Distributed Trajectory Estimation with Privacy and Communication Constraints: a Two-Stage Distributed Gauss-Seidel Approach

Siddharth Choudhary, Luca Carlone, Carlos Nieto, John Rogers, Henrik I. Christensen, Frank Dellaert

***Abstract*— We propose a distributed algorithm to estimate the 3D trajectories of multiple cooperative robots from relative pose measurements. Our approach leverages recent results [1] which show that the maximum likelihood trajectory is well approximated by a sequence of two quadratic subproblems. The main contribution of the present work is to show that these subproblems can be solved in a distributed manner, using the *distributed Gauss-Seidel* (DGS) algorithm. Our approach has several advantages. It requires minimal information exchange, which is beneficial in presence of communication and privacy constraints. It has an anytime flavor: after few iterations the trajectory estimates are already accurate, and they asymptotically convergence to the centralized estimate. The DGS approach scales well to large teams, and it has a straightforward implementation. We test the approach in simulations and field tests, demonstrating its advantages over related techniques.**

I. Introduction

The use of multiple cooperative robots has the potential to enable fast information gathering, and more efficient coverage and monitoring of large areas. For military applications, multi robot systems promise more efficient operation and improved robustness to adversarial attacks. In civil applications (e.g., pollution monitoring, surveillance, search and rescue), the use of several inexpensive, heterogeneous, agile platforms is an appealing alternative to monolithic single robot systems.

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 explores an unknown environment and each robot has to estimate its trajectory from its own sensor data and leveraging information exchanged with the teammates. Trajectory estimation constitutes the backbone for many estimation and control tasks (e.g., geo-tagging sensor data, 3D map reconstruction, position-aware task allocation). Indeed, in our application, trajectory estimation enables distributed 3D reconstruction and localization (Fig. 1).

We consider a realistic scenario, in which the robots 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. We aim at designing a technique that allows each robot to estimate its own trajectory, while asking for minimal knowledge of the trajectory of the teammates. This "privacy constraint" has a clear motivation in a military application: in case one robot is captured, it cannot provide sensitive information about the areas covered by the other robots. 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.

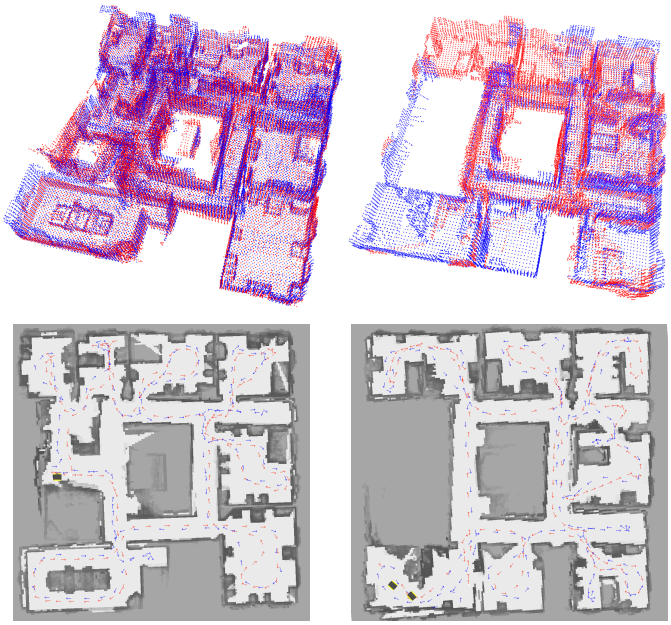


Fig. 1. In our field experiments, distributed trajectory estimation enables 3D reconstruction of an entire building using two robots (red, blue). Each column of the figure shows the reconstructed point cloud of a floor (top), and the estimated trajectories overlaid on an occupancy grid map (bottom).

Related work in robotics. Distributed estimation in multi robot systems is an active field of research, with special attention being paid to communication constraints [2], heterogeneity [3], [4], consistency [5], and robust data association [6]. Robotic literature offers distributed implementations of different estimation techniques, including Extended Kalman filters [7], information filters [8], and particle filters [9], [10]. More recently, the community reached a large consensus on the use of maximum likelihood (ML) estimation, which, applied to trajectory estimation, is often referred to as *pose graph optimization*. 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

S. Choudhary, C. Nieto, H.I. Christensen, and F. Dellaert are with the College of Computing, Georgia Institute of Technology, USA, {siddharth.choudhary, carlos.nieto}@gatech.edu, {hic, dellaert}@cc.gatech.edu

L. Carlone is with the Laboratory for Information & Decision Systems, Massachusetts Institute of Technology, USA, lcarlone@mit.edu

J. Rogers is with U.S. Army Research Laboratory (ARL), USA, john.g.rogers59.civ@mail.mil

This work was partially funded by the ARL MAST CTA Project 1436607.

at a centralized inference engine, which performs the optimization [11], [12], [3], [13], [6].

