Asynchronous Distributed Smoothing and Mapping via On-Manifold Consensus ADMM

Daniel McGann[1], Kyle Lassak[2], and Michael Kaess[1]

***Abstract*— In this paper we present a fully distributed, asynchronous, and general purpose optimization algorithm for Consensus Simultaneous Localization and Mapping (CSLAM). Multi-robot teams require that agents have timely and accurate solutions to their state as well as the states of the other robots in the team. To optimize this solution we develop a CSLAM back-end based on Consensus ADMM called MESA (Manifold, Edge-based, Separable ADMM). MESA is fully distributed to tolerate failures of individual robots, asynchronous to tolerate communication delays and outages, and general purpose to handle any CSLAM problem formulation. We demonstrate that MESA exhibits superior convergence rates and accuracy compare to existing state-of-the art CSLAM back-end optimizers.**

I. Introduction

Collaborative teams of autonomous robots are desired across numerous application domains. Teams can work more efficiently than individual robots making them useful for applications like search and rescue [1] and scientific exploration [2]. Moreover, heterogeneous teams of robots can use individual specializations to perform tasks beyond the capabilities of a single platform [3]. A key capability required for the effective deployment of multi-robot teams is Collaborative Simultaneous Localization and Mapping (CSLAM) [4]. For downstream tasks like planning, robots need both an estimate of their own state as well as an estimate of states for relevant members of the team. In this work we focus on the CSLAM "back-end" responsible for composing noisy measurements into a state estimate for the robotic team.

Multi-robot teams require this state estimate to be both *accurate* (i.e. optimal for the available information) and *consistent* (i.e. robots agree on a single solution). Furthermore, teams require a method for computing this estimate that is *efficient* with respect to both runtime and communication overhead as well as *resilient* to practical communication network behavior like delayed/dropped messages or communication outages between robots. Finally, robotic teams need a method that is *general purpose* so that teams can make use of all available map representations and sensing modalities.

An attractive approach to the CSLAM back-end is distributed optimization as it can adapt to failures of individual robots. Recent work has investigated the general classes of distributed optimization algorithms and observed that Consensus Alternating Direction Method of Multipliers (C-ADMM) displays superior convergence rates to alternative distributed optimization approaches [5]. However, C-ADMM is yet unexplored in the context of CSLAM.

This work was partially supported by NASA award 80NSSC22PA952, the NSF Graduate Research Fellowship Program, and DEVCOM Army Research Laboratory Distributed, Collaborative, Intelligent Systems and Technology Collaborative Research Alliance (DCIST-CRA) W911NF-17-2-0181.

[1] D. McGann and M. Kaess are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. {danmcgann, kaess}@cmu.edu

[2] K. Lassak is with Astrobotic Technology Inc., Pittsburgh, PA, USA, kyle.lassak@astrobotic.com

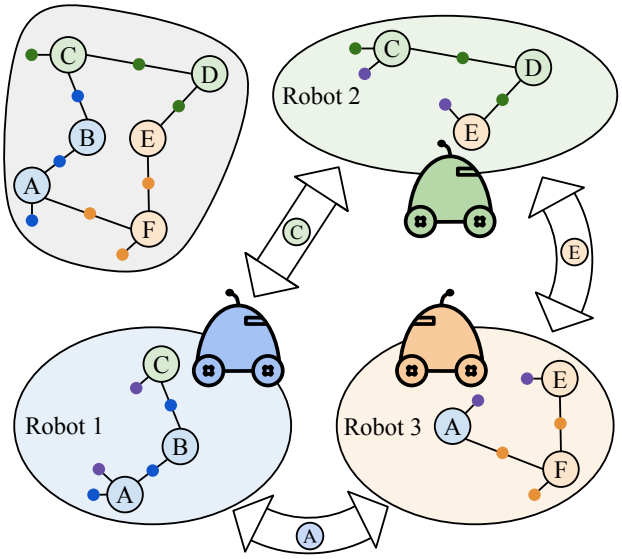


Fig. 1: Example multi-robot factor graph (gray) and its corresponding distribution between agents (blue, green, and orange) in MESA. When asynchronously communicating (arrows), the robots send only their solutions for shared variables (A, C, E). These are used to construct "Biased Priors" (purple factors) incorporated into each robot's local graph to enforce consistency across the team.

