Article

The International Journal of Robotics Research
2017, Vol. 36(12) 1286–1311

DOI: 10.1177/0278364917732640
journals.sagepub.com/home/ijr

Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models

Siddharth Choudhary[1], Luca Carlone[2], Carlos Nieto[1], John Rogers[3], Henrik I Christensen[4] and Frank Dellaert[1]

Abstract

We consider the following problem: a team of robots is deployed in an unknown environment and it has to collaboratively build a map of the area without a reliable infrastructure for communication. The backbone for modern mapping techniques is pose graph optimization*, which estimates the trajectory of the robots, from which the map can be easily built. The first contribution of this paper is a set of distributed algorithms for pose graph optimization: rather than sending all sensor data to a remote sensor fusion server, the robots exchange very partial and noisy information to reach an agreement on the pose graph configuration. Our approach can be considered as a distributed implementation of a two-stage approach that already exists, where we use the Successive Over-Relaxation and the Jacobi Over-Relaxation as workhorses to split the computation among the robots. We also provide conditions under which the proposed distributed protocols converge to the solution of the centralized two-stage approach. As a second contribution, we extend the proposed distributed algorithms to work with the object-based map models. The use of object-based models avoids the exchange of raw sensor measurements (e.g. point clouds or RGB-D data) further reducing the communication burden. Our third contribution is an extensive experimental evaluation of the proposed techniques, including tests in realistic Gazebo simulations and field experiments in a military test facility. Abundant experimental evidence suggests that one of the proposed algorithms (the Distributed Gauss–Seidel method) has excellent performance. The Distributed Gauss–Seidel method requires minimal information exchange, has an anytime flavor, scales well to large teams (we demonstrate mapping with a team of 50 robots), is robust to noise, and is easy to implement. Our field tests show that the combined use of our distributed algorithms and object-based models reduces the communication requirements by several orders of magnitude and enables distributed mapping with large teams of robots in real-world problems. The source code is available for download at https://cognitiverobotics.github.io/distributed-mapper/*

Keywords

SLAM, multi-robot systems, distributed semantic mapping

1. Introduction

The deployment of large teams of cooperative autonomous robots has the potential to enable fast information gathering, and more efficient coverage and monitoring of vast areas. For military applications such as surveillance, reconnaissance, and battle damage assessment, multi-robot systems promise more efficient operation and improved robustness in contested spaces. In civil applications (e.g. pollution monitoring, precision agriculture, search and rescue, disaster response), the use of several inexpensive, heterogeneous, agile platforms is an appealing alternative to monolithic single robot systems.

The deployment of multi-robot systems in the real world poses many technical challenges, ranging from coordination and formation control, to task allocation and distributed sensor fusion. In this paper, we tackle a specific instance of the sensor fusion problem. We consider the case in which a team of robots explore an unknown environment and each robot has to estimate its trajectory from its own sensor data and by leveraging information exchange with the teammates. Trajectory estimation, also called *pose graph optimization*, is relevant as it constitutes the backbone for

[1]College of Computing, Georgia Institute of Technology, USA
[2]Laboratory for Information & Decision Systems, Massachusetts Institute of Technology, USA
[3]US Army Research Laboratory, USA
[4]Institute of Contextual Robotics, UC San Diego, USA

Corresponding author:
Siddharth Choudhary, Institute for Robotics and Intelligent Machines, College of Computing, Georgia Institute of Technology 801 Atlantic Drive NW, Atlanta, GA 30332-3000, USA.
Email: siddharth.choudhary@gatech.edu

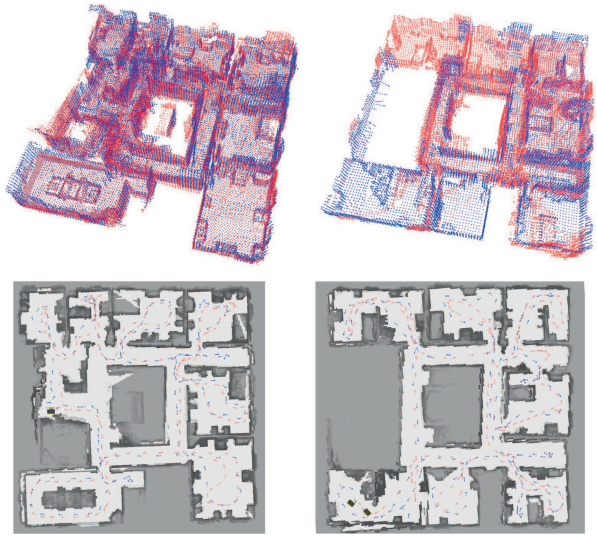


Fig. 1. In our field experiments, distributed trajectory estimation enables 3D reconstruction of an entire building using two robots (red, blue). Each column of the figure shows the reconstructed point cloud of a floor (top), and the estimated trajectories overlaid on an occupancy grid map (bottom). Detailed trajectory comparisons are shown in Figure 29.

many estimation and control tasks (e.g. geo-tagging sensor data, mapping, position-aware task allocation, formation control). Indeed, in our application, trajectory estimation enables distributed 3D mapping and localization (Figure 1).

We consider a realistic scenario, in which the robots can only communicate when they are within a given distance. Moreover, also during a rendezvous (i.e. when the robots are close enough to communicate) they cannot exchange a large amount of information, due to *bandwidth constraints*. Our goal is to design a technique that allows each robot to estimate its own trajectory, while asking for minimal knowledge of the trajectory of its 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. Ideally, we want our distributed mapping approach to scale to very large teams of robots. Our ultimate vision is to deploy a swarm of agile robots (e.g. micro aerial vehicles) that can coordinate by sharing minimal information and using on-board sensing and computation. The present paper takes a step in this direction and presents distributed mapping techniques that are shown to be extremely effective in large simulations (with up to 50 robots) and in real-world problem (with up to four robots).

Contribution. We consider a distributed maximum-likelihood (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, in which robots can assume arbitrary positions and rotations, as this setup is of great interest in many robotics applications (e.g. navigation on uneven terrain, underwater and aerial vehicles). We also consider a fully distributed setup, in which the robots communicate and acquire relative measurements only during rendezvous.

We present two key contributions to solve the distributed mapping problem. *The first contribution is a set of distributed algorithms that enable distributed inference at the estimation back-end*. This contribution is presented in Section 3. Our approach can be understood as a distributed implementation of the *chordal initialization* discussed in Carlone et al. (2015b). The chordal initialization (recalled in Section 3.2) consists in approximating the ML trajectory estimate by solving two quadratic optimization subproblems. The insight of the present work is that these quadratic subproblems can be solved in a distributed fashion, leveraging distributed linear system solvers. 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 Section 2). In Section 3.3 we discuss the conditions under which the distributed algorithms converge to the same estimate of the chordal initialization (Carlone et al., 2015b), which has been extensively shown to be accurate and resilient to measurement noise. We remark that while we use well-known distributed linear system solvers, their application to 3D pose estimation is novel and leads to practical techniques that largely outperform state-of-the-art methods (see Section 5).

The second contribution is the use of high-level object-based models at the estimation front-end and as map representation. This contribution is presented in Section 4. Traditional approaches for multi-robot mapping rely on feature-based maps which are composed of low level primitives like points and lines (Davison et al., 2007). These maps become memory-intensive for long-term operation, contain a lot of redundant information (e.g. it is unnecessary to represent a planar surface with thousands of points), and lack the semantic information necessary for performing wider range of tasks (e.g. manipulation tasks, human-robot interaction). To solve these issues, we present an approach for multi-robot SLAM which uses object landmarks (Salas–Moreno et al., 2013) in a multi-robot mapping setup. We show that this approach further reduces the information exchange among robots, results in compact human-understandable maps, and has lower computational complexity when compared to low-level feature-based mapping.

The third contribution is an extensive experimental evaluation including realistic simulations in Gazebo and field tests in a military facility. This contribution is presented in Section 5. The experiments demonstrate that one of the proposed algorithms, namely the *Distributed Gauss–Seidel*

method[1], provides accurate trajectory estimates, reduces communication overhead, scales to large teams, and is robust to noise. Finally, our field tests show that the combined use of our distribute algorithms and object-based models reduces the communication requirements by several orders of magnitude and enables distributed mapping with large teams of robots. To the best of our knowledge, this is one of the first large-scale demonstrations of multi-robot mapping in realistic, mixed indoor/outdoor scenarios.

Section 6 concludes the paper and discusses current and future work towards real-world robust mapping with large swarms of flying robots.

2. Related work

Multi-robot localization and mapping. Distributed estimation in multi-robot systems is currently an active field of research, with special attention being paid to communication constraints (Paull et al., 2015), heterogeneous teams (Bailey et al., 2011; Indelman et al., 2012), estimation consistency (Bahr et al., 2009), and robust data association (Dong et al., 2015; Indelman et al., 2014). Robotic literature offers distributed implementations of different estimation techniques, including Extended Kalman filters (Roumeliotis and Bekey, 2002; Zhou and Roumeliotis, 2006), information filters (Thrun and Liu, 2003), and particle filters (Carlone et al., 2011; Howard, 2006). 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 linearization errors) and particle filters (e.g. particle depletion), and frame the estimation problem in terms of nonlinear optimization.

In multi-robot systems, ML trajectory estimation can be performed by collecting all measurements at a centralized inference engine, which performs the optimization (Andersson and Nygards, 2008; Bailey et al., 2011; Kim et al., 2010). Variants of these techniques invoke partial exchanges of raw or preprocessed sensor data (Indelman et al., 2014; Lazaro et al., 2011).

In many applications, however, it is not practical to collect all measurements at a single inference engine. When operating in hostile environments, 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 collecting 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. (2009) proposed an algorithm for cooperative localization based on distributed conjugate gradient. Franceschelli and Gasparri (2010) proposed a gossip-based algorithm for distributed pose estimation and investigate its convergence in a noiseless setup. Aragues et al. (2011) used a distributed Jacobi approach to estimate a set of 2D poses, or the centroid of a formation (Aragues et al., 2012a). Aragues et al. (2012b) investigated consensus-based approaches for map merging. Knuth and Barooah (2013) estimated 3D poses using distributed gradient descent. Cunningham et al. (2010) used Gaussian elimination, and developed an approach, called DDF-SAM, in which each robot exchanges a Gaussian marginal over the *separators* (i.e. the variables shared by multiple robots); the approach is further extended in Cunningham et al. (2013), to avoid the storage of redundant data.

Another related body of work is the literature on parallel and hierarchical approaches for mapping. Also in this case, Gaussian elimination and Schur complement have been used as a key component for hierarchical approaches for large-scale mapping (Grisetti et al., 2010; Ni and Dellaert, 2010; Suger et al., 2014). *Decoupled stochastic mapping* was one of the earliest approaches for submapping proposed by Leonard and Feder (2001). Leonard and Newman (2003) proposed a constant-time SLAM solution which achieves near-optimal results under the assumption that the robot makes repeated visits to all regions of the environment. Frese et al. (2005) used multi-level relaxations resulting in a linear time update. Frese (2006) proposed the TreeMap algorithm which divides the environment according to a balanced binary tree. Estrada et al. (2005) presented a hierarchical SLAM approach which consists of a set of local maps and enforces loop consistency when calculating the optimal estimate at the global level. Ni et al. (2007) presented an *exact* submapping approach within a ML framework, and propose to cache the factorization of the submaps to speed-up computation. Grisetti et al. (2010) proposed 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 (2010) extended their previous approach to include multiple levels and use nested dissection to minimize the dependence between two subtrees. Grisetti et al. (2012) computed a good initial estimate for global alignment through a submapping approach. Zhao et al. (2013) proposed an approximation for large-scale SLAM by solving for a sequence of submaps and joining them in a divide-and-conquer manner using linear least squares. Suger et al. (2014) presented an approximate SLAM approach based on hierarchical decomposition to reduce the memory consumption for pose graph optimization.

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 (Paull et al., 2015). 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 (Cunningham et al., 2013). The need of a linearization point also characterizes gradient-based techniques (Knuth and Barooah, 2013). In many practical problems, however, no initial guess is available, and one has to develop ad hoc initialization techniques (Indelman et al., 2014).