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

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.* [14] propose an algorithm for cooperative localization based on distributed conjugate gradient. Franceschelli and Gasparri [15] propose a gossip-based algorithm for distributed pose estimation and investigate its convergence in a noiseless setup. Aragues *et al.* [16] use a distributed Jacobi approach to estimate a set of 2D poses. Knuth and Barooah [17] estimate 3D poses using distributed gradient descent. Cunningham *et al.* [18], [19] use Gaussian elimination, and develop an approach, called DDF-SAM, in which robots exchange Gaussian marginals over the *separators* (variables observed by multiple robots).

While Gaussian elimination has become a popular approach, it has two major shortcomings. First, the marginals to be exchanged among the robots are dense, hence the communication cost is quadratic in the number of separators. This motivated the use of sparsification techniques [2]. Second, Gaussian elimination is performed on a linearized version of the problem, hence these approaches require good linearization points and complex bookkeeping to ensure consistent linearization across the robots [19].

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 fields, the goal is to estimate the current state of an agent (e.g., a sensor or a camera) from relative measurements between the agents. A large body of literature deals with distributed localization from distance measurements (e.g., [20], [21]). The case of position estimation from linear measurements is considered in [22], [23], [24], [25]; the related problem of *centroid estimation* is tackled in [26]. Distributed rotation estimation has been studied in the context of attitude synchronization [27], camera network calibration [28], sensor network localization [29], and distributed consensus on manifolds [30].

Contribution. We consider a distributed ML trajectory estimation problem in which the robots have to collaboratively estimate their trajectories while minimizing the amount of exchanged information. We focus on a fully 3D case, as this setup is of great interest in many robotics applications (e.g., navigation on uneven terrain, UAVs). We also consider a fully distributed setup, in which the robots can communicate and acquire relative measurements only during rendezvous events. Our approach can be understood as a distributed implementation of the *chordal initialization* discussed in [1]. The chordal initialization [1] (recalled in Section III) 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 use a distributed Gauss-Seidel algorithm, with flagged-initialization [23]. Our approach has many practical advantages. The amount of communication required at each iteration is linear in the number of separators. It does not require linearization points. It quickly converges to the centralized estimate. It scales well to large teams. Moreover, upon convergence, it returns the same estimate of the chordal initialization [1], which has been extensively shown to be accurate and resilient to large measurement noise. We test the two-stage distributed Gauss-Seidel approach in both simulations and field tests, demonstrating that it is accurate and more parsimonious, communication-wise, than related techniques.

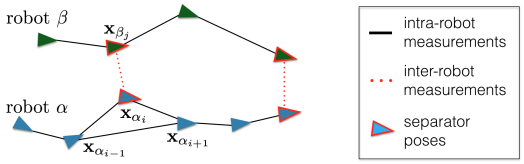


Fig. 2. An instance of multi robot trajectory estimation: two robots (α in blue, and β in dark green) traverse and unknown environment, collecting intra-robot measurements (solid black lines). During rendezvous, each robot can observe the pose of the other robot (dotted red lines). These are called inter-robot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

II. Problem Statement: Multi Robot Trajectory Estimation

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

We model each trajectory as a finite set of poses (triangles in Fig. 2); the pose assumed by robot α at time i is denoted with $\boldsymbol{x}_{\alpha_i}$ (we use Roman letters to denote time indices). We consider a 3D setup, i.e., $\boldsymbol{x}_{\alpha_i} \in \mathrm{SE}(3)$; 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 $\boldsymbol{x}_\alpha = [\boldsymbol{x}_{\alpha_1}, \boldsymbol{x}_{\alpha_2}, \ldots]$.

Measurements. We assume that each robot acquires relative pose measurements. In practice these are obtained by post-processing raw sensor data (e.g., scan matching on 3D laser scans). We consider two types of measurements: intra-robot and inter-robot measurements. The *intra-robot measurements* involve the poses of a single robot at different time instants; common examples of intra-robot measurements are odometry measurements (which constrain consecutive robot poses, e.g., $\boldsymbol{x}_{\alpha_i}$ and $\boldsymbol{x}_{\alpha_{i+1}}$ in Fig. 2) or loop closures (which constrain non-consecutive poses, e.g., $\boldsymbol{x}_{\alpha_{i-1}}$ and $\boldsymbol{x}_{\alpha_{i+1}}$ in Fig. 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 Fig. 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 $\boldsymbol{R}_{\beta_j}$ in 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 noise $\boldsymbol{t}_\epsilon$. According to our definition, intra robot measurements are in the form $\bar{\boldsymbol{z}}^{\alpha_i}_{\alpha_k}$, for some robot α and for $i \neq k$; inter-robot measurements 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 $\mathcal{E}_S$. The set of all available measurements is then $\mathcal{E} = \mathcal{E}_I \cup \mathcal{E}_S$.