Contributions: In this paper we first revisit C-ADMM and discuss extensions relevant to our proposed algorithm. We then propose a novel algorithm based on C-ADMM called **M**anifold, **E**dge-based, **S**eparable **A**DMM (MESA), an efficient, general purpose CSLAM back-end that permits asynchronous communication (see Fig. 1). We then explore variants of MESA derived from different manifold constraint formulations. We conclude by validating MESA under a variety of conditions seen in CSLAM problems and demonstrating that MESA achieves superior performance to existing distributed CSLAM back-end optimizers.

II. Related Work

ADMM was previously proposed to solve CSLAM problems by Choudhary et al. who proposed Multi-Block ADMM (MB-ADMM) to solve generic SLAM problems distributed on a cluster [6]. MB-ADMM, however, is not resilient as it requires synchronized communication and unlike C-ADMM is proven to diverge for even some linear problems [7].

Other works have investigated alternate optimization methods for solving generic CSLAM problems. Cunningham et al. proposed Distributed Data Fusion Smoothing and Mapping (DDF-SAM) and DDF-SAM2. These algorithms share marginal information between robots being careful to avoid double counting [8], [9]. These approaches, however, fail to enforce equal linearization points between robots resulting in potentially inconsistent and inaccurate solutions. In another vein, Murai et al. proposed a Loopy Believe Propagation (LBP) based method [10], [11]. While distributed LBP

demonstrates good performance, its convergence rate and consistency are unexplored in currently published work.

Another line of research has focused on distributed Pose Graph Optimization (PGO). PGO is a subset of CSLAM problems making these methods limited in their applicability. Choudhary et al. proposed the Distributed Gauss-Seidel (DGS) algorithm [12]. DGS solves the chordal relaxation of the PGO problem [13] using distributed successive over relaxation. DGS often performs well, however, it requires synchronous communication and thus is not resilient to unreliable networks. Furthermore, it optimizes an approximation of the PGO objective, limiting its accuracy. Cristofalo et al. proposed another PGO method called GeoD which avoids approximation and optimizes the "geodesic" PGO formulation enabling better accuracy [14]. GeoD treats each pose as an independent agent and optimizes via distributed gradient descent making it inefficient and slow to converge.

Recent work on distributed PGO relaxes PGO to a Semi-Definite Program (SDP) [15]. The proposed algorithm DC2-PGO and its asynchronous variant ASAPP solve CSLAM as a SDP with distributed Riemannian Gradient Descent (RGD). DC2-PGO and ASAPP are very accurate, can provide certificates of correctness, and ASAPP's asynchronous communication model makes it resilient to network issues [16]. However, like GeoD, by optimizing via distributed gradient descent DC2-PGO and ASAPP exhibit slow convergence.

As we can see, prior distributed SLAM methods struggle to achieve a sufficient solution. General purpose methods lack either consistency, accuracy, or resiliency. While some PGO methods achieve all of these goals they are slow to converge and limited in their applicability. The proposed algorithm, MESA, strives to achieve all of these goals.

III. Consensus ADMM

We begin by reviewing Consensus ADMM [17], a popular fully distributed optimization method for solving consensus optimization problems of the form in (1).

$$\begin{aligned} \underset{\bar{x}\in\mathbb{R}^{|\mathcal{R}|n}}{\arg\min} \quad & \sum_{i\in\mathcal{R}} f_i(x_i) \\ \text{s.t.} \quad & x_i = x_j \ \forall\ (i,j)\in\mathcal{E} \end{aligned} \tag{1}$$

Where $\mathcal{R}$ is the set of all agents, f_i is the local objective of agent i, $x_i \in \mathbb{R}^n$ is the copy of decision variables held by agent i, and $\bar{x} \in \mathbb{R}^{|\mathcal{R}|n}$ is the concatenation of all agent's local decision variables. Agents communicate over an undirected network made up of the agents $\mathcal{R}$ and communication links $\mathcal{E}$. The constraints force neighboring agents to agree on a solution. By induction this forces all agents to converge to a single joint solution to the problem.

