous Systems*, vol. 66, pp. 86 – 103, 2015.

[71] KRIZHEVSKY, A., SUTSKEVER, I., and HINTON, G. E., "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pp. 1106–1114, 2012.

[72] KUIPERS, B., "The spatial semantic hierarchy," *Artificial Intelligence*, vol. 119, pp. 191 – 233, 2000.

[73] KUNDU, A., LI, Y., DELLAERT, F., LI, F., and REHG, J. M., *Joint Semantic Segmentation and 3D Reconstruction from Monocular Video*, pp. 703–718. Cham: Springer International Publishing, 2014.

[74] LAI, K., BO, L., and FOX, D., "Unsupervised feature learning for 3d scene labeling," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3050–3057, May 2014.

[75] LAI, K., BO, L., REN, X., and FOX, D., "A large-scale hierarchical multi-view rgb-d object dataset," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1817–1824, May 2011.

[76] LAZARO, M., PAZ, L., PINIES, P., CASTELLANOS, J., and GRISETTI, G., "Multi-robot SLAM using condensed measurements," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 1069–1076, 2011.

[77] LECUN, Y., BOSER, B., DENKER, J. S., HENDERSON, D., HOWARD, R. E., HUBBARD, W., and JACKEL, L. D., "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, pp. 541–551, Dec. 1989.

[78] LEONARD, J. and FEDER, H., "Decoupled stochastic mapping," *IEEE Journal of Oceanic Engineering*, pp. 561–571, October 2001.

[79] LEONARD, J. and NEWMAN, P., "Consistent, convergent, and constant-time SLAM," in *Intl. Joint Conf. on AI (IJCAI)*, 2003.

[80] LIN, T.-Y., MAIRE, M., BELONGIE, S., HAYS, J., PERONA, P., RAMANAN, D., DOLLÃĄR, P., and ZITNICK, C. L., "Microsoft coco: Common objects in context," in *European Conference on Computer Vision (ECCV)*, (ZÃijrich), 2014. Oral.

[81] MARTINEC, D. and PAJDLA, T., "Robust rotation and translation estimation in multiview reconstruction," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, 2007.

[82] MCCORMAC, J., HANDA, A., DAVISON, A. J., and LEUTENEGGER, S., "Semanticfusion: Dense 3D semantic mapping with convolutional neural networks," *CoRR*, vol. abs/1609.05130, 2016.

[83] MU, B., LIU, S. Y., PAULL, L., LEONARD, J., and HOW, J. P., "Slam with objects using a nonparametric pose graph," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4602–4609, Oct 2016.

[84] MUR-ARTAL, RAÚL, M. J. M. M. and TARDÓS, J. D., "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[85] MUR-ARTAL, R. and TARDÓS, J. D., "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *arXiv preprint arXiv:1610.06475*, 2016.

[86] NERURKAR, E., ROUMELIOTIS, S., and MARTINELLI, A., "Distributed maximum a posteriori estimation for multi-robot cooperative localization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 1402–1409, May 2009.

[87] NEWCOMBE, R., DAVISON, A., IZADI, S., KOHLI, P., HILLIGES, O., SHOTTON, J., MOLYNEAUX, D., HODGES, S., KIM, D., and FITZGIBBON, A., "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, pp. 127–136, 2011.

[88] NI, K., STEEDLY, D., and DELLAERT, F., "Tectonic SAM: Exact; out-of-core; submap-based SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Rome; Italy), April 2007.

[89] NI, K. and DELLAERT, F., "Multi-level submap based slam using nested dissection," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[90] NIESSNER, M., ZOLLHÖFER, M., IZADI, S., and STAMMINGER, M., "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, 2013.

[91] NIETO-GRANDA, C., ROGERS III, J. G., TREVOR, A. J. B., and CHRISTENSEN, H. I., "Semantic map partitioning in indoor environments using regional analysis," pp. 1451–1456, Oct 2010.

[92] NUCHTER, A. and HERTZBERG, J., "Towards semantic maps for mobile robots," *Robot. Auton. Syst.*, vol. 56, pp. 915–926, Nov. 2008.

[93] OLFATI-SABER, R., "Swarms on sphere: A programmable swarm with synchronous behaviors like oscillator networks," in *IEEE Conf. on Decision and Control*, pp. 5060–5066, 2006.

[94] PAULL, L., HUANG, G., SETO, M., and LEONARD, J., "Communication-constrained multi-AUV cooperative SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015.

[95] PILLAI, S. and LEONARD, J., "Monocular slam supported object recognition," in *Proceedings of Robotics: Science and Systems (RSS)*, (Rome, Italy), July 2015.

[96] PIOVAN, G., SHAMES, I., FIDAN, B., BULLO, F., and ANDERSON, B., "On frame and orientation localization for relative sensing networks," *Automatica*, vol. 49, no. 1, pp. 206–213, 2013.