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 assumed 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 [31] 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 [31] that the ML estimate $\boldsymbol{x} \doteq \{(\boldsymbol{R}_{\alpha_i}, \boldsymbol{t}_{\alpha_i}), \forall \alpha \in \Omega, \forall i\}$ can be computed as solution of the following optimization problem:

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

In (3), we use of the *chordal distance* $\|\boldsymbol{R}_{\beta_j} - \boldsymbol{R}_{\alpha_i} \bar{\boldsymbol{R}}^{\alpha_i}_{\beta_j}\|_{\mathtt{F}}$ to quantify rotation errors, while the majority of related works in robotics uses the *geodesic distance* [1].

A centralized approach to solve the multi robot PGO problem (3) works as follows. A robot collects all measurements $\mathcal{E}$. Then, the optimization problem (3) is solved using iterative optimization on manifold [32] or fast approximations [1].

In this paper we consider the more interesting case in which it is not possible to collect all measurements at a centralized estimator. This problem can be stated as follows.

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

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

Note that, while the measurements $\mathcal{E}^\alpha_I$ and $\mathcal{E}^\alpha_S$ are known by robot α, gathering the estimates from robots β requires communication, hence we want our distributed algorithm to exchange a very small portion of the trajectory estimates.

The next section presents our solution to Problem 1.

III. Two-Stage Distributed Trajectory Estimation

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 [1] (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 these two stages can be solved in distributed fashion, exploiting existing distributed linear system solvers. We propose the use of a Distributed Gauss-Seidel algorithm.

To help readability, we start with a centralized description of the approach, which is an adaptation of the chordal initialization of [1] to the multi robot case. Then we tailor the discussion to the distributed setup in Section III-B.

A. Trajectory Estimation: Centralized Description

The approach proceeds in two stages. The first stage solves for the unknown rotations of the robots by solving a relaxed problem. The second 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 subproblem:

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

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 nonconvex (due to the nonconvex constraints $\boldsymbol{R}_{\alpha_i} \in \mathrm{SO}(3)$), many algorithms to approximate its solution are available in literature. Here we use the approach proposed in [33] and reviewed in [1]. The approach

first solves the quadratic relaxation obtained by dropping the constraints $\boldsymbol{R}_{\alpha_i} \in \mathrm{SO}(3)$, and then projects the relaxed solution to $\mathrm{SO}(3)$. In formulas, the quadratic relaxation is:

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

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

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

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

Since (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 $\breve{\boldsymbol{R}}_{\alpha_i}, \forall \alpha \in \boldsymbol{\Omega}, \forall i$. 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 [33] projects them to $\mathrm{SO}(3)$, and gets the rotation estimate $\hat{\boldsymbol{R}}_{\alpha_i} = \mathtt{project}(\breve{\boldsymbol{R}}_{\alpha_i})$, $\forall \alpha \in \boldsymbol{\Omega}, \forall i$. The projection only requires to perform an SVD of $\breve{\boldsymbol{R}}_{\alpha_i}$ and can be performed independently for each rotation [1].

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

$$\begin{aligned} \min_{\substack{\boldsymbol{t}_{\alpha_i}, \boldsymbol{\theta}_{\alpha_i} \in \mathbb{R}^3 \\ \forall \alpha \in \boldsymbol{\Omega}, \forall i}} & \sum_{(\alpha_i, \beta_j) \in \mathcal{E}} \omega_t^2 \left\| \boldsymbol{t}_{\beta_j} - \boldsymbol{t}_{\alpha_i} - \hat{\boldsymbol{R}}_{\alpha_i} \mathrm{Exp}\left(\boldsymbol{\theta}_{\alpha_i}\right) \bar{\boldsymbol{t}}_{\beta_j}^{\alpha_i} \right\|^2 \\ & + \frac{\omega_R^2}{2} \left\| \hat{\boldsymbol{R}}_{\beta_j} \mathrm{Exp}\left(\boldsymbol{\theta}_{\beta_j}\right) - \hat{\boldsymbol{R}}_{\alpha_i} \mathrm{Exp}\left(\boldsymbol{\theta}_{\alpha_i}\right) \bar{\boldsymbol{R}}_{\beta_j}^{\alpha_i} \right\|_{\mathtt{F}}^2 \end{aligned} \tag{8}$$

The reparametrization allowed to drop the constraints (we are now trying to estimate vectors in $\mathbb{R}^3$), but moved the nonconvexity to the objective ($\mathrm{Exp}\left(\cdot\right)$ is nonlinear in its argument). In order to solve (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:

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

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:

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

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 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 $\boldsymbol{\theta}_{\alpha_i}$, hence 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 [1], in common problem instances (i.e., for reasonable measurement noise) it returns a solution that is very close to the ML estimate. Second, the approach only requires the solution of 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 [1]). ■

B. Distributed Trajectory Estimation

In this section we show that the two-stage approach described in Section III-A can be implemented in a decentralized fashion. Since the approach only requires solving two linear systems, every distributed linear system solver can be used as workhorse to split the computation among the robots. For instance, one could adapt the Gaussian elimination approach of [18] to solve the systems (7), (12). In this section we propose an alternative approach, based on the Distributed Gauss-Seidel (DGS) algorithm, and we discuss its advantages.