Applying standard ADMM to problem (1) produces an algorithm that requires centralized updates [18]. To produce a fully distributed algorithm, C-ADMM augments problem (1) with additional variables. For each edge in the communication network $(i,j)\in\mathcal{E}$, C-ADMM introduces two new variables $z_{(i,j)}$ and $z_{(j,i)}$. The variable $z_{(i,j)}$ is held by agent i and can be interpreted as agent i's estimate of agent j's local solution. We will refer to these variables as "edge variables" and use them rewrite (1) as (2).

$$\begin{aligned} \underset{\bar{x}\in\mathbb{R}^{|\mathcal{R}|n},\ \bar{z}\in\mathcal{Z}}{\arg\min} \quad & \sum_{i\in\mathcal{R}} f_i(x_i) \\ \text{s.t.} \quad & x_i = z_{(i,j)},\ x_j = z_{(j,i)}\ \forall\ (i,j)\in\mathcal{E} \end{aligned} \tag{2}$$

Where $\bar{z}$ is the concatenation of all $z_{(i,j)}$ and $\mathcal{Z}$ is the space of $\mathbb{R}^{2|\mathcal{E}|n}$ which has been further constrained such that $z_{(i,j)} = z_{(j,i)}$. This augmentation increases the number of constraints but does not change their meaning. C-ADMM solves (2) according to the update process (3, 4, 5, 6).

$$\bar{x}^{k+1} = \underset{\bar{x}\in\mathbb{R}^{|\mathcal{R}|n}}{\arg\min}\ \mathcal{L}(\bar{x}, \bar{z}^k, \bar{\lambda}^k, \beta^k) \tag{3}$$

$$\bar{z}^{k+1} = \underset{\bar{z}\in\mathcal{Z}}{\arg\min}\ \mathcal{L}(\bar{x}^{k+1}, \bar{z}, \bar{\lambda}^k, \beta^k) \tag{4}$$

$$\bar{\lambda}^{k+1} = \bar{\lambda}^k + \beta^k(D\bar{x}^{k+1} - \bar{z}^{k+1}) \tag{5}$$

$$\beta^{k+1} = \alpha\beta^k \tag{6}$$

Where $\bar{\lambda}$ is the concatenation of all dual variables (one corresponding to each constraint), $D \in \mathbb{R}^{(2|\mathcal{E}|n)\times(|\mathcal{R}|n)}$ maps each x_i in $\bar{x}$ to all corresponding $z_{(i,j)}$ in $\bar{z}$, β is the penalty term, α is a scaling factor hyper-parameter, k is the iteration count, and $\mathcal{L}$ is the problem's Augmented Lagrangian (7).

$$\sum_{i\in\mathcal{R}} f_i(x_i) + \sum_{j\in\mathcal{N}_i} \left\langle \lambda_{(i,j)}, x_i - z_{(i,j)} \right\rangle + \frac{\beta}{2}\left\| x_i - z_{(i,j)} \right\|^2 \tag{7}$$

Where $\mathcal{N}_i$ are the neighbors of agent i. Solving these iterates can be fully distributed as (3) can be solved by each agent minimizing its local Augmented Lagrangian independently. After this minimization C-ADMM stipulates that all agents communicate their results to all neighbors to allow each agent to independently solve (4), (5), and (6). When all f_i are convex C-ADMM converges linearly [19].

Within existing literature there are two extensions to the base C-ADMM algorithm that will be relevant to MESA. Firstly, C-ADMM assumes that each agent's objective relies on all optimization parameters. However, there are a large class of problems where each agent's objective relies on only a subset of the variables. We refer to such problems as "separable" consensus optimization problems, a reference to Separable Optimization Variable ADMM (SOVA) [20]. SOVA proposed a simple but effective extension in which agents hold only the variables required by their local objective and we impose constraints on only the variables shared between agents. Separable problems take the form of (8).

$$\begin{aligned} \underset{\{x_0\in\mathbb{R}^{n_0},\ \ldots,\ x_r\in\mathbb{R}^{n_r}\}}{\arg\min} \quad & \sum_{i\in\mathcal{R}} f_i(x_i) \\ \text{s.t.} \quad & A_{(i,j)}x_i = B_{(i,j)}x_j\ \forall\ (i,j)\in\mathcal{E} \end{aligned} \tag{8}$$

Where $A_{(i,j)} \in \mathbb{R}^{n_{(i,j)}\times n_i}$ and $B_{(i,j)} \in \mathbb{R}^{n_{(i,j)}\times n_j}$ map the variables shared between agents i and j into a common space.