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 (Anderson et al., 2010; Calafiore et al., 2012; Simonetto and Leus, 2014; Wei et al., 2015). The case of position estimation from linear measurements is also considered in the literature (Barooah and Hespanha, 2005, 2007; Carron et al., 2014; Freris and Zouzias, 2015; Russell et al., 2011; Todescato et al., 2015); the related problem of *centroid estimation* is tackled in Aragues et al. (2012a).

Distributed rotation estimation has been studied in the context of attitude synchronization (Hatanaka et al., 2010; Olfati–Saber, 2006; Thunberg et al., 2011), camera network calibration (Tron et al., 2012a; Tron and Vidal, 2009), sensor network localization (Piovan et al., 2013), and distributed consensus on manifolds (Sarlette and Sepulchre, 2009; Tron et al., 2012b).

High-level map representations. *Semantic mapping* using high-level object-based representation has gathered a large amount of interest from the robotics community. Kuipers (2000) modeled the environment as a spatial semantic hierarchy, where each level expresses states of partial knowledge corresponding to different levels of representations. Nieto–Granda et al. (2010) 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 (2007) presented a 3D generative model for representing places using objects. The object models are learned in a supervised manner. Civera et al. (2011) proposed a semantic SLAM algorithm that annotates the low-level 3D point based maps with precomputed object models. Rogers et al. (2011) recognized door signs and read their text labels (e.g. room numbers) which are used as landmarks in SLAM. Trevor et al. (2012) used planar surfaces corresponding to walls and tables as landmarks in a mapping system. Bao et al. (2012) modeled semantic structure from motion as a joint inference problem where they jointly recognized and estimated the location of high-level semantic scene components such as regions and objects in 3D. SLAM++, proposed by Salas–Moreno et al. (2013), 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. (2012) used learned object models to reconstruct dense 3D models from a single scan of the indoor scene. Choudhary et al. (2014) 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 and Leonard (2015) developed a SLAM-aware object recognition system which resulted in a considerably stronger recognition performance when compared to related techniques. Gálvez-López et al. (2016) presented a real-time monocular object-based SLAM using a large database of 500 3D objects and showed that exploiting object rigidity both improves the map and the ability to 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. Related work in dense semantic mapping can be seen in the literature (Finman et al., 2013; Koppula et al., 2011; Kundu et al., 2014; McCormac et al., 2016; Nüchter and Hertzberg, 2008; Pronobis and Jensfelt, 2012; Valentin et al., 2015; Vineet et al., 2015).

3. Dealing with bandwidth constraints I: Distributed algorithms

The first contribution of this paper is to devise distributed algorithms that the robots can implement to reach a consensus on a globally optimal trajectory estimate using minimal communication. Section 3.1 introduces the mathematical notation and formalizes the problem. Section 3.2 presents a centralized algorithm, while Section 3.3 presents the corresponding distributed implementations.

3.1. 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 $\mathbf{\Omega} = \{\alpha, \beta, \gamma, \ldots\}$. 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 Figure 2); the pose assumed by robot α at time i is denoted with $\boldsymbol{x}_{\alpha_i}$ (we use Roman letters to denote time

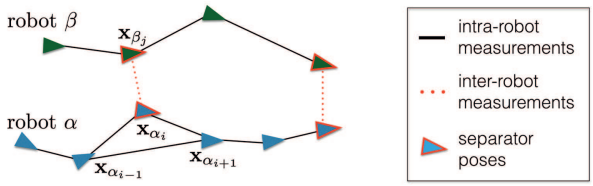


Fig. 2. An instance of multi-robot trajectory estimation: two robots (α in blue, and β in dark green) traverse an unknown environment, collecting intra-robot measurements (solid black lines). During rendezvous, each robot can observe the pose of the other robot (dotted red lines). These are called inter-robot 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.

indices). We are interested in a 3D setup, i.e. $\boldsymbol{x}_{\alpha_i} \in \mathrm{SE}(3)$, where 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 $\boldsymbol{x}_\alpha = [\boldsymbol{x}_{\alpha_i}, \boldsymbol{x}_{\alpha_2}, \ldots]$.

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 Figure 2) or loop closures (which constrain non-consecutive poses, e.g. $\boldsymbol{x}_{\alpha_{i-1}}$ and $\boldsymbol{x}_{\alpha_{i+1}}$ in Figure 2).

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 Figure 2). 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

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

where the relative pose measurement $\bar{\boldsymbol{z}}^{\alpha_i}_{\beta_j}$ includes the relative rotation measurements $\bar{\boldsymbol{R}}^{\alpha_i}_{\beta_j}$, 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 $\bar{\boldsymbol{t}}^{\alpha_i}_{\beta_j}$, 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 $\bar{\boldsymbol{z}}^{\alpha_i}_{\alpha_k}$, for some robot α and for two time instants $i \neq k$; inter-robot measurements, instead, are in the form $\bar{\boldsymbol{z}}^{\alpha_i}_{\beta_j}$ for two robots $\alpha \neq \beta$.

In the following, we denote with $\mathcal{E}^\alpha_I$ 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. $\mathcal{E}_I = \cup_{\alpha \in \boldsymbol{\Omega}} \mathcal{E}^\alpha_I$. The set of inter-robot measurements involving robot α is denoted with $\mathcal{E}^\alpha_S$ (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 $\mathcal{E}^\alpha_I$ and $\mathcal{E}^\alpha_S$.

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

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

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 Carlone et al. (2015a) 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 (Carlone et al., 2015a) that the ML estimate $\boldsymbol{x} \doteq \{(\boldsymbol{R}_{\alpha_i}, \boldsymbol{t}_{\alpha_i}), \forall \alpha \in \boldsymbol{\Omega}, \forall i\}$[2] can be computed as solution of the following optimization problem

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

The peculiarity of (3) is the use of the *chordal distance* $\|\boldsymbol{R}_{\beta_j} - \boldsymbol{R}_{\alpha_i} \bar{\boldsymbol{R}}^{\alpha_i}_{\beta_j}\|_{\mathsf{F}}$ to quantify rotation errors, while the majority of related works in robotics uses the *geodesic distance*, see Carlone et al. (2015b). We remark that the chordal distance closely approximates the geodesic distance for small residual errors, as shown in Carlone and Dellaert (2015), and produces similar estimates in practice. We refer the reader to the surveys in Hartley et al. (2013) and Carlone et al. (2015b) for a more general overview on rotation estimation.

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 (Dellaert, 2012), fast approximations (Carlone et al., 2015b), or convex relaxations (Rosen et al., 2016).

In this paper, 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 $\mathbf{\Omega}_r \subseteq \mathbf{\Omega} \setminus \{\alpha\}$, and that*

- *takes as input: (i) the intra-robot measurements $\mathcal{E}_I^\alpha$; (ii) the subset of inter-robot measurements $\mathcal{E}_S^\alpha$; (iii) partial estimates of the trajectory of robots $\beta \in \mathbf{\Omega}_r$;*
- *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 $\beta \in \mathbf{\Omega}_r$ requires communication, hence we want our distributed algorithm to exchange a very small portion of the trajectory estimates.

The following 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 Carlone et al. (2015b) to the multi-robot case. Then we tailor the discussion to the distributed setup in Section 3.3.

3.2. 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 Carlone et al. (2015b) (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 a distributed fashion, exploiting existing distributed linear system solvers. In the rest of this section we review the two-stage approach of Carlone et al. (2015b), while we discuss the use of distributed solvers in Section 3.3.

The two-stage approach of Carlone et al. (2015b) 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

$$\min_{\substack{\boldsymbol{R}_{\alpha_i} \in \mathrm{SO}(3) \\ \forall \alpha \in \mathbf{\Omega}, \forall i}} \sum_{(\alpha_i, \beta_j) \in \mathcal{E}} \omega_R^2 \left\| \boldsymbol{R}_{\beta_j} - \boldsymbol{R}_{\alpha_i} \bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i} \right\|_{\mathsf{F}}^2 \tag{4}$$

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 (4) is non-convex (due to the non-convex 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 Martinec and Pajdla (2007) and reviewed in Carlone et al. (2015b). 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 SO(3). In formulas, the quadratic relaxation is

$$\min_{\substack{\boldsymbol{R}_{\alpha_i} \in \mathbb{R}^{3\times 3} \\ \forall \alpha \in \mathbf{\Omega}, \forall i}} \sum_{(\alpha_i, \beta_j) \in \mathcal{E}} \omega_R^2 \left\| \boldsymbol{R}_{\beta_j} - \boldsymbol{R}_{\alpha_i} \bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i} \right\|_{\mathsf{F}}^2 \tag{5}$$

which simply rewrites (4) without the constraints. Since (5) is quadratic in the unknown rotations $\boldsymbol{R}_{\alpha_i}, \forall \alpha \in \mathbf{\Omega}, \forall i$, we can rewrite it as

$$\min_{\boldsymbol{r}} \|\boldsymbol{A}_r \boldsymbol{r} - \boldsymbol{b}_r\|^2 \tag{6}$$

where we stacked all the entries of the unknown rotation matrices $\boldsymbol{R}_{\alpha_i}, \forall \alpha \in \mathbf{\Omega}, \forall i$ 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 non-zero vector $\boldsymbol{b}_r$ follows from setting one of the rotations to be the reference frame, e.g. $\boldsymbol{R}_{\alpha_i} = \mathbf{I}_3$).

Since (5) is a linear least-squares problem, its solution can be found by solving the normal equations

$$(\boldsymbol{A}_r^\mathsf{T} \boldsymbol{A}_r)\, \boldsymbol{r} = \boldsymbol{A}_r^\mathsf{T} \boldsymbol{b}_r \tag{7}$$

Let us denote with $\breve{\boldsymbol{r}}$ the solution of (7). Rewriting $\breve{\boldsymbol{r}}$ in matrix form, we obtain the matrices $\breve{\boldsymbol{R}}_{\alpha_i}$, $\forall \alpha \in \mathbf{\Omega}, \forall i$. Since these rotations were obtained from a relaxation of (4), they are not guaranteed to satisfy the constraints $\boldsymbol{R}_{\alpha_i} \in \mathrm{SO}(3)$; therefore the approach (Martinec and Pajdla, 2007) projects them to SO(3), and gets the rotation estimate $\hat{\boldsymbol{R}}_{\alpha_i} = \texttt{project}(\breve{\boldsymbol{R}}_{\alpha_i})$, $\forall \alpha \in \mathbf{\Omega}, \forall i$. The projection only requires to perform an SVD of $\breve{\boldsymbol{R}}_{\alpha_i}$ and can be performed independently for each rotation (Carlone et al., 2015b).