In the linear systems (7) and (12) 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 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 $\boldsymbol{H}$ and $\boldsymbol{g}$; in (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 DGS algorithm, 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 DGS algorithm [34] 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 $\alpha \in \boldsymbol{\Omega}$:

$$\boldsymbol{y}_\alpha^{(k+1)} = \boldsymbol{H}_{\alpha\alpha}^{-1}\left(-\sum_{\delta\in\boldsymbol{\Omega}_\alpha^+}\boldsymbol{H}_{\alpha\delta}\boldsymbol{y}_\delta^{(k+1)} - \sum_{\delta\in\boldsymbol{\Omega}_\alpha^-}\boldsymbol{H}_{\alpha\delta}\boldsymbol{y}_\delta^{(k)} + \boldsymbol{g}_\alpha\right) \tag{15}$$

where $\boldsymbol{\Omega}_\alpha^+$ is the set of robots that already computed the $(k+1)$-th estimate, while $\boldsymbol{\Omega}_\alpha^-$ is the set of robots that still have to perform the update (15), excluding node α (intuitively: each robot uses the latest estimate). Comparing (15) and (14), we see that if the sequence produced by the iterations (15) converges to a fixed point, then such point satisfies (14), and indeed solves the original linear system (13).

The DGS algorithm can be understood in a simple way: at each iteration, each robot estimates its own variables ($\boldsymbol{y}_\alpha^{(k+1)}$) assuming that the variables of the other robots are fixed ($\boldsymbol{y}_\delta^{(k)}$, $\boldsymbol{y}_\delta^{(k+1)}$); iterating this procedure, the robots reach an agreement on the estimates, and converge to the solution of (13). Using the DGS approach, the robots solve (7) and (12) in a distributed manner.

We now discuss two crucial aspects: the amount of communication required by the DGS and its convergence.

Remark 2 (Information Exchange in DGS): In this remark we stress that to execute the Gauss-Seidel iterations (15), 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 $\mathcal{E}_S^\alpha$. For instance, in the graph of Fig. 3 robot α only needs the estimates of $\boldsymbol{y}_{\beta_1}$ and $\boldsymbol{y}_{\beta_3}$, while does not require any knowledge about the other poses of β.

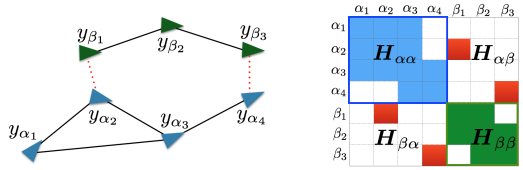


Fig. 3. Example: (left) trajectory estimation problem and (right) corresponding block structure of the matrix $\boldsymbol{H}$.

To understand this fact, we note that (7) and (12) model an estimation problem from pairwise relative measurements. It is well known that the matrix $\boldsymbol{H}$ (sometimes called the *Hessian* [35]) arising in these problems has a block structure defined by the Laplacian matrix of the underlying graph [23]. For instance, Fig. 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.

Exploiting the block sparsity of $\boldsymbol{H}$, we simplify the DGS iterations (15) as:

$$\boldsymbol{y}_\alpha^{(k+1)} = \boldsymbol{H}_{\alpha\alpha}^{-1}\left(-\sum_{(\alpha_i,\delta_j)\in\mathcal{E}_S^{\alpha+}}\boldsymbol{H}_{\alpha_i\delta_j}\boldsymbol{y}_{\delta_j}^{(k+1)} - \sum_{(\alpha_i,\delta_j)\in\mathcal{E}_S^{\alpha-}}\boldsymbol{H}_{\alpha_i\delta_j}\boldsymbol{y}_{\delta_j}^{(k)} + \boldsymbol{g}_\alpha\right) \tag{16}$$

where we removed the contributions of the zero blocks from the sum in (15); 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 robots that have not perform the iteration yet (as before: each robot simply uses its latest estimate).

Eq. (16) highlights that DGS only requires the estimates for poses involved in its inter-robot measurements $\mathcal{E}_S^\alpha$. The approach involves almost no "privacy violation": robot δ only sends an estimate of its rendezvous poses. ■

The following proposition, whose proof trivially follows from [34, Proposition 6.10] (and the fact that the involved matrices are positive definite), ensures that the proposed distributed algorithm converges to the desired solution.