[97] PRONOBIS, A. and JENSFELT, P., "Large-scale semantic mapping and reasoning with heterogeneous modalities," in *IEEE International Conference on Robotics and Automation (ICRA)*, may 2012.

[98] RANGANATHAN, A. and DELLAERT, F., "Semantic Modeling of Places using Objects," in *Robotics: Science and Systems (RSS)*, (Atlanta; USA), 2007.

[99] REDMON, J., DIVVALA, S. K., GIRSHICK, R. B., and FARHADI, A., "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[100] REDMON, J. and FARHADI, A., "Yolo9000: Better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016.

[101] ROGERS, J. G., TREVOR, A. J. B., NIETO-GRANDA, C., and CHRISTENSEN, H. I., "Simultaneous localization and mapping with learned object recognition and semantic data association," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1264–1270, Sept 2011.

[102] ROSEN, D., CARLONE, L., BANDEIRA, A., and LEONARD, J., "SE-Sync: A certifiably correct algorithm for synchronization over the special euclidean group," 2016.

[103] ROUMELIOTIS, S. and BEKEY, G., "Distributed multi-robot localization," *IEEE Trans. Robot. Automat.*, August 2002.

[104] RUSINKIEWICZ, S., HALL-HOLT, O., and LEVOY, M., "Real-time 3d model acquisition," *ACM Trans. Graph.*, vol. 21, pp. 438–446, July 2002.

[105] RUSSELL, W., KLEIN, D., and HESPANHA, J., "Optimal estimation on the graph cycle space," *IEEE Trans. Signal Processing*, vol. 59, no. 6, pp. 2834–2846, 2011.

[106] RUSU, R. B., *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Technische Universität München, 2009.

[107] RUSU, R. B., MARTON, Z. C., BLODOW, N., DOLHA, M. E., and BEETZ, M., "Functional object mapping of kitchen environments," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2008.

[108] SALAS-MORENO, R. F., NEWCOMBE, R. A., STRASDAT, H., KELLY, P. H., and DAVISON, A. J., "SLAM++: Simultaneous localisation and mapping at the level of objects," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[109] SARLETTE, A. and SEPULCHRE, R., "Consensus optimization on manifolds," *SIAM J. Control and Optimization*, vol. 48, no. 1, pp. 56–76, 2009.

[110] SEGAL, A., HAEHNEL, D., and THRUN, S., "Generalized-icp," in *Proceedings of Robotics: Science and Systems*, (Seattle, USA), June 2009.

[111] SIMONETTO, A. and LEUS, G., "Distributed maximum likelihood sensor network localization," *IEEE Trans. Signal Processing*, vol. 52, no. 6, 2014.

[112] SINGH, A., SHA, J., NARAYAN, K. S., ACHIM, T., and ABBEEL, P., "Bigbird: A large-scale 3d database of object instances," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 509–516, May 2014.

[113] STEINBRÃIJCKER, F., STURM, J., and CREMERS, D., "Real-time visual odometry from dense rgb-d images," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 719–722, Nov 2011.

[114] STÜCKLER, J. and BEHNKE, S., "Multi-resolution surfel maps for efficient dense 3d modeling and tracking," *J. Vis. Comun. Image Represent.*, vol. 25, pp. 137–147, Jan. 2014.

[115] STURM, J., ENGELHARD, N., ENDRES, F., BURGARD, W., and CREMERS, D., "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[116] SUGER, B., TIPALDI, G., SPINELLO, L., and BURGARD, W., "An Approach to Solving Large-Scale SLAM Problems with a Small Memory Footprint," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2014.

[117] SÜNDERHAUF, N., DAYOUB, F., MCMAHON, S., TALBOT, B., SCHULZ, R., CORKE, P. I., WYETH, G., UPCROFT, B., and MILFORD, M., "Place categorization and semantic mapping on a mobile robot," *CoRR*, vol. abs/1507.02428, 2015.

[118] SÜNDERHAUF, N., DAYOUB, F., SHIRAZI, S., UPCROFT, B., and MILFORD, M., "On the performance of convnet features for place recognition," *CoRR*, vol. abs/1501.04158, 2015.

[119] SÜNDERHAUF, N., PHAM, T., LATIF, Y., MILFORD, M., and REID, I. D., "Meaningful maps - object-oriented semantic mapping," *CoRR*, vol. abs/1609.07849, 2016.

[120] SÜNDERHAUF, N., SHIRAZI, S., JACOBSON, A., DAYOUB, F., PEPPERELL, E., UPCROFT, B., and MILFORD, M., "Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free," in *Robotics: Science and Systems XI, Sapienza University of Rome, Rome, Italy, July 13-17, 2015*, 2015.