Stage 2: Full pose recovery via single GN iteration. In the previous stage we obtained an estimate for the rotations $\hat{\boldsymbol{R}}_{\alpha_i}, \forall \alpha \in \mathbf{\Omega}, \forall i$. 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 $\mathrm{Exp}\,(\cdot)$ is the exponential map for SO(3), and $\boldsymbol{\theta}_{\alpha_i} \in \mathbb{R}^3$ (this is our new parametrization for the rotations). With this parametrization, equation (3) becomes

$$\begin{aligned} \min_{\substack{\boldsymbol{t}_{\alpha_i}, \boldsymbol{\theta}_{\alpha_i} \\ \forall \alpha \in \mathbf{\Omega}, \forall i}} & \sum_{(\alpha_i, \beta_j) \in \mathcal{E}} \omega_t^2 \left\| \boldsymbol{t}_{\beta_j} - \boldsymbol{t}_{\alpha_i} - \hat{\boldsymbol{R}}_{\alpha_i} \mathrm{Exp}\left(\boldsymbol{\theta}_{\alpha_i}\right) \bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i} \right\|^2 \\ & + \frac{\omega_R^2}{2} \left\| \hat{\boldsymbol{R}}_{\beta_j} \mathrm{Exp}\left(\boldsymbol{\theta}_{\beta_j}\right) - \hat{\boldsymbol{R}}_{\alpha_i} \mathrm{Exp}\left(\boldsymbol{\theta}_{\alpha_i}\right) \bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i} \right\|_{\mathsf{F}}^2 \end{aligned} \tag{8}$$

Note that the reparametrization allowed us to drop the constraints (we are now trying to estimate vectors in $\mathbb{R}^3$), but moved the non-convexity to the objective ($\mathrm{Exp}\,(\cdot)$ is non-linear in its argument). In order to solve (8), we take a

quadratic approximation of the cost function. For this purpose, we use the following first-order approximation of the exponential map

$$\mathrm{Exp}\left(\boldsymbol{\theta}_{\alpha_i}\right) \simeq \mathbf{I}_3 + \boldsymbol{S}(\boldsymbol{\theta}_{\alpha_i}) \tag{9}$$

where $\boldsymbol{S}(\boldsymbol{\theta}_{\alpha_i})$ is a skew symmetric matrix whose entries are defined by the vector $\boldsymbol{\theta}_{\alpha_i}$. Substituting (9) into (8) we get the desired quadratic approximation

$$\begin{aligned}\min_{\substack{\boldsymbol{t}_{\alpha_i},\boldsymbol{\theta}_{\alpha_i}\in\mathbb{R}^3\\ \forall\alpha\in\boldsymbol{\Omega},\forall i}} \sum_{(\alpha_i,\beta_j)\in\mathcal{E}} \omega_t^2 \left\| \boldsymbol{t}_{\beta_j} - \boldsymbol{t}_{\alpha_i} - \hat{\boldsymbol{R}}_{\alpha_i}\bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i} - \hat{\boldsymbol{R}}_{\alpha_i}\boldsymbol{S}(\boldsymbol{\theta}_{\alpha_i})\,\bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i}\right\|^2 \\ + \frac{\omega_R^2}{2}\left\| \hat{\boldsymbol{R}}_{\beta_j} - \hat{\boldsymbol{R}}_{\alpha_i}\bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i} + \hat{\boldsymbol{R}}_{\beta_j}\boldsymbol{S}(\boldsymbol{\theta}_{\beta_j}) - \hat{\boldsymbol{R}}_{\alpha_i}\boldsymbol{S}(\boldsymbol{\theta}_{\alpha_i})\,\bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i}\right\|_{\mathsf{F}}^2\end{aligned} \tag{10}$$

Rearranging the unknown $\boldsymbol{t}_{\alpha_i}, \boldsymbol{\theta}_{\alpha_i}$ of all robots into a single vector $\boldsymbol{p}$, we rewrite (10) as a linear least-squares problem

$$\min_{\boldsymbol{p}} \quad \|\boldsymbol{A}_p\,\boldsymbol{p} - \boldsymbol{b}_p\|^2 \tag{11}$$

whose solution can be found by solving the linear system

$$(\boldsymbol{A}_p^{\mathsf{T}}\boldsymbol{A}_p)\boldsymbol{p} = \boldsymbol{A}_p^{\mathsf{T}}\boldsymbol{b}_p \tag{12}$$

From the solution of (12) 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: $\boldsymbol{R}_{\alpha_i} = \hat{\boldsymbol{R}}_{\alpha_i}\mathrm{Exp}\left(\boldsymbol{\theta}_{\alpha_i}\right)$.

Remark 1. Advantage of Centralized Two-Stage Approach. *The approach reviewed in this section has three advantages. First, as shown in Carlone et al. (2015b), 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 solving 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 (Carlone et al., 2015b)) or is unavailable.* ■

3.3. Distributed pose graph optimization

In this section we show that the two-stage approach described in Section 3.2 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 a workhorse to split the computation among the robots. For instance, one could adapt the Gaussian elimination approach of Cunningham et al. (2010) to solve the linear systems (7) and (12). In this section we propose alternative approaches, based on the Distributed Jacobi Over-Relaxation and Distributed Successive Over-Relaxation algorithms, and we discuss their advantages.

In both (7) and (12) 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 (7), 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 (12), such that $\boldsymbol{p}_\alpha$ describes the trajectory of robot α. Therefore, (7) and (12) can be framed in the general form

$$\boldsymbol{H}\boldsymbol{y} = \boldsymbol{g} \Leftrightarrow \begin{bmatrix} \boldsymbol{H}_{\alpha\alpha} & \boldsymbol{H}_{\alpha\beta} & \cdots \\ \boldsymbol{H}_{\beta\alpha} & \boldsymbol{H}_{\beta\beta} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_\alpha \\ \boldsymbol{y}_\beta \\ \vdots \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}_\alpha \\ \boldsymbol{g}_\beta \\ \vdots \end{bmatrix} \tag{13}$$

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 (13) 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 (13) 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

$$\boldsymbol{H}_{\alpha\alpha}\boldsymbol{y}_\alpha = -\sum_{\delta\in\boldsymbol{\Omega}\setminus\{\alpha\}} \boldsymbol{H}_{\alpha\delta}\boldsymbol{y}_\delta + \boldsymbol{g}_\alpha \qquad \forall\alpha \in \boldsymbol{\Omega} \tag{14}$$

The set of equations (14) is the same as the original system (13), but clearly exposes the contribution of the variables associated to each robot. The equations (14) constitute the basis for the *Successive Over-Relaxation* (SOR) and the *Jacobi Over-Relaxation* (JOR) methods that we describe in the following sections.

3.3.1. Distributed Jacobi over-relaxation (JOR). The distributed JOR algorithm (Bertsekas and Tsitsiklis, 1989) starts at an arbitrary initial estimate $\boldsymbol{y}^{(0)} = [\boldsymbol{y}_\alpha^{(0)}, \boldsymbol{y}_\beta^{(0)}, \ldots]$ and solves the linear system (13) by repeating the following iterations

$$\begin{aligned}\boldsymbol{y}_\alpha^{(k+1)} &= (1-\gamma)\boldsymbol{y}_\alpha^{(k)} \\ &+ (\gamma)\boldsymbol{H}_{\alpha\alpha}^{-1}\left(-\sum_{\delta\in\boldsymbol{\Omega}\setminus\{\alpha\}} \boldsymbol{H}_{\alpha\delta}\boldsymbol{y}_\delta^{(k)} + \boldsymbol{g}_\alpha\right) \quad \forall\alpha \in \boldsymbol{\Omega}\end{aligned} \tag{15}$$

where γ is the *relaxation factor*. Intuitively, at each iteration robot α attempts to solve equation (14) (the second summand in (15) is the solution of (14) with the estimates of the other robots kept fixed), while remaining close to the previous estimate $\boldsymbol{y}_\alpha^{(k)}$ (first summand in (15)). If the iterations (15) converge to a fixed point, say $\boldsymbol{y}_\alpha$ $\forall\alpha$, then the resulting estimate solves the linear system (14)

exactly (Bertsekas and Tsitsiklis, 1989, page 131). To prove this fact we only need to rewrite (15) 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 (14).

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 ones 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 equation (13). Using the distributed JOR approach, the robots solve (7) and (12) 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 (15) 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 (Bertsekas and Tsitsiklis, 1989). *Consider the linear systems* (13) *and define the block diagonal matrix* $\boldsymbol{D} \doteq diag\left(\boldsymbol{H}_{\alpha\alpha}, \boldsymbol{H}_{\beta\beta}, \ldots\right)$. *Moreover, define the matrix*

$$\boldsymbol{M} = (1-\gamma)\mathbf{I} - \gamma\boldsymbol{D}^{-1}(\boldsymbol{H}-\boldsymbol{D}) \tag{16}$$

where $\mathbf{I}$ *is the identity matrix of suitable size. Then, the JOR iterations* (15) *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.*

This proposition is the same as Proposition 6.1 in Bertsekas and Tsitsiklis (1989) (the condition that $\mathbf{I} - \boldsymbol{M}$ is invertible is guaranteed to hold as noted in the footnote on page 144 of Bertsekas and Tsitsiklis (1989)).

It is non-trivial to establish whether our linear systems (7) and (12) 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.

3.3.2. Distributed successive over-relaxation (SOR). The distributed SOR algorithm (Bertsekas and Tsitsiklis, 1989) 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 $\alpha \in \boldsymbol{\Omega}$

$$\boldsymbol{y}_\alpha^{(k+1)} = (1-\gamma)\boldsymbol{y}_\alpha^{(k)} + (\gamma)\boldsymbol{H}_{\alpha\alpha}^{-1}\left(-\sum_{\delta\in\boldsymbol{\Omega}_\alpha^+}\boldsymbol{H}_{\alpha\delta}\boldsymbol{y}_\delta^{(k+1)} - \sum_{\delta\in\boldsymbol{\Omega}_\alpha^-}\boldsymbol{H}_{\alpha\delta}\boldsymbol{y}_\delta^{(k)} + \boldsymbol{g}_\alpha\right) \tag{17}$$

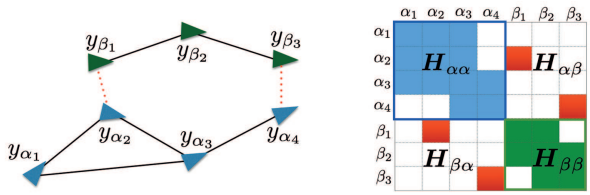


Fig. 3. Example: (left) trajectory estimation problem and (right) corresponding block structure of the matrix $\boldsymbol{H}$.

where γ is the *relaxation factor*, $\boldsymbol{\Omega}_\alpha^+$ is the set of robots which have already computed the $(k+1)$-th estimate, while $\boldsymbol{\Omega}_\alpha^-$ is the set of robots that still have to perform the update (17), excluding node α (intuitively: each robot uses the latest estimate). As for the JOR algorithm, by comparing (17) and (14), we see that if the sequence produced by the iterations (17) converges to a fixed point, then such point satisfies (14), and indeed solves the original linear system (13). When $\gamma = 1$, the distributed SOR method is known as the *distributed Gauss–Seidel* (DGS) method.

The following proposition, whose proof trivially follows from Proposition 6.10, p. 154 of Bertsekas and Tsitsiklis (1989) (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* (17) *applied to the linear systems* (7) *and* (12) *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 Proposition 6.10, p. 154 of Bertsekas and Tsitsiklis (1989), for $\gamma \notin (0, 2)$, the SOR iterations (17) 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.

3.3.3. Communication requirements for JOR and SOR. In this section we observe that to execute the JOR and SOR iterations (15) and (17), 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 Figure 3 robot α only needs the estimates of $\boldsymbol{y}_{\beta_1}$ and $\boldsymbol{y}_{\beta_3}$, while it does not require any knowledge about the other poses of β.

To understand this fact, we note that both (7) and (12) model an estimation problem from pairwise relative measurements. It is well known that the matrix $\boldsymbol{H}$ (sometimes called the *Hessian* (Dellaert, 2005)) underlying these problems has a block structure defined by the Laplacian matrix of the underlying graph (Barooah and Hespanha, 2007). For instance, Figure 3 (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 (15) iterations as

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

where we simply removed the contributions of the zero blocks from the sum in (15).

Similarly we can simplify the SOR (17) iterations as

$$\boldsymbol{y}_\alpha^{(k+1)} = (1-\gamma)\boldsymbol{y}_\alpha^{(k)} + (\gamma)\boldsymbol{H}_{\alpha\alpha}^{-1}\left(-\sum_{(\alpha_i,\delta_j)\in\mathcal{E}_S^{\alpha+}} \boldsymbol{H}_{\alpha_i\delta_j}\boldsymbol{y}_{\delta_j}^{(k+1)} - \sum_{(\alpha_i,\delta_j)\in\mathcal{E}_S^{\alpha-}} \boldsymbol{H}_{\alpha_i\delta_j}\boldsymbol{y}_{\delta_j}^{(k)} + \boldsymbol{g}_\alpha\right) \quad (19)$$

where we removed the contributions of the zero blocks from the sum in (17); 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 the robots which already performed the $(k+1)$-th iteration, while $\mathcal{E}_S^{\alpha-}$ is the set of measurements involving the robots which have not performed the iteration yet (as before: each robot simply uses its latest estimate).

Equations (18) and (19) 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.

3.3.4. 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 (18) or (19) requires the robots to exchange their estimate of the separators).

In this work, we follow the path of Barooah and Hespanha (2005) 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 (18) or (19) without considering the inter-robot measurements, i.e. equations (18)-(19) 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, β also 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 (18) or SOR (19) update. Barooah and Hespanha (2005) 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.