Proposition 3 (Convergence of DGS): The Gauss-Seidel iterations (15) converge to the solution of (13) from any initial estimate $\boldsymbol{y}^{(0)} = [\boldsymbol{y}_\alpha^{(0)}, \boldsymbol{y}_\beta^{(0)}, \ldots]$. ■

Flagged Initialization. According to Proposition 3, the DGS approach converges from any initial condition $\boldsymbol{y}^{(0)}$. However, starting from a "good" initial condition can reduce the number of iterations to converge, and in turn reduce the communication burden (each iteration (16) requires the robots to exchange their estimate of the separators).

In this work, we follow the path of [22] and adopt a flagged initialization. A flagged initialization scheme only alters the first DGS iteration as follows. Before the first iteration, all robots are marked as "uninitialized". Robot α performs its iteration (16) without considering the inter-robot measurements, i.e., eq. (16) becomes $\boldsymbol{y}_\alpha^{(k+1)} = \boldsymbol{H}_{\alpha\alpha}^{-1}\boldsymbol{g}_\alpha$; 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 DGS iteration, also β marks itself as initialized. Repeating this procedure, all robots become initialized after performing the first iteration. The following iterations proceed according to the standard DGS update (16). The following section shows that flagged initialization significantly speeds up convergence.

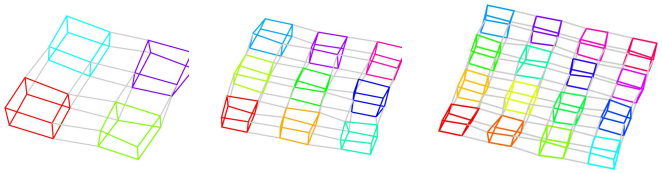
IV. Experiments

We evaluate the DGS approach in large simulations (Section IV-A) and field tests (Section IV-B). The results demonstrate that the proposed approach is accurate, scalable, robust to noise, and parsimonious in terms of communication.

A. Simulation Results

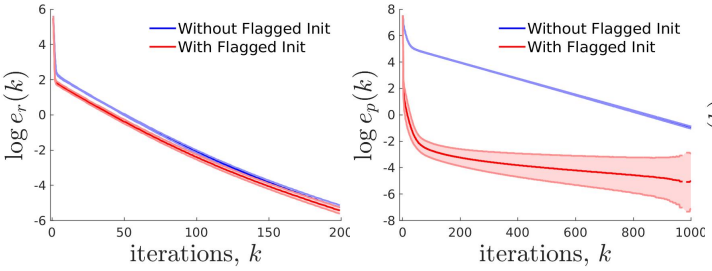
In this section, we characterize the performance of the proposed approach in terms of convergence, scalability (in the number of robots and separators), and sensitivity to noise. For our tests, we created simulation datasets in six different configurations with increasing number of robots: 4, 9, 16, 25, 36 and 49 robots. The robots are arranged in a 3D grid with each robot moving on a cube, as shown in Fig. 4. When the robots are at contiguous corners, they can communicate (gray links). Unless specified otherwise, we generate measurement noise from a zero-mean Gaussian distribution with standard deviation $\sigma_R = 5^\circ$ for the rotations and $\sigma_t = 0.2$m for the translations. Results are averaged over 10 Monte Carlo runs.

In our problem, the DGS approach is used to sequentially solve two linear systems, (7) and (12), which return the minimizers of (6) and (11), respectively. Defining, $m_r \doteq$



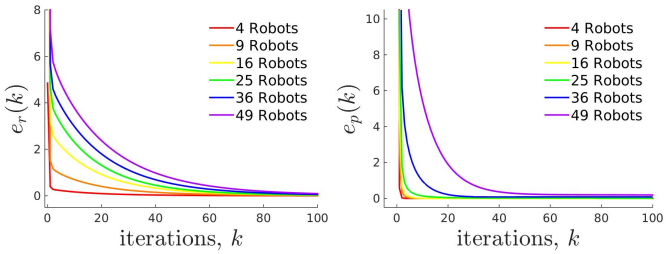
(a) 4 Robots (b) 9 Robots (c) 16 Robots

Fig. 4. Simulated 3D datasets with different number of robots. Robots are shown in different colors. Gray links denote inter-robot measurements.



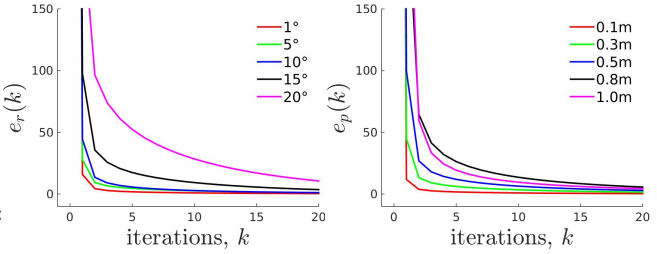
(a) Rotation Estimation Error (b) Pose Estimation Error

Fig. 5. 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 Estimation Error (b) Pose Estimation Error

Fig. 6. Convergence for scenarios with increasing number of robots.