[121] SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V., and RABINOVICH, A., "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[122] TATENO, K., TOMBARI, F., and NAVAB, N., "When 2.5d is not enough: Simultaneous reconstruction, segmentation and recognition on dense slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2295–2302, May 2016.

[123] THRUN, S., BURGARD, W., and FOX, D., *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.

[124] THRUN, S. and LIU, Y., "Multi-robot SLAM with sparse extended information filters," in *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*, (Sienna, Italy), Springer, 2003.

[125] THUNBERG, J., MONTIJANO, E., and HU, X., "Distributed attitude synchronization control," in *IEEE Conf. on Decision and Control*, 2011.

[126] TODESCATO, M., CARRON, A., CARLI, R., and SCHENATO, L., "Distributed localization from relative noisy measurements: a robust gradient based approach," in *European Control Conference*, 2015.

[127] TREVOR, A. J. B., ROGERS III, J. G., and CHRISTENSEN, H. I., "Planar surface SLAM with 3D and 2D sensors," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (St. Paul, MN), IEEE, May 2012.

[128] TREVOR, A., ROGERS, J., and CHRISTENSEN, H., "Omnimapper: A modular multimodal mapping framework," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[129] TREVOR, A. J. B., GEDIKLI, S., RUSU, R. B., and CHRISTENSEN, H. I., "Efficient organized point cloud segmentation with connected components," in *Semantic Perception Mapping and Exploration (SPME)*, May 2013.

[130] TRON, R., AFSARI, B., and VIDAL, R., "Intrinsic consensus on SO(3) with almost global convergence," in *IEEE Conf. on Decision and Control*, 2012.

[131] TRON, R., AFSARI, B., and VIDAL, R., "Riemannian consensus for manifolds with bounded curvature," *IEEE Trans. on Automatic Control*, 2012.

[132] TRON, R. and VIDAL, R., "Distributed image-based 3-D localization in camera networks," in *IEEE Conf. on Decision and Control*, 2009.

[133] VALENTIN, J., VINEET, V., CHENG, M.-M., KIM, D., SHOTTON, J., KOHLI, P., NIESSNER, M., CRIMINISI, A., IZADI, S., and TORR, P., "Semanticpaint: Interactive 3D labeling and learning at your fingertips," *ACM Trans. Graph.*, vol. 34, no. 5, pp. 154:1–154:17, 2015.

[134] VINEET, V., MIKSIK, O., LIDEGAARD, M., NIEÃ§NER, M., GOLODETZ, S., PRISACARIU, V. A., KÃĎHLER, O., MURRAY, D. W., IZADI, S., PÃL'REZ, P., and TORR, P. H. S., "Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 75–82, May 2015.

[135] WEI, M., ARAGUES, R., SAGUES, C., and CALAFIORE, G., "Noisy range network localization based on distributed multidimensional scaling," *IEEE Sensors Journals*, vol. 15, no. 3, pp. 854–874, 2015.

[136] WEISE, T., WISMER, T., LEIBE, B., and GOOL, L. V., "In-hand scanning with online loop closure," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pp. 1630–1637, Sept 2009.

[137] WHELAN, T., MCDONALD, J. B., KAESS, M., FALLON, M. F., JOHANNSSON, H., and LEONARD, J. J., "Kintinuous: Spatially extended Kinect-Fusion," in *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, (Sydney, Australia), July 2012.

[138] WHELAN, T., SALAS-MORENO, R. F., GLOCKER, B., DAVISON, A. J., and LEUTENEGGER, S., "Elasticfusion: Real-time dense slam and light source estimation," *Intl. J. of Robotics Research, IJRR*, 2016.

[139] ZENG, A., SONG, S., NIESSNER, M., FISHER, M., XIAO, J., and FUNKHOUSER, T., "3dmatch: Learning local geometric descriptors from rgb-d reconstructions," in *CVPR*, 2017.

[140] ZHAO, L., HUANG, S., and DISSANAYAKE, G., "Linear SLAM: A linear solution to the feature-based and pose graph SLAM based on submap joining," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[141] ZHOU, B., KHOSLA, A., LAPEDRIZA, À., OLIVA, A., and TORRALBA, A., "Object detectors emerge in deep scene cnns," *CoRR*, vol. abs/1412.6856, 2014.

[142] ZHOU, B., LAPEDRIZA, À., XIAO, J., TORRALBA, A., and OLIVA, A., "Learning deep features for scene recognition using places database," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 487–495, 2014.

[143] ZHOU, K., GONG, M., HUANG, X., and GUO, B., "Data-parallel octrees for surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, pp. 669–681, May 2011.

[144] ZHOU, X. and ROUMELIOTIS, S., "Multi-robot SLAM with unknown initial correspondence: The robot rendezvous case," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 1785–1792, IEEE, 2006.