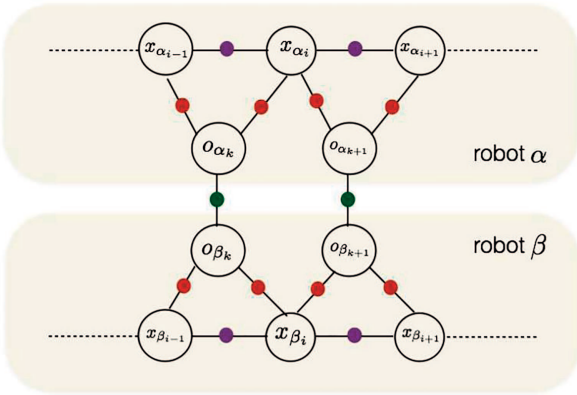


Fig. 4. Factor graph representation of Multi-Robot Object based SLAM. $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{x}_{\beta_i}$ denote the poses assumed by robot α and β at time i respectively. The pose of the k^{th} object as estimated by robot α and β is denoted with $\boldsymbol{o}_k^\alpha$ and $\boldsymbol{o}_k^\beta$ respectively. Green dots show inter-robot factors whereas orange and purple dots shows intra-robot factors.

4. Dealing with bandwidth constraints II: Compressing sensor data via object-based representations

The second contribution of this paper is the use of high-level object-based models at the estimation front-end and as a map representation for multi-robot SLAM. Object-based abstractions are crucial to further reduce the memory storage and the information exchange among the robots.

Previous approaches for multi-robot mapping rely on feature-based maps which become memory-intensive for long-term operations, contain a large amount of redundant information, and lack the semantic understanding necessary to perform a wider range of tasks (e.g. manipulation, human-robot interaction). To solve these issues, we present an approach for multi-robot SLAM which uses object landmarks (Salas–Moreno et al., 2013) in a multi-robot mapping setup.

Section 4.1 introduces the additional mathematical notation and formalizes the problem of distributed object-based SLAM. Section 4.2 presents the implementation details of our distributed object-based SLAM system.

4.1. Distributed object-based SLAM

We consider a multi-robot system as defined in Section 3.1. 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 $\boldsymbol{x}_\alpha = [\boldsymbol{x}_{\alpha_i}, \boldsymbol{x}_{\alpha_2}, \ldots]$. In addition, we denote with $\boldsymbol{o}_k^\alpha \in \mathrm{SE}(3)$ the pose of the k^{th} object as estimated by robot α (Figure 4).

Measurements. Similar to distributed pose graph optimization (Section 3.1), 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 Figure 4), and the object measurements which constrains robot poses with the corresponding visible object landmarks (e.g. $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{o}_k^\alpha$ in Figure 4). Contrarily to Section 3.1, 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 Figure 4 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 equation (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

$$\bar{\boldsymbol{z}}_{\boldsymbol{o}_k^\beta}^{\boldsymbol{o}_k^\alpha} \doteq (\mathbf{I}, \mathbf{0}), \quad \text{with:} \begin{cases} \boldsymbol{R}_{\boldsymbol{o}_k^\beta}^{\boldsymbol{o}_k^\alpha} = (\boldsymbol{R}_{\boldsymbol{o}_k^\alpha})^\mathsf{T} \boldsymbol{R}_{\boldsymbol{o}_k^\beta} \boldsymbol{R}_\epsilon = \mathbf{I} \\ \bar{\boldsymbol{t}}_{\boldsymbol{o}_k^\beta}^{\boldsymbol{o}_k^\alpha} = (\boldsymbol{R}_{\boldsymbol{o}_k^\alpha})^\mathsf{T} (\boldsymbol{t}_{\boldsymbol{o}_k^\beta} - \boldsymbol{t}_{\boldsymbol{o}_k^\alpha}) + \boldsymbol{t}_\epsilon = \mathbf{0} \end{cases} \tag{20}$$

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 3.1, 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

$$\begin{aligned} \boldsymbol{x}^\star, \boldsymbol{o}^\star = \arg\max_{\boldsymbol{x}, \boldsymbol{o}} & \prod_{(\boldsymbol{x}_{\alpha_i}, \boldsymbol{x}_{\alpha_{i+1}}) \in \mathcal{E}_I} \underbrace{\mathcal{L}(\bar{\boldsymbol{z}}_{\boldsymbol{x}_{\alpha_{i+1}}}^{\boldsymbol{x}_{\alpha_i}} \mid \boldsymbol{x})}_{\text{odometry factors}} \\ & \prod_{(\boldsymbol{x}_{\alpha_i}, \boldsymbol{o}_k^\alpha) \in \mathcal{E}_O} \underbrace{\mathcal{L}(\bar{\boldsymbol{z}}_{\boldsymbol{o}_k^\alpha}^{\boldsymbol{x}_{\alpha_i}} \mid \boldsymbol{x}, \boldsymbol{o})}_{\text{intra-robot object-measurement factors}} \\ & \prod_{(\boldsymbol{o}_i^\alpha, \boldsymbol{o}_j^\beta) \in \mathcal{E}_S} \underbrace{\mathcal{L}(\bar{\boldsymbol{z}}_{\boldsymbol{o}_j^\beta}^{\boldsymbol{o}_i^\alpha} \mid \boldsymbol{x}, \boldsymbol{o})}_{\text{inter-robot object-object factors}} \end{aligned} \tag{21}$$

where we used the same assumptions on measurement noise as in Section 3.1. Defining $\mathcal{X} = \boldsymbol{x} \cup \boldsymbol{o}$, we rewrite equation (21) as

$$\mathcal{X}^\star = \arg\max_{\mathcal{X}} \prod_{(\alpha_i, \beta_j) \in \mathcal{E}} \mathcal{L}(\bar{\boldsymbol{z}}_{\beta_j}^{\alpha_i} \mid \mathcal{X}) \tag{22}$$

Since the optimization problem in equation (22) has the same structure of the one in equation (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.

4.2. Object-based SLAM implementation

Object detection and pose estimation. Each robot collects RGB-D data using a depth camera, and measures its ego-motion through wheel odometry. In our approach, each RGB frame (from RGB-D) is passed to the you only look once (YOLO) object detector (Redmon et al., 2015), 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 (Singh et al. (2014)). 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* (Trevor et al., 2013). The segmented object is then matched to the 3D template of the detected object class to estimate its pose.

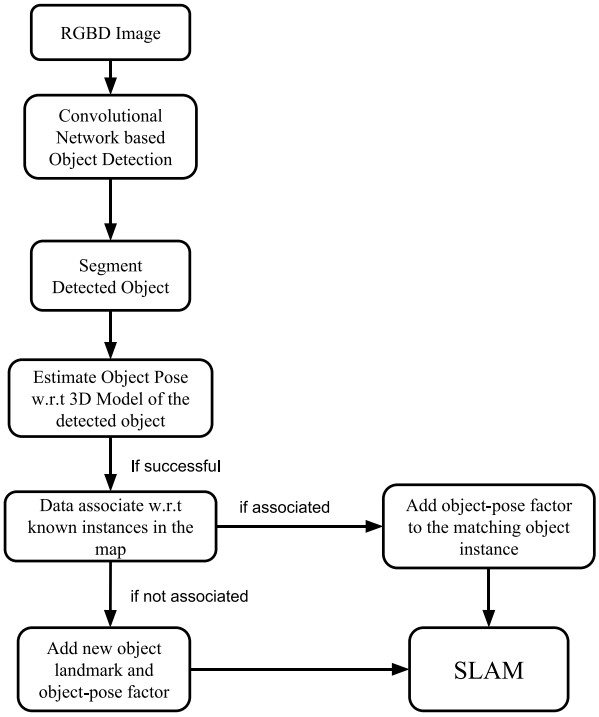


Fig. 5. Flowchart of object-based SLAM.

We extract Persistent Feature Histograms (PFHRGB) features (Rusu et al., 2008) in the source (object segment) and target (object model) point clouds and register the two point clouds in a Sample Consensus Initial Alignment framework (Rusu, 2009). If we have at least 12 inlier correspondences, GICP (generalized iterative closest point Segal et al. (2009) 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 a 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 (Trevor et al., 2012). 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 Figure 5.

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.[3] 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 a 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.

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

Fig. 6. Simulated 3D datasets with different number of robots. Robots are shown in different colors. Gray links denote inter-robot measurements.

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.

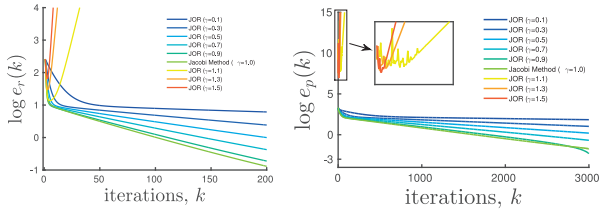
5. Experiments

We evaluate the distributed JOR and SOR along with DJ and DGS approaches (with and without using objects) in large-scale simulations (Sections 5.1 and 5.2) and field tests (Sections 5.3 and 5.4). The results demonstrate that: (i) the DGS dominates the other algorithms considered in this paper in terms of convergence speed; (ii) the DGS algorithm is accurate, scalable, and robust to noise; (iii) the DGS requires less communication than techniques from related work (i.e. DDF-SAM); (iv) in real applications, the combination of DGS and object-based mapping reduces the communication requirements by several orders of magnitude compared to approaches exchanging raw measurements.

5.1. Simulation results: Multi-robot pose graph optimization

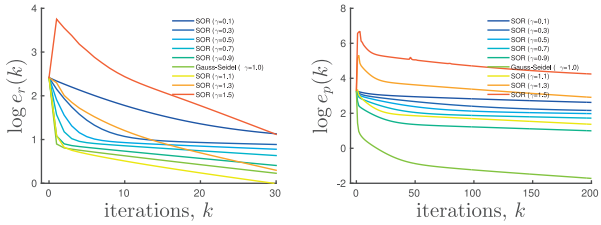
In this section, we characterize the performance of the proposed approaches in terms of convergence, scalability (in the number of robots and separators), and sensitivity to noise.

Simulation setup and performance metrics. For our tests, we created simulation datasets in six different configurations with an 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, as shown in Figure 6. When



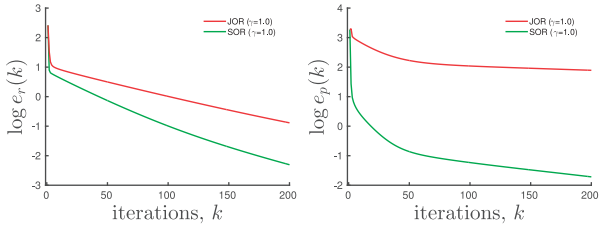
(a) Rotation estimation (b) Pose estimation

Fig. 7. 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 $\gamma > 1$.



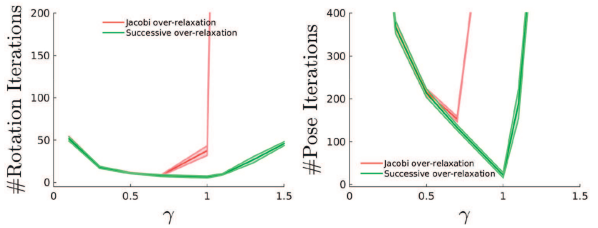
(a) Rotation error (b) Pose error

Fig. 8. SOR: convergence of (a) rotation estimation and (b) pose estimation for different values of γ (grid scenario, 49 robots).



(a) Rotation error (b) Pose error

Fig. 9. JOR vs SOR: convergence of (a) rotation estimation and (b) pose estimation for the JOR and SOR algorithms with $\gamma = 1$ (grid scenario, 49 robots).



(a) Rotation estimation (b) Pose estimation

Fig. 10. JOR vs 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.