(a) Rotation Noise (b) Translation Noise

Fig. 7. Convergence for increasing levels of noise (scenario with 49 Robots). (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.

$\min_{\boldsymbol{r}} \|\boldsymbol{A}_r \boldsymbol{r} - \boldsymbol{b}_r\|^2$, we use the following metric, named the *rotation estimation error*, to quantify the error in solving (7):

$$e_r(k) = \|\boldsymbol{A}_r \boldsymbol{r}^{(k)} - \boldsymbol{b}_r\|^2 - m_r \tag{17}$$

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

$$e_p(k) = \|\boldsymbol{A}_p \boldsymbol{p}^{(k)} - \boldsymbol{b}_p\|^2 - m_p \tag{18}$$

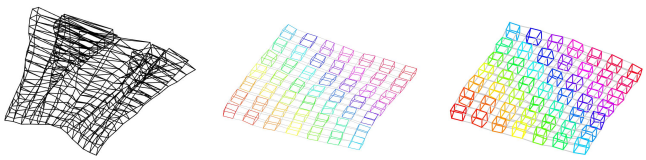
with $m_p \doteq \min_{\boldsymbol{p}} \|\boldsymbol{A}_p \boldsymbol{p} - \boldsymbol{b}_p\|^2$. Ideally, we want $e_r(k)$ and $e_p(k)$ to quickly converge to zero for increasing k.

Flagged Initialization. Let us start by discussing the advantage of the flagged initialization. We compare the flagged initialization against a naive initialization in which the variables ($\boldsymbol{r}^{(0)}$ and $\boldsymbol{p}^{(0)}$, respectively) are initialized to zero. The results, for the dataset with 49 robots, are shown in Fig. 5. In both cases the estimation errors go to zero, but the convergence is faster when using the flagged initialization. The speed-up is significant for the second linear system (Fig. 5b). We noticed a similar advantage across all tests. In the rest of the paper we use the flagged initialization.

Convergence. Fig. 6 shows the average errors $e_r(k)$ and $e_p(k)$ for all the simulated datasets. In all cases the errors quickly converge to zero. For large number or robots the convergence rate becomes slightly slower, while in all cases the errors is negligible in few tens of iterations.

Fig. 7 shows similar statistics for increasing levels of noise and for the scenario with 49 robots. Also in this case, while larger noise implies longer convergence tails, the error becomes sufficiently small after few tens of iterations.

Fig. 8 shows the estimated trajectory after 10 and 1000 iterations of the DGS algorithm for the 49-robot grid. The odometric estimate is shown for visualization purposes, while it is not used in our algorithm. We can see that the estimate



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

Fig. 8. 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 estimate after 10 and 1000 iterations.

after 10 iterations is already visually accurate. The DGS algorithm has an any-time flavor: the trajectory estimates are already accurate after few iterations and asymptotically converge to the centralized estimate.

Accuracy and scalability in the number of robots. While in the previous paragraphs we considered the errors for each subproblem ($e_r(k)$ and $e_p(k)$), in this section we 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 [1] and against a standard (centralized) Gauss-Newton method, available in gtsam [32]. Since the accuracy of the proposed approach depends on the number of iterations, we need to set a stopping condition for the DGS 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 $\|\boldsymbol{p}^{(k+1)} - \boldsymbol{p}^{(k)}\| \leq \eta_p$).

Table I reports the number of iterations and the cost attained in problem (3) (the lower the better), 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 [1], the last two columns of the table confirm that the centralized two-

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

TABLE I

NUMBER OF ITERATIONS AND COST ATTAINED IN PROBLEM (3) FOR INCREASING NUMBER OF 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	#Iter	Cost	Cost	Cost
1	0.05	8.5	2.1	51.0	1.8	1.8	1.8
5	0.1	21.8	14.8	197.8	14.0	14.0	13.9
10	0.2	35.6	58.4	277.7	56.6	56.6	56.0
15	0.3	39.8	130.5	236.8	128.4	129.3	126.0

TABLE II

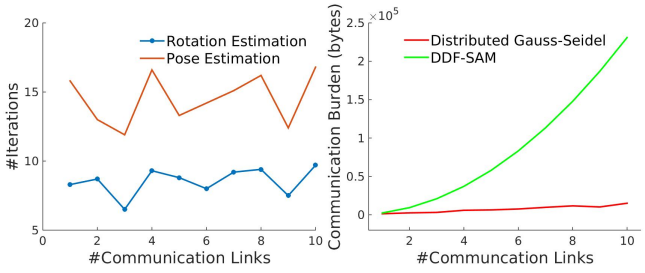
NUMBER OF ITERATIONS AND COST ATTAINED IN PROBLEM (3) FOR INCREASING MEASUREMENT NOISE.