Secondly, C-ADMM assumes an algorithmic structure in which, at each iteration, all agents communicate with all neighbors. However, C-ADMM has been proven to converge for significantly less restrictive communication models. Edge-based C-ADMM assumes that at each iteration of the algorithm only a subset of communication edges are active and each agent, therefore, communicates with only some of its neighbors [21]. Without loss of generality this model can be simplified to assume that at each iteration only two

agents communicate with each other. Under this Edge-based communication model C-ADMM is proven to convergence with rate $O(1/k)$ where k is the number of iterations [22].

IV. Methodology

In this section we first define the CSLAM problem and demonstrate that it can be transformed into an instance of a Separable Consensus ADMM problem with on-manifold decision variables. We then propose a novel algorithm MESA for solving general CSLAM problems with asynchronous communication. Finally, we derive variants of the algorithm driven by different approaches to modeling constraints between the manifold decision variables found in CSLAM.

We focus on solving generalized CSLAM problems, which can be defined as Maximum-A-Posterori (MAP) inference.

$$\Theta_{MAP} = \arg\max_{\Theta \in \Omega} P(\Theta | M) \tag{9}$$

Where Θ are all the variables of interest, Ω is the product manifold formed by the manifold of each element within Θ, and M is the set of measurements. When we assume our measurements are affected by Gaussian noise (i.e. $m \sim \mathcal{N}\left(\mu_m, \Sigma_m\right)$), problems of this form can be solved via nonlinear least squares optimization [23].

$$\Theta_{MAP} = \arg\min_{\Theta \in \Omega} \sum_{m \in M} \left\| h_m(\Theta) - m \right\|^2_{\Sigma_m} \tag{10}$$

Where h_m is the measurement prediction function that computes the expected m from an estimate of the state.

A. Manifold, Edge-based, Separable ADMM (MESA)

In the case of a multi-robot team $\mathcal{R}$, the variables Θ and measurements M are distributed across robots. Let M_i denote the measurements made by robot i and Θ_i denote the variables that are observed by the measurements in M_i. Multiple robots may observe the same variable (e.g. two robots observe the same landmark). Therefore, while M_i are disjoint subsets of M, each Θ_i is a non-disjoint subset of Θ. Let $\mathcal{S}_{(i,j)} \triangleq \Theta_i \cap \Theta_j$ represent the variables shared between robots i and j where for $\theta_s \in \mathcal{S}_{(i,j)}$ we denote θ_{s_i} as the copy of that variable owned by robot i. We can therefore re-write (10) to reflect this distribution of information.

$$\begin{aligned} \Theta_{MAP} = \underset{\Theta_i \in \Omega_i \forall i \in \mathcal{R}}{\arg\min} \quad & \sum_{i \in \mathcal{R}} \sum_{m \in M_i} \left\| h_m(\Theta_i) - m \right\|^2_{\Sigma_m} \\ \text{s.t.} \quad & q_s(\theta_{s_i}, \theta_{s_j}) = 0 \\ & \forall\ \theta_s \in \mathcal{S}_{(i,j)}\ \forall\ (i,j) \in \mathcal{E} \end{aligned} \tag{11}$$

Where q_s is a function that compares the equality appropriately for the manifold to which θ_s belongs and returns $0 \in \mathbb{R}^d$ if and only if θ_{s_i} and θ_{s_j} are equal. The generic function q_s is used as there exists potentially many ways to compare equality of on-manifold objects. Concrete implementations of q_s will be explored in detail in Sec. IV-C.

From (11) we can see that the general CSLAM problem is an instance of a separable optimization problem (8) as each robot's cost function affects only a subset of the global variables and robots share sparse sets of variables. However, unlike (8) and all standard ADMM formulations, CSLAM problems typically include on-manifold decision variables which we handle with generic constraint functions q_s.