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, (7) and (12), which return the minimizers of (6) and (11), 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 (7)

$$e_r(k) = \|\boldsymbol{A}_r\boldsymbol{r}^{(k)} - \boldsymbol{b}_r\|^2 - m_r \tag{23}$$

$e_r(k)$ 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

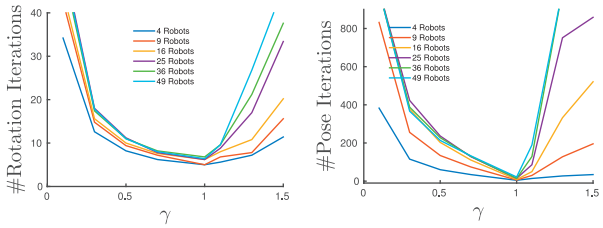
$$e_p(k) = \|\boldsymbol{A}_p\boldsymbol{p}^{(k)} - \boldsymbol{b}_p\|^2 - m_p \tag{24}$$

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.

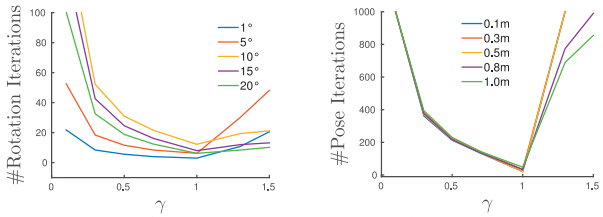
Comparisons among the distributed algorithms. In this section, we consider the scenario with 49 robots. We start by studying the convergence properties of the JOR and SOR algorithms in isolation. Then we compare the two algorithms in terms of convergence speed. Figure 7 shows the rotation and the pose error versus the number of iterations for different choices of the parameter γ for the JOR algorithm. Figure 7a confirms the result of Proposition 2: JOR applied to the rotation subproblem converges as long as $\gamma \leq 1$. Figure 7a 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. Figure 8 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 $\gamma \in (0, 2)$. Figure 8a shows that choices of γ close to 1 ensure fast convergence rates, while Figure 8b 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. Figure 9 compares the convergence rate of SOR and



(a) Rotation error (b) Pose error

Fig. 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 number of robots.



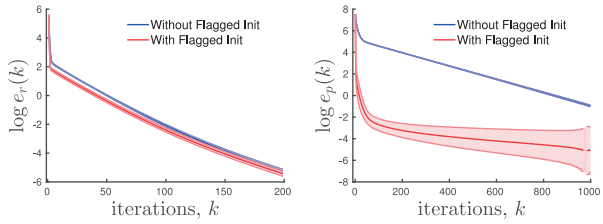
(a) Rotation error (b) Pose error

Fig. 12. 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.

JOR for both the rotation subproblem (Figure 9a) and the pose subproblem (Figure 9b). 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. Figure 10 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.

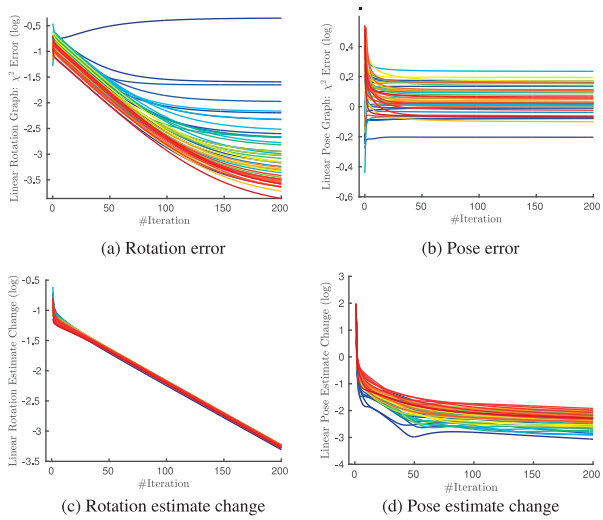
We conclude this section by showing that setting $\gamma = 1$ in SOR ensures faster convergence regardless the number of robots and the measurement noise. Figure 11 compares the number of iterations required to converge for an increasing number of robots for varying γ values. Similarly Figure 12 compares the number of iterations required to converge for increasing noise for varying γ values. 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 paper 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 Figure 13. In both cases the estimation errors go to zero, but the convergence is faster when



(a) Rotation error (b) Pose error

Fig. 13. 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.



(a) Rotation error (b) Pose error

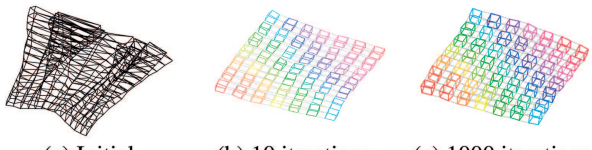
(c) Rotation estimate change (d) Pose estimate change

Fig. 14. DGS: convergence statistics of rotation estimation and pose estimation for each robot (49 Robots). Robots are represented by different color lines.

using the flagged initialization. The speed-up is significant for the second linear system (Figure 13b). We noticed a similar advantage across all tested scenarios. Therefore, in the rest of the paper we always adopt the flagged initialization.

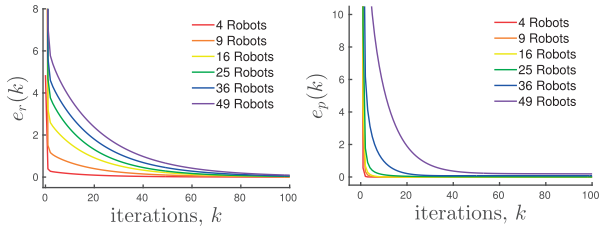
Stopping conditions and anytime flavor. This section provides extra insights on the convergence of the DGS method. Figure 14a to Figure 14b shows the evolution of the rotation and pose error for *each* robot in the 49-robot grid: the error associated to each robot (i.e. to each subgraph corresponding to a robot trajectory) is not monotonically decreasing and the error for some robot can increase to bring down the overall error. Figure 14c to Figure 14d reports the change in the rotation and pose estimate for individual robots. Estimate changes become negligible within a 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 η_p).

Figure 15 shows the estimated trajectory after 10 and 1000 iterations of the DGS algorithm for the 49-robot grid.



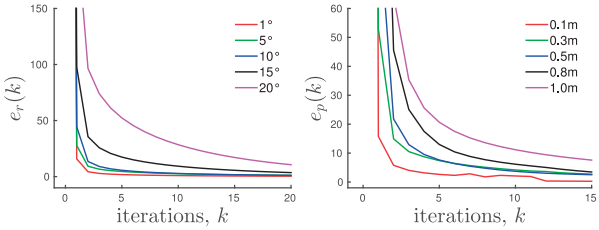
(a) Initial (b) 10 iterations (c) 1000 iterations

Fig. 15. 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.



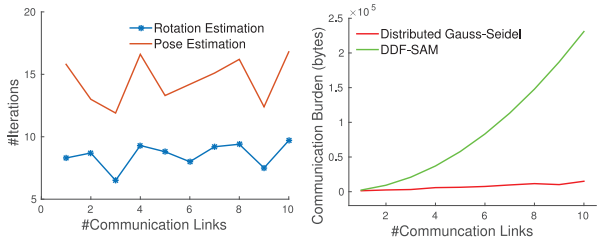
(a) Rotation error (b) Pose error

Fig. 16. DGS: convergence for scenarios with an increasing number of robots.



(a) Rotation noise (b) Translation noise

Fig. 17. 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.



(a) (b)

Fig. 18. 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 an increasing number of separators.

The odometric estimate (Figure 15a) 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. Figure 16 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 numbers of robots, the convergence rate becomes slightly slower, while in all cases the errors are negligible for a 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 Carlone et al. (2015b) and against a standard (centralized) Gauss–Newton (GN) method, available in gtsam (Dellaert (2012)). We use the cost attained in problem (3) by each technique as an accuracy metric (the lower the better). Table 1 reports the number of iterations and the cost attained in problem (3), for the compared techniques. The number of iterations is the sum of the number of iterations required to solve (7) and (12). The cost of the DGS approach is given for two choices of the thresholds η_r and η_p. As already reported in Carlone et al. (2015b), 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. Figure 17 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 a 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 Carlone et al. (2015b) and the Gauss–Newton method. The DGS approach is able to replicate the accuracy of the centralized two-stage approach, regardless of the noise level, while the choice of thresholds $\eta_r = \eta_p = 10^{-1}$ ensures accurate estimation within a 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. Figure 18a

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 ten Monte Carlo runs.

#Robots	Distributed Gauss–Seidel						Centralized	
	$\eta_r = \eta_p = 10^{-1}$			$\eta_r = \eta_p = 10^{-2}$			Two-stage	GN
	#Iter	Cost	% Diff.w/ GN	#Iter	Cost	% Diff.w/ GN	cost	cost
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

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

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.

Figure 18b compares the information exchanged in the DJ algorithm against a state-of-the-art algorithm, DDF-SAM (Cunningham et al. (2010)). In DDF-SAM, each robot sends $K_{GN}\left[s\,B_p + \left(s\,B_p\right)^2\right]$ 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 robot sends $K^r_{\text{DGS}}\,(s\,B_r) + K^p_{\text{DGS}}\left(s\,B_p\right)$ bytes, where K^r_{DGS} and K^p_{DGS} are the number of iterations required by the DGS algorithm to solve the linear systems (7) and (12), respectively, and B_r is the size of a rotation (in bytes). We assume $B_r = 9$ doubles (72 bytes)[4] and $B_p = 6$ doubles (48 bytes). The number of iterations K^r_{DGS} and K^p_{DGS} are plotted in Figure 18a. From Figure 18b 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 in Gazebo simulations as shown in Figure 19. 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 (0.5 m in our tests). Results are averaged over 100 Monte Carlo runs.

Figure 19 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. Figure 20a 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. Figure 20b reports trajectory samples for different robots in our Monte Carlo analysis. Figures 21 and 22 show the average rotation and pose errors for increasing levels of noise in the Gazebo simulations with two robots and four robots.

5.2. Simulation results: Multi-robot object-based SLAM

In this section, we characterize the performance of the DGS algorithms associated with our object-based model described in Section 4. 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

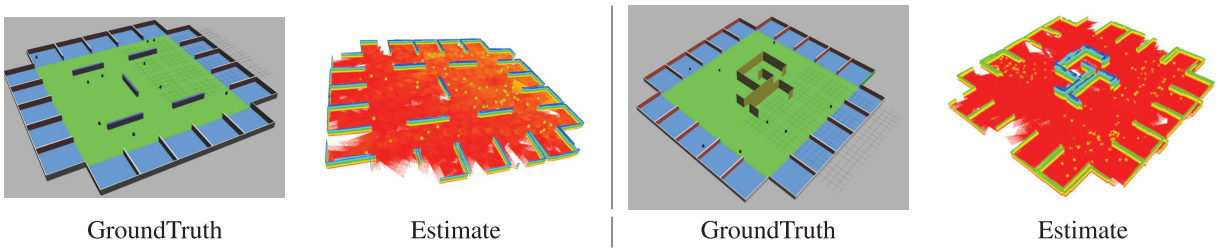
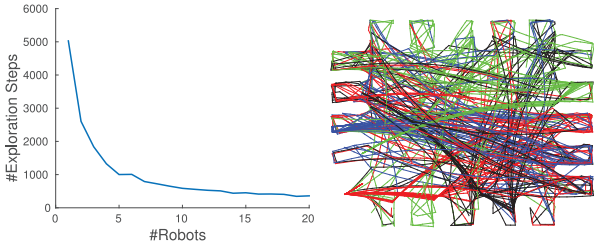
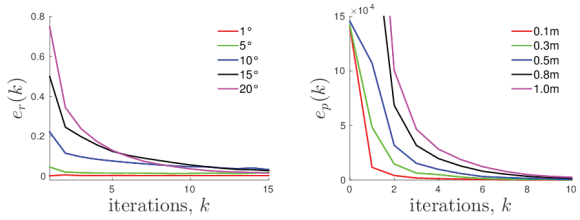


Fig. 19. Gazebo tests: ground truth environments and aggregated point clouds corresponding to the DGS estimate.



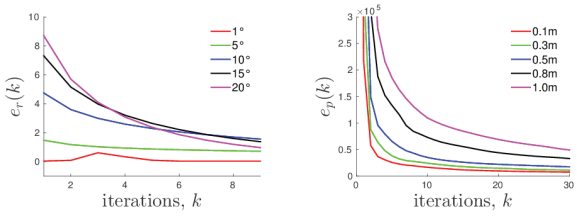
(a) Exploration steps (b) Monte Carlo runs

Fig. 20. (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.