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, with 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. We further test the accuracy of our approach by evaluating the cost and the number of iterations for increasing levels of noise. Table II shows that DGS is able to replicate the accuracy of the centralized two-stage approach, regardless of the noise level.

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 steps. The number of communication links were varied from 1 (single communication) to 10 (communication at every time), hence the number of separators (for each robot) ranges from 1 to 10. Fig. 9a shows the number of iterations required by the DGS algorithm ($\eta_r = \eta_p = 10^{-1}$). The number of iterations is fairly insensitive to the number of communication links.

Fig. 9b compares the information exchanged in the DGS algorithm against a state-of-the-art algorithm, DDF-SAM [18]. In DDF-SAM, each robot sends $K_{\mathsf{GN}} \left[s\, B_p + (s\, B_p)^2 \right]$ bytes, where K_{GN} is the number of iterations required by a GN method applied to problem (3) (we consider the best case $K_{\mathsf{GN}} = 1$), s is the number of separators and B_p is the size of a pose in bytes. In the DGS algorithm, each robots sends $K^r_{\mathrm{DJ}}\ (s\, B_r) + K^p_{\mathrm{DJ}}\ (s\, B_p)$ bytes, where K^r_{DJ} and K^p_{DJ} are the number of iterations required by



(a) (b)

Fig. 9. (a) Number of iterations versus number of separators for the DGS algorithm. (b) Communication burden (bytes of exchanged information) for the DGS and DDF-SAM algorithms, for increasing number of separators.

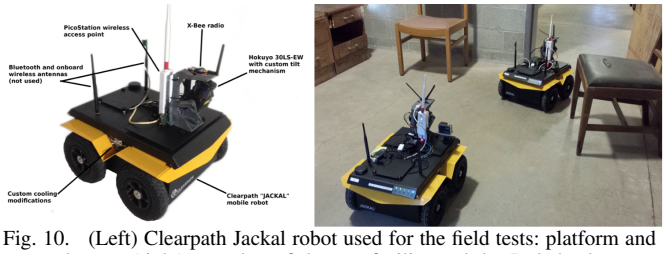


Fig. 10. (Left) Clearpath Jackal robot used for the field tests: platform and sensor layout; (right) snapshot of the test facility and the Jackal robots.

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)[1] and $B_p = 6$ doubles (48 bytes). From Fig. 9b 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 savings.

B. Field Experiments

We tested the DGS approach on field data collected by two Jackal robots (Fig. 10), moving in a MOUT (*Military Operations in Urban Terrain*) facility. Each robot collects 3D scans using an Hokuyo 30LS-EW with a custom tilt mechanism, and uses IMU and wheel odometry to measure its ego-motion. 3D scans are used to compute inter-robot measurements (via ICP) during rendezvous. We evaluated our approach in three different floors of a building.

Fig. 11 shows the trajectories of the two robots in three runs. The figure compares the DGS and the corresponding centralized estimate. Examples of 3D point clouds and occupancy grid maps, reconstructed from our trajectory estimates, are given in Fig. 1. Quantitative results are given in Table III, which reports the cost attained by the DGS algorithm (against a centralized benchmark) and the number of iterations.

#Test	Distributed Gauss-Seidel				Centralized	
	$\eta_r = \eta_p = 10^{-1}$		$\eta_r = \eta_p = 10^{-2}$		Two-Stage	GN
	#Iter	Cost	#Iter	Cost	Cost	Cost
1	10	0.30	78	0.24	0.23	0.23
2	16	0.62	511	0.56	0.54	0.54
3	28	1.20	606	0.87	0.84	0.84

TABLE III

PERFORMANCE OF DGS ON FIELD DATA.

[1] In the linear system (7) we relax the orthogonality constraints hence we cannot parametrize the rotations with a minimal 3-parameter representation.

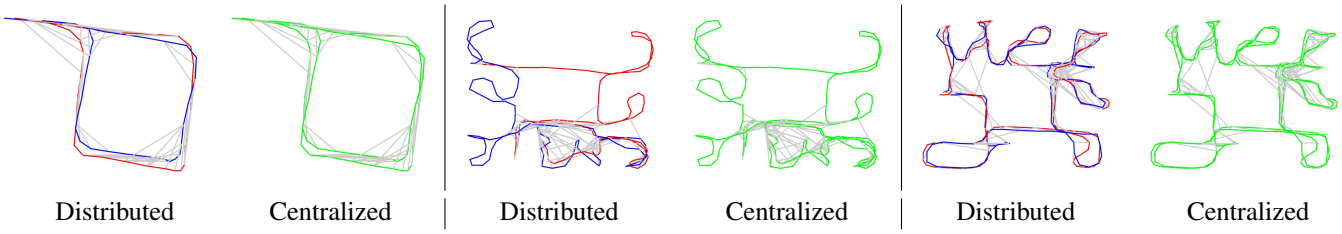


Fig. 11. Field tests: estimated trajectories for the distributed Gauss-Seidel algorithm and for the centralized two-stage approach [1]. Trajectories of the two robots are shown in red and blue, while inter-robot measurements are shown in light gray.