(a) Rotation noise (b) Translation noise

Fig. 21. Convergence for increasing levels of noise (scenario with two 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.



(a) Rotation noise (b) Translation noise

Fig. 22. Convergence for increasing levels of noise (scenario with four 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.

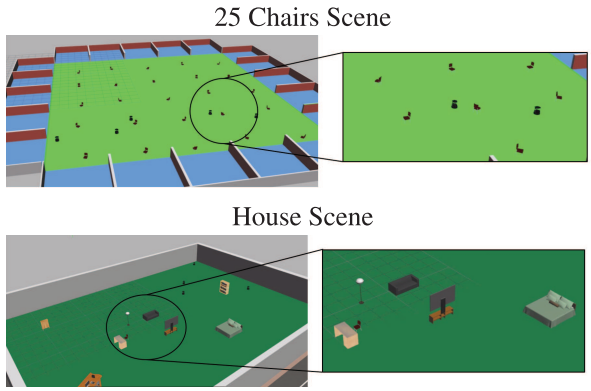


Fig. 23. Multi-robot object-based SLAM in Gazebo: the `25 Chairs` and `House` scenarios simulated in Gazebo.

the `House` scenario, we placed furniture as objects in order to simulate a living room environment. Figure 23 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. (2012), 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

$$ATE^* = \left(\frac{1}{\sum_{\alpha \in \mathbf{\Omega}} n_\alpha} \sum_{\alpha \in \mathbf{\Omega}} \sum_{i=1}^{n_\alpha} \| \boldsymbol{t}_{\alpha_i} - \boldsymbol{t}^*_{\alpha_i} \|^2 \right)^{\frac{1}{2}} \tag{25}$$

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. 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

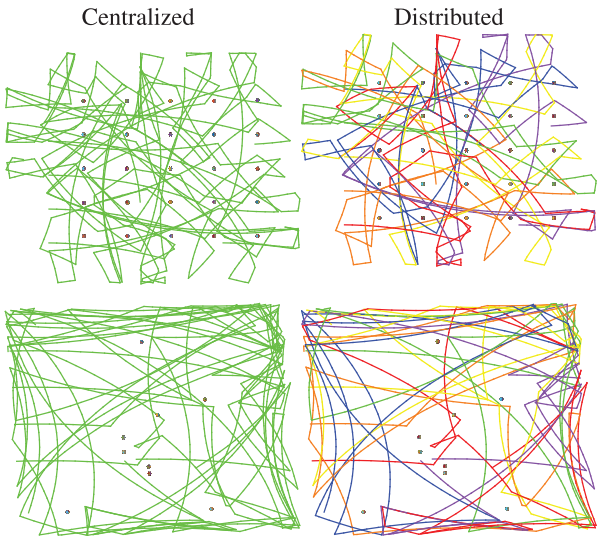


Fig. 24. Trajectories of the six robots and object locations (shows as red dots) estimated using the centralized GN method and the proposed DGS method for the `25 Chairs` (top) and `House` scenarios (bottom).

the proposed approach versus a centralized GN method

$$ARE^* = \left(\frac{1}{\sum_{\alpha \in \mathbf{\Omega}} n_\alpha} \sum_{\alpha \in \mathbf{\Omega}} \sum_{i=1}^{n_\alpha} \| \mathrm{Log} \left((\boldsymbol{R}^*_{\alpha_i})^\mathsf{T} \boldsymbol{R}_{\alpha_i} \right) \|^2 \right)^{\frac{1}{2}} \quad (26)$$

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. Figure 24 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/nYm2sSHuGjo and https://youtu.be/nXJamypPvVY.

Table 3 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 as accurate as the centralized Gauss–Newton method and the number of iterations does not increase with increasing the number of robots, making our approach scalable to large teams. Usually, a 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 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 4 shows that our approach is able to replicate the accuracy of the centralized Gauss–Newton method, regardless of the noise level.

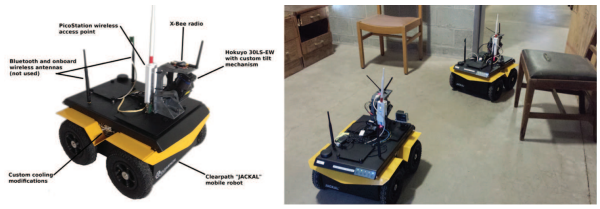


Fig. 25. (Left) Clearpath Jackal robot used for the field tests: platform and sensor layout; (right) snapshot of the test facility with the two Jackal robots.

Fig. 26. Clearpath Jackal robot moving on gravel.

5.3. *Field experiments: Multi-robot pose graph optimization*

We tested the DGS approach on field data collected by two to four Jackal robots (Figure 25), moving in a military training facility. Each robot collects 3D scans using Velodyne 32E, and uses IMU and wheel odometry to measure its ego-motion[5]. 3D scans are used to compute inter-robot measurements (via ICP Trevor et al. (2014)) during rendezvous. We evaluated our approach in multiple buildings in the military training facility.

Figures 27 to 29 show the aggregated 3D point clouds (left), the estimates trajectories (center), and the aggregated occupancy grid map (right) over multiple runs. 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 Figure 27 corresponds to experiments with 4 robots moving in indoor environment, while Figure 28 corresponds to tests performed in a mixed indoor-outdoor scenario (with robots moving on gravel when outdoor, Figure 26). The four tests of Figure 29 correspond to early results with 2 robots for which we do not have a comparison against DDF-SAM.

Quantitative results are given in Table 5, 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. 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

Table 3. 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 ten Monte Carlo runs.

#Robots	Distributed Gauss–Seidel				Centralized	ATE* (m)		ARE* (deg)	
	$\eta = 10^{-1}$		$\eta = 10^{-2}$		GN	Poses	Lmrks.	Poses	Lmrks.
	#Iter	Cost	#Iter	Cost	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

Table 4. 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 six robots. ATE* and ARE* are measured using $\eta = 10^{-1}$ as stopping condition.

Measurement noise		Distributed Gauss–Seidel				Centralized	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

to tests performed in a mixed indoor-outdoor scenario (Figure 28). 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 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 linearlization errors resulting in higher cost. Figure 30 shows the corresponding histogram visualization comparing the cost attained by the DGS algorithm and centralized Two-Stage and GN algorithm.

5.4. Field experiments: Multi-robot object-based SLAM

We test the combination of the DGS method and our object-based model on field data collected by two Jackal robots (Figure 32) moving in a military training facility. We scattered the environment with a set of objects (Figure 31) from the BigBird dataset (Singh et al. (2014)). Each robot is equipped with an Asus Xtion RGB-D sensor and uses wheel odometry to measure its ego-motion. We use the RGB-D sensor for object detection and object pose estimation.

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. The `stadium` datasets were collected in an indoor basketball stadium with the robot trajectories bounded in a roughly rectangular area. The `house` dataset was collected in 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. Similarly to the standard GN method, our current distributed optimization technique (DGS) is not robust to outliers (more comments in Section 6). 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 the 12 objects present in the environment. Similarly, 5 objects were detected in `stadium-2` and `house`. Figure 33 shows a snapshot of the bounding box of the detected object in three different scenes. Videos showing YOLO object detection results on other datasets are available at https://youtu.be/urZiIJK2IYk and https://youtu.be/-F6JpVmOrc0.

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) in our real tests.

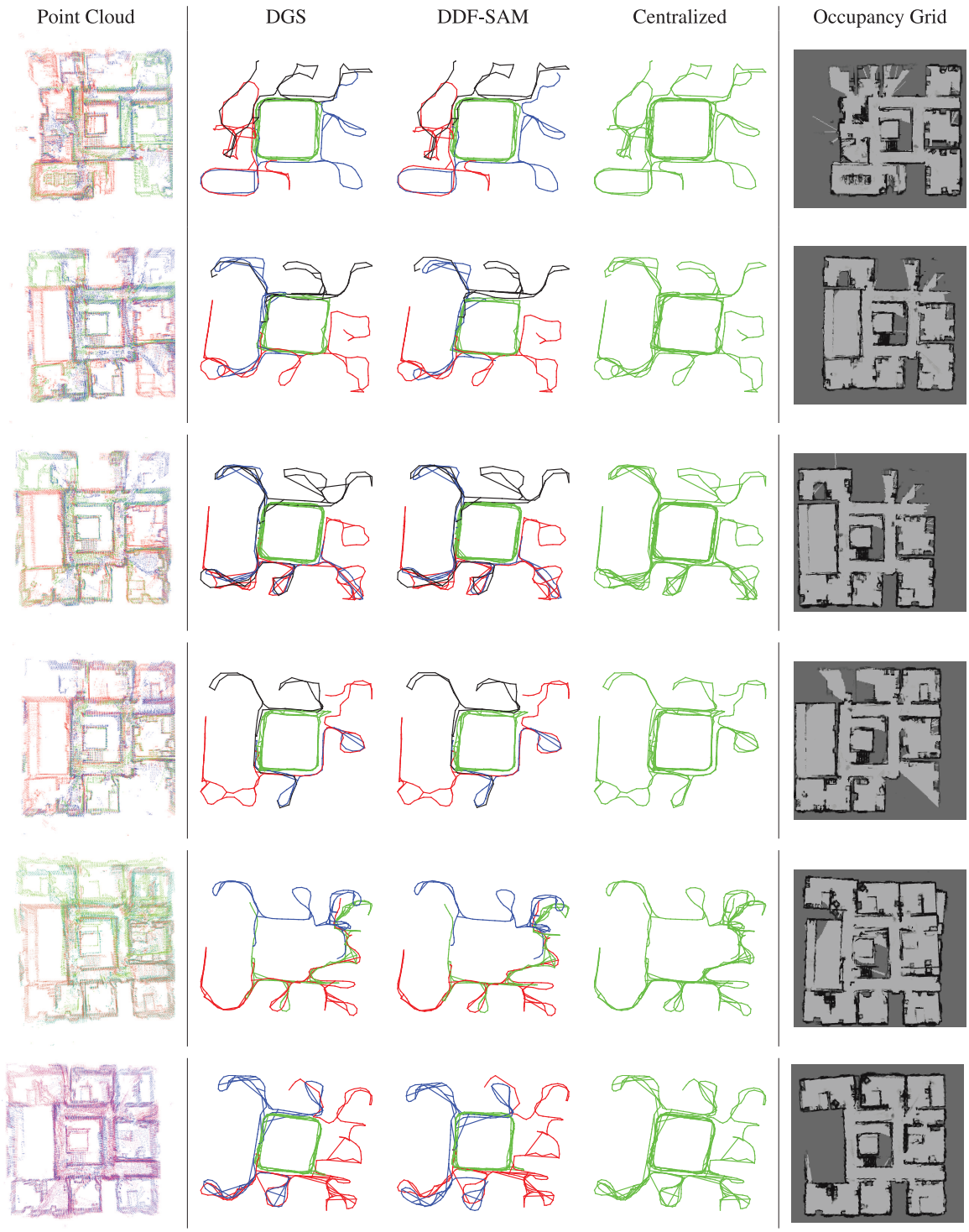


Fig. 27. Indoor scenarios: (Left) aggregated point cloud obtained from the DGS trajectory estimate. (Center) estimated trajectories for DGS, GN and DDF-SAM (robots shown in red, blue, green, and black for the distributed techniques). (Right) overall occupancy grid map obtained from the DGS trajectory estimate.

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.

Fig. 28. 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.