V. Conclusions and Future Work

We proposed a distributed Gauss-Seidel algorithm to estimate the 3D trajectories of multiple cooperative robots from relative pose measurements. The approach has the following merits: (i) communication scales linearly in the number of separators and respects agents' privacy, (ii) the 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.

References

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

[2] L. Paull, G. Huang, M. Seto, and J. Leonard, "Communication-constrained multi-AUV cooperative SLAM," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015.

[3] T. Bailey, M. Bryson, H. Mu, J. Vial, L. McCalman, and H. Durrant-Whyte, "Decentralised cooperative localisation for heterogeneous teams of mobile robots," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[4] V. Indelman, P. Gurfil, E. Rivlin, and H. Rotstein, "Graph-based distributed cooperative navigation for a general multi-robot measurement model," *Intl. J. of Robotics Research*, vol. 31, no. 9, August 2012.

[5] A. Bahr, M. Walter, and J. Leonard, "Consistent cooperative localization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2009, pp. 3415–3422.

[6] J. Dong, E. Nelson, V. Indelman, N. Michael, and F. Dellaert, "Distributed real-time cooperative localization and mapping using an uncertainty-aware expectation maximization approach," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015.

[7] S. Roumeliotis and G. Bekey, "Distributed multi-robot localization," *IEEE Trans. Robot. Automat.*, August 2002.

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

[9] A. Howard, "Multi-robot simultaneous localization and mapping using particle filters," *Intl. J. of Robotics Research*, vol. 25, no. 12, pp. 1243–1256, 2006. [Online]. Available: http://cres.usc.edu/cgi-bin/print_pub_details.pl?pubid=514

[10] L. Carlone, M. K. Ng, J. Du, B. Bona, and M. Indri, "Simultaneous localization and mapping using Rao-Blackwellized particle filters in multi robot systems," *J. of Intelligent and Robotic Systems*, vol. 63, no. 2, pp. 283–307, 2011.

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

[12] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller, "Multiple relative pose graphs for robust cooperative mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Anchorage, Alaska, May 2010, pp. 3185–3192.

[13] M. Lazaro, L. Paz, P. Pinies, J. Castellanos, and G. Grisetti, "Multi-robot SLAM using condensed measurements," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 1069–1076.

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

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

[16] R. Aragues, L. Carlone, G. Calafiore, and C. Sagues, "Multi-agent localization from noisy relative pose measurements," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 364–369.

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

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

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

[20] B. Anderson, I. Shames, G. Mao, and B. Fidan, "Formal theory of noisy sensor network localization," *SIAM Journal on Discrete Mathematics*, vol. 24, no. 2, pp. 684–698, 2010.

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

[22] P. Barooah and J. Hespanha, "Semantic structure from motion," in *Intl. Conf. on Intelligent Sensing and Information Processing*, 20105, pp. 226–231.

[23] ——, "Estimation on graphs from relative measurements," *Control System Magazine*, vol. 27, no. 4, pp. 57–74, 2007.

[24] W. Russell, D. Klein, and J. Hespanha, "Optimal estimation on the graph cycle space," *IEEE Trans. Signal Processing*, vol. 59, no. 6, pp. 2834–2846, 2011.

[25] A. Carron, M. Todescato, R. Carli, and L. Schenato, "An asynchronous consensus-based algorithm for estimation from noisy relative measurements," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 3, pp. 2325–5870, 2014.

[26] R. Aragues, L. Carlone, G. Calafiore, and C. Sagues, "Distributed centroid estimation from noisy relative measurements," *Systems & Control Letters*, vol. 61, no. 7, pp. 773–779, 2012.

[27] J. Thunberg, E. Montijano, and X. Hu, "Distributed attitude synchronization control," in *IEEE Conf. on Decision and Control*, 2011.

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

[29] G. Piovan, I. Shames, B. Fidan, F. Bullo, and B. Anderson, "On frame and orientation localization for relative sensing networks," *Automatica*, vol. 49, no. 1, pp. 206–213, 2013.

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

[31] L. Carlone, D. Rosen, G. Calafiore, J. Leonard, and F. Dellaert, "Lagrangian duality in 3D SLAM: Verification techniques and optimal solutions," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

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

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

[34] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

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