Communication bandwidth requirements. Table 6 also compares the average communication requirements in the case of transmission of dense point clouds and object-based models. When using point clouds, the robots are required to send at least one RGB-D 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,

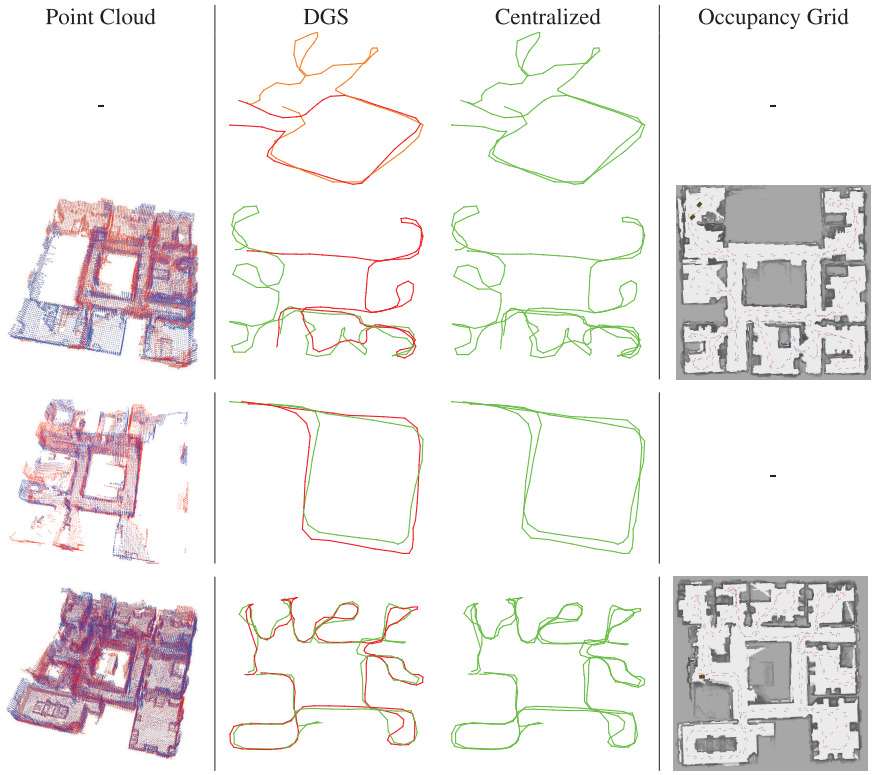


Fig. 29. Early tests with two robots: (Left) aggregated point cloud obtained from the DGS trajectory estimate. (Center) estimated trajectories for DGS and GN. (Right) overall occupancy grid map obtained from the DGS trajectory estimate.

Table 5. 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	#vertices	#edges	#links	Distributed Gauss–Seidel								Centralized		DDF-SAM
				$\eta_r = \eta_p = 10^{-1}$				$\eta_r = \eta_p = 10^{-2}$				Two-stage	GN	
				#Iter	Cost	ATE*	ARE*	#Iter	Cost	ATE*	ARE*	Cost	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	-

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.

Accuracy. Figure 34 shows the trajectories of the two robots in three runs and the object pose estimates. The figure compares our approach and the corresponding centralized GN estimate. Quantitative results are given in Table 7, which reports the cost attained by our approach,

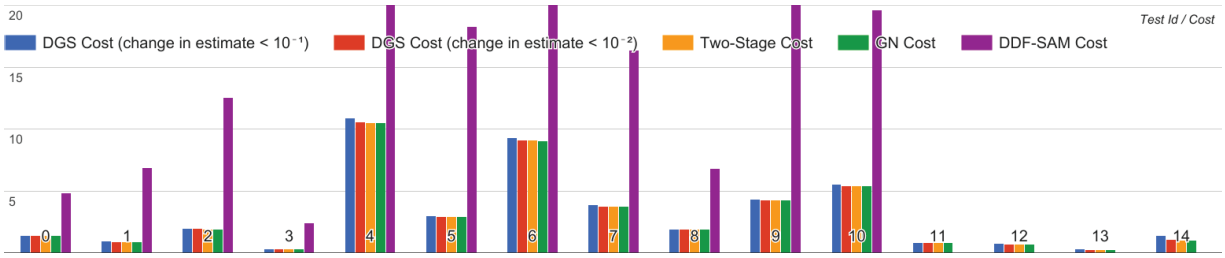


Fig. 30. Histogram visualization comparing the cost attained by DGS algorithm on field data as compared to the centralized Two-Stage, GN and DDF-SAM method. It shows that all the proposed approaches are close to GN, while DDF-SAM has worse performance. The length of y-axis (cost) is limited to 20 for visualization purposes. Additional quantitative results are given in Table 5.

Fig. 31. Objects from the BigBird dataset used in the field experiments of Section 5.4.

Fig. 32. Snapshot of the test facility, the two Clearpath Jackal robots, and the objects used for object-based SLAM for the tests of Section 5.4.

Fig. 33. Snapshot of the YOLO object detection in two difference scenes: (left) `stadium` dataset, (right) `house` dataset.

the number of iterations, ATE*, ARE* as compared to the centralized GN approach. The table confirms that our distributed approach converges in few iterations and is practically as accurate as the centralized GN method; in particular the mismatch between the DGS and the GN estimates is in the order of millimeters for the position estimates and tenth of degrees for the rotation estimates. Note that for these indoor experiments the wheel odometry is fairly accurate, since the robot moves on wooden or tiled floor. This results in better performance for the proposed technique and in small costs in GN. The initial cost, instead, is large mostly because of the error in the initial alignment between the two robots.

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

6. Conclusions and future work

We investigate distributed algorithms to estimate the 3D trajectories of multiple cooperative robots from relative pose measurements. Our first 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.

Our second contribution is to extend the DGS approach to use objects as landmarks for multi-robot mapping. We show that using object-based abstractions in a multi-robot 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.

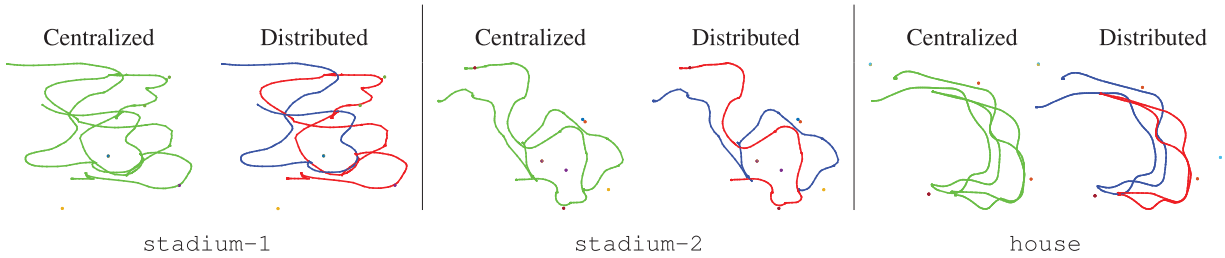


Fig. 34. Real tests: estimated trajectories and object poses for our approach and for the centralized GN method. Trajectories of the two robots are shown as red and blue lines, while objects are shown as colored dots.

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

We are currently extending the approach proposed in this paper in several directions. First, our current approach for object-based mapping assumes that a model of each observed object is known in advance. However it can be challenging to store a large number of object models, and to account for intra-class variations. As a future work, we plan to extend our approach to the case where object models are not previously known (at an instance level) and instead object shapes are jointly optimized within our SLAM framework.

Second, 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.

Third, 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.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. They are also grateful to Jason Gregory and David Baran for their help with the data collection in a military training facility.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was partially supported by the ARL MAST CTA Project 1436607.

Notes

1. When γ or the relaxation factor is 1, the Distributed Successive Over-Relaxation (SOR) method is known as the Distributed Gauss–Seidel (DGS) method.
2. In order to simplify notations, $\forall i$ refers to the list of time indices for each robot.
3. 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. In the linear system (7) we relax the orthogonality constraints hence we cannot parametrize the rotations with a minimal 3-parameter representation.
5. https://github.com/CognitiveRobotics/omnimapper

References

Anderson B, Shames I, Mao G, et al. (2010) Formal theory of noisy sensor network localization. *SIAM Journal on Discrete Mathematics* 24(2): 684–698.

Andersson L and Nygards J (2008) C-SAM: Multi-robot SLAM using square root information smoothing. In: *IEEE international conference on robotics and automation (ICRA)*. Pasadena, CA, May 2010, pp. 2798–2805.

Aragues R, Carlone L, Calafiore G, et al. (2011) Multi-agent localization from noisy relative pose measurements. In: *IEEE international conference on robotics and automation (ICRA)*, Shanghai, China, 9–13 May 2011, pp. 364–369.

Aragues R, Carlone L, Calafiore G, et al. (2012a) Distributed centroid estimation from noisy relative measurements. *Systems & Control Letters* 61(7): 773–779.

Aragues R, Cortes J and Sagues C (2012b) Distributed consensus on robot networks for dynamically merging feature-based maps. *IEEE Transactions on Robotics*.

Bahr A, Walter M and Leonard J (2009) Consistent cooperative localization. In: *IEEE international conference on robotics and automation (ICRA)*, Kobe, Japan, 12–17 May 2009, pp. 3415–3422.

Bailey T, Bryson M, Mu H, et al. (2011) Decentralised cooperative localisation for heterogeneous teams of mobile robots. In: *IEEE international conference on robotics and automation (ICRA)*. Shanghai, China, 9–13 May 2011, pp. 2859–2865.

Bao SYZ, Bagra M, Chao YW, et al. (2012) Semantic structure from motion with points, regions, and objects. In: *Conference on computer vision and pattern recognition*, Providence, RI, USA, pp. 2703–2710.

Barooah P and Hespanha J (2005) Semantic structure from motion. In: *International conference on intelligent sensing and information processing*, pp. 226–231.

Barooah P and Hespanha J (2007) Estimation on graphs from relative measurements. *Control System Magazine* 27(4): 57–74.

Bertsekas D and Tsitsiklis J (1989) *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs: Prentice–Hall.

Calafiore G, Carlone L and Wei M (2012) A distributed technique for localization of agent formations from relative range measurements. *IEEE Transactions on Systems, Man, and Cybernetics - Part A* 42(5): 1083–4427.

Carlone L and Dellaert F (2015) Duality-based verification techniques for 2D SLAM. In: *IEEE international conference on robotics and automation (ICRA)*, Seattle, WA, USA, pp. 4589–4596.

Carlone L, Ng MK, Du J, et al. (2011) Simultaneous localization and mapping using Rao–Blackwellized particle filters in multi robot systems. *Journal of Intelligent and Robotic Systems* 63(2): 283–307.

Carlone L, Rosen D, Calafiore G, et al. (2015a) Lagrangian duality in 3D SLAM: Verification techniques and optimal solutions. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)*. Hamburg, Germany, pp. 125–132.

Carlone L, Tron R, Daniilidis K, et al. (2015b) Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization. In: *IEEE international conference on robotics and automation (ICRA)*, Seattle, WA, USA, pp. 4597–4604.

Carron A, Todescato M, Carli R, et al. (2014) An asynchronous consensus-based algorithm for estimation from noisy relative measurements. *IEEE Transactions on Control of Network Systems* 1(3): 2325–5870.

Choudhary S, Trevor AJB, Christensen HI, et al. (2014) SLAM with object discovery, modeling and mapping. In: *2014 IEEE/RSJ international conference on intelligent robots and systems*, Chicago, USA, 14–18 September 2014, pp. 1018–1025.

Civera J, Gálvez-López D, Riazuelo L, et al. (2011) Towards semantic SLAM using a monocular camera. In: *2011 IEEE/RSJ international conference on intelligent robots and systems, IROS*, San Francisco, USA, 25–30 September 2011, pp. 1277–1284.

Cunningham A, Indelman V and Dellaert F (2013) DDF-SAM 2.0: Consistent distributed smoothing and mapping. In: *IEEE international conference on robotics and automation (ICRA)*, Karlsruhe, Germany., pp. 5220–5227.

Cunningham A, Paluri M and Dellaert F (2010) DDF-SAM: Fully distributed SLAM using constrained factor graphs. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)*. Taipei, Taiwan, pp. 3025–3030.

Davison AJ, Reid ID, Molton ND, et al. (2007) Monoslam: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(6): 1052–1067.

Dellaert F (2005) Square root SAM: Simultaneous location and mapping via square root information smoothing. In: *Robotics: Science and systems (RSS)*. Cambridge, Massachusetts, 8–11 June 2005,

Dellaert F (2012) Factor graphs and GTSAM: A hands-on introduction. Technical Report GT-RIM-CP&R-2012-002, Georgia Institute of Technology, USA.

Dong J, Nelson E, Indelman V, et al. (2015) Distributed real-time cooperative localization and mapping using an uncertainty-aware expectation maximization approach. In: *IEEE international conference on robotics and automation (ICRA)*. Seattle, WA, USA, pp. 5807–5814.

Estrada C, Neira J and Tardos J (2005) Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics* 21(4): 588–596.

Finman R, Whelan T, Kaess M, et al. (2013) Toward lifelong object segmentation from change detection in dense RGB-D maps. In: *2013 European conference on mobile robots*, Barcelona, Spain, 25–27 September 2013, pp. 178–185.

Franceschelli M and Gasparri A (2010) On agreement problems with Gossip algorithms in absence of common reference frames. In: *IEEE international conference on robotics and automation (ICRA)*, Anchorage, AK, USA, volume 337, pp. 4481–4486.

Freris N and Zouzias A (2015) Fast distributed smoothing of relative measurements. In: *IEEE conference on decision and control*, Maui, HI, USA, pp. 1411–1416.

Frese U (2006) Treemap: An $O(\log n)$ algorithm for indoor simultaneous localization and mapping. *Autonomous Robots* 21(2): 103–122.

Frese U, Larsson P and Duckett T (2005) A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics* 21(2): 196–207.

Gálvez-López D, Salas M, Tardós JD, et al. (2016) Real-time monocular object SLAM. *Robotics and Autonomous Systems* 75: 435–449.

Grisetti G, Kuemmerle R, Stachniss C, et al. (2010) Hierarchical optimization on manifolds for online 2D and 3D mapping. In: *IEEE international conference on robotics and automation (ICRA)*, Anchorage, Alaska. pp. 273–278.

Grisetti G, Kümmerle R and Ni K (2012) Robust optimization of factor graphs by using condensed measurements. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)*. Vilamoura, Portugal, pp. 581–588.

Hartley R, Trumpf J, Dai Y, et al. (2013) Rotation averaging. *International Journal of Computer Vision* 103(3): 267–305.

Hatanaka T, Fujita M and Bullo F (2010) Vision-based cooperative estimation via multi-agent optimization. In: *IEEE conference on decision and control*. Atlanta, GA, USA, pp. 2492–2497.

Howard A (2006) Multi-robot simultaneous localization and mapping using particle filters. *International Journal of Robotics Research* 25(12): 1243–1256.

Indelman V, Gurfil P, Rivlin E, et al. (2012) Graph-based distributed cooperative navigation for a general multi-robot measurement model. *International Journal of Robotics Research* 31(9): 1057–1080.

Indelman V, Nelson E, Michael N, et al. (2014) Multi-robot pose graph localization and data association from unknown initial relative poses via expectation maximization. In:

IEEE international conference on robotics and automation (ICRA). Hong Kong, China, pp. 593–600.

Kim B, Kaess M, Fletcher L, et al. (2010) Multiple relative pose graphs for robust cooperative mapping. In: *IEEE international conference on robotics and automation (ICRA)*, Anchorage, Alaska, pp. 3185–3192.

Kim YM, Mitra NJ, Yan DM, et al. (2012) Acquiring 3D indoor environments with variability and repetition. *ACM Transactions on Graphics* 31(6): 138.

Knuth J and Barooah P (2013) Collaborative localization with heterogeneous inter-robot measurements by Riemannian optimization. In: *IEEE international conference on robotics and automation (ICRA)*, Karlsruhe, Germany, pp. 1534–1539.

Koppula H, Anand A, Joachims T, et al. (2011) Semantic labeling of 3D point clouds for indoor scenes. In: *Advances in neural information processing systems (NIPS)*.Granada, Spain, pp. 244–252.

Kuipers B (2000) The spatial semantic hierarchy. *Artificial Intelligence* 119: 191–233.

Kundu A, Li Y, Dellaert F, et al. (2014) *Joint Semantic Segmentation and 3D Reconstruction from Monocular Video*. Cham: Springer International Publishing.

Lazaro M, Paz L, Pinies P, et al. (2011) Multi-robot SLAM using condensed measurements. In: *IEEE international conference on robotics and automation (ICRA)*, Tokyo, Japan, pp. 1069–1076.

Leonard J and Feder H (2001) Decoupled stochastic mapping. *IEEE Journal of Oceanic Engineering* 26(4): 561–571.

Leonard J and Newman P (2003) Consistent, convergent, and constant-time SLAM. In: *International joint conference on AI (IJCAI)*. Acapulco, Mexico, pp. 1143–1150.

Martinec D and Pajdla T (2007) Robust rotation and translation estimation in multiview reconstruction. In: *IEEE conference on computer vision and pattern recognition (CVPR)*, Minneapolis, MN, USA, pp. 1–8.

McCormac J, Handa A, Davison AJ, et al. (2016) Semanticfusion: Dense 3D semantic mapping with convolutional neural networks. *CoRR*. arXiv:1609.05130 [cs.CV].

Nerurkar E, Roumeliotis S and Martinelli A (2009) Distributed maximum a posteriori estimation for multi-robot cooperative localization. In: *IEEE international conference on robotics and automation (ICRA)*, Kobe, Japan, pp. 1402–1409.

Ni K and Dellaert F (2010) Multi-level submap based SLAM using nested dissection. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)*. Taipei, Taiwan, pp. 2558–2565.

Ni K, Steedly D and Dellaert F (2007) Tectonic SAM: Exact; out-of-core; submap-based SLAM. In: *IEEE international conference on robotics and automation (ICRA)*, Rome, Italy. pp. 1678–1685.

Nieto–Granda C, Rogers III JG, Trevor AJB, et al. (2010) Semantic map partitioning in indoor environments using regional analysis. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Taipei, Taiwan, pp. 1451–1456.

Nüchter A and Hertzberg J (2008) Towards semantic maps for mobile robots. *Robotics and Autonomous Systems* 56(11): 915–926.

Olfati–Saber R (2006) Swarms on sphere: A programmable swarm with synchronous behaviors like oscillator networks. In: *IEEE conference on decision and control*, San Diego, CA, USA, pp. 5060–5066.

Paull L, Huang G, Seto M, et al. (2015) Communication-constrained multi-AUV cooperative SLAM. In: *IEEE international conference on robotics and automation (ICRA)*. Seattle, WA, USA, pp. 509–516.

Pillai S and Leonard J (2015) Monocular SLAM supported object recognition. In: *Proceedings of robotics: Science and systems (RSS)*, Rome, Italy. arXiv:1506.01732 [cs.RO].

Piovan G, Shames I, Fidan B, et al. (2013) On frame and orientation localization for relative sensing networks. *Automatica* 49(1): 206–213.

Pronobis A and Jensfelt P (2012) Large-scale semantic mapping and reasoning with heterogeneous modalities. In: *IEEE international conference on robotics and automation (ICRA)*, Saint Paul, MN, USA.

Ranganathan A and Dellaert F (2007) Semantic modeling of places using objects. In: *Robotics: Science and systems (RSS)*, Atlanta, USA, June. DOI: 10.15607/RSS.2007.III.001.

Redmon J, Divvala SK, Girshick RB, et al. (2015) You only look once: Unified, real-time object detection. *CoRR*. arXiv:1506.02640 [cs.CV].

Rogers JG, Trevor AJB, Nieto–Granda C, et al. (2011) Simultaneous localization and mapping with learned object recognition and semantic data association. In: *2011 IEEE/RSJ international conference on intelligent robots and systems*, San Francisco, CA, USA, pp. 1264–1270.

Rosen D, Carlone L, Bandeira A, et al. (2016) SE-Sync: A certifiably correct algorithm for synchronization over the special euclidean group. arXiv:1612.07386 [cs.RO]

Roumeliotis S and Bekey G (2002) Distributed multi-robot localization. *IEEE Transactions on Robotics and Automation*. pp. 781–795.

Russell W, Klein D and Hespanha J (2011) Optimal estimation on the graph cycle space. *IEEE Transactions on Signal Processing* 59(6): 2834–2846.

Rusu RB (2009) *Semantic 3D object maps for everyday manipulation in human living environments*. PhD Thesis, Technische Universität München, Germany.

Rusu RB, Marton ZC, Blodow N, et al. (2008) Functional object mapping of kitchen environments. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Nice, France, pp. 3525–3532.

Salas–Moreno RF, Newcombe RA, Strasdat H, et al. (2013) SLAM++: Simultaneous localisation and mapping at the level of objects. In: *IEEE conference on computer vision and pattern recognition (CVPR)*. Portland, OR, USA, pp. 1352–1359.

Sarlette A and Sepulchre R (2009) Consensus optimization on manifolds. *SIAM Journal on Control and Optimization* 48(1): 56–76.

Segal A, Haehnel D and Thrun S (2009) Generalized-ICP. In: *Proceedings of robotics: Science and systems*, Seattle, USA.

Simonetto A and Leus G (2014) Distributed maximum likelihood sensor network localization. *IEEE Transactions on Signal Processing* 52(6): 1424–1437.

Singh A, Sha J, Narayan KS, et al. (2014) Bigbird: A large-scale 3D database of object instances. In: *2014 IEEE international conference on robotics and automation (ICRA)*, Hong Kong, China, pp. 509–516.

Sturm J, Engelhard N, Endres F, et al. (2012) A benchmark for the evaluation of RGB-D SLAM systems. In: *Proceedings of the international conference on intelligent robot systems (IROS)*. Vilamoura, Portugal.

Suger B, Tipaldi G, Spinello L, et al. (2014) An approach to solving large-scale SLAM problems with a small memory footprint. In: *IEEE international conference on robotics and automation (ICRA)*. Hong Kong, China, pp. 3632–3637.

Thrun S and Liu Y (2003) Multi-robot SLAM with sparse extended information filters. In: *Proceedings of the 11th international symposium of robotics research (ISRR'03)*, Sienna, Italy. Springer. pp. 254–266.

Thunberg J, Montijano E and Hu X (2011) Distributed attitude synchronization control. In: *IEEE conference on decision and control*. Beijing, China, pp. 958–963.

Todescato M, Carron A, Carli R, et al. (2015) Distributed localization from relative noisy measurements: A robust gradient based approach. In: *European control conference*. Linz, Austria, pp. 914–919.

Trevor A, Rogers J and Christensen H (2014) Omnimapper: A modular multimodal mapping framework. In: *IEEE international conference on robotics and automation (ICRA)*. Hong Kong, China, pp. 1983–1900.

Trevor A, Gedikli S, Rusu RB, et al. (2013) Efficient organized point cloud segmentation with connected components. In: Proceedings of semantic perception mapping and exploration, pp. 1–6.

Trevor AJB, Rogers III JG and Christensen HI (2012) Planar surface SLAM with 3D and 2D sensors. In: *IEEE international conference on robotics and automation (ICRA)*, St Paul, USA. IEEE. pp. 3041–3048.

Tron R, Afsari B and Vidal R (2012a) Intrinsic consensus on SO(3) with almost global convergence. In: *IEEE conference on decision and control*. Maui, HI, USA, pp. 2052–2058.

Tron R, Afsari B and Vidal R (2012b) Riemannian consensus for manifolds with bounded curvature. *IEEE Transactions on Automatic Control*. pp. 921–934.

Tron R and Vidal R (2009) Distributed image-based 3-D localization in camera networks. In: *IEEE conference on decision and control*. Shanghai, China, pp. 901–908.

Valentin J, Vineet V, Cheng MM, et al. (2015) Semanticpaint: Interactive 3D labeling and learning at your fingertips. *ACM Transactions on Graphics* 34(5): 154:1–154:17.

Vineet V, Miksik O, Lidegaard M, et al. (2015) Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In: *2015 IEEE international conference on robotics and automation (ICRA)*, Seattle, WA, USA, pp. 75–82.

Wei M, Aragues R, Sagues C, et al. (2015) Noisy range network localization based on distributed multidimensional scaling. *IEEE Sensors Journals* 15(3): 854–874.

Zhao L, Huang S and Dissanayake G (2013) Linear SLAM: A linear solution to the feature-based and pose graph SLAM based on submap joining. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)*. Tokyo, Japan, pp. 24–30.

Zhou X and Roumeliotis S (2006) Multi-robot SLAM with unknown initial correspondence: The robot rendezvous case. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 1785–1792. Beijing, China: IEEE.