To make the CSLAM problem fully distributed we apply the same method as C-ADMM and augment the problem with edge variables. Specifically, for every shared variable θ_s shared along edge (i,j) we introduce edge variables $z_{(i,j)_s}$ and $z_{(j,i)_s}$. The addition of edge variables allows us to re-write the constraints of our problem as follows in (12).

$$\begin{aligned} \Theta_{MAP} = \underset{\Theta_i \in \Omega_i \forall i \in \mathcal{R},\ Z \in \mathcal{Z}}{\arg\min} \quad & \sum_{i \in \mathcal{R}} \sum_{m \in M_i} \left\| h(\Theta_i) - m \right\|^2_{\Sigma_m} \\ \text{s.t.} \quad & q_s(\theta_{s_i}, z_{(i,j)_s}) = 0 \\ & q_s(\theta_{s_j}, z_{(j,i)_s}) = 0 \\ & \forall\ \theta_s \in \mathcal{S}_{(i,j)}\ \forall\ (i,j) \in \mathcal{E} \end{aligned} \tag{12}$$

Where Z is the set of all $z_{(i,j)_s}$ and $\mathcal{Z}$ is the appropriate product manifold further constrained such that $z_{(i,j)_s} = z_{(j,i)_s}$. Next we use (3, 4, 5, 6, 7) to derive on-manifold, separable, C-ADMM iterates required to solve (12).

$$\begin{aligned} \Theta_i^{k+1} = & \underset{\Theta_i \in \Omega_i}{\arg\min} \quad \sum_{m \in M_i} \left\| h_m(\Theta_i^k) - m \right\|^2_{\Sigma_m} \\ & + \sum_{j \in \mathcal{N}_i} \sum_{s \in \mathcal{S}_{(i,j)}} \left\langle \lambda^k_{(i,j)_s}, q_s\left(\theta_{s_i}, z^k_{(i,j)_s}\right) \right\rangle \\ & + \sum_{j \in \mathcal{N}_i} \sum_{s \in \mathcal{S}_{(i,j)}} \frac{\beta^k}{2} \left\| q_s\left(\theta_{s_i}, z^k_{(i,j)_s}\right) \right\|^2 \end{aligned} \tag{13}$$

$$\begin{aligned} z^{k+1}_{(i,j)_s} = & \underset{z_s \in \mathcal{Z}_s}{\arg\min} \\ & \left\langle \lambda^k_{(i,j)_s}, q_s\left(\theta^{k+1}_{s_i}, z_s\right) \right\rangle + \frac{\beta^k}{2} \left\| q_s\left(\theta^{k+1}_{s_i}, z_s\right) \right\|^2 \\ & + \left\langle \lambda^k_{(j,i)_s}, q_s\left(\theta^{k+1}_{s_j}, z_s\right) \right\rangle + \frac{\beta^k}{2} \left\| q_s\left(\theta^{k+1}_{s_j}, z_s\right) \right\|^2 \end{aligned} \tag{14}$$

$$\lambda^{k+1}_{(i,j)_s} = \lambda^k_{(i,j)_s} + \beta^k \left(q_s\left(\theta^{k+1}_{s_i}, z^{k+1}_{(i,j)_s}\right) \right) \tag{15}$$

$$\beta^{k+1} = \alpha \beta^k \tag{16}$$

Where we have introduced a dual variable $\lambda_{(i,j)_s}$ for each constraint and written the iterates for individual variables where they can be solved for independently.

Using these iterates with a C-ADMM algorithm, however, is insufficient for typical CSLAM problems. Multi-robot teams operating in the field cannot assume reliable network infrastructure. Therefore, robots may not be able to synchronously communicate with all neighbors as is required by C-ADMM. To make our algorithm resilient to these conditions we combine these iterates with an Edge-based communication model. Within the Edge-based model, communication dropouts and message delay/loss are modeled simply as iterations in which the affected robots do not communicate. The resulting asynchronous algorithm can tolerate expected network conditions. To fully address asynchronous communication we further modify the iterates adding unique penalty terms $\beta_{(i,j)}, \beta_{(j,i)}$ for each edge in $\mathcal{E}$.

The edge-based communication model combined with our on-manifold, separable, C-ADMM iterates produces MESA (Alg. 1) which is also outlined in Fig. 1.

Remark 1 (MESA Initialization)**:** *Without loss of generality we define $\lambda^0_{(i,j)_s} = \mathbf{0}$ where $\mathbf{0}$ is of proper dimension. Further, we assume $z_{(i,j)_s} = \theta_{s_i}$ until the first communication between robots i and j.*

Algorithm 1 Manifold, Edge-based, Separable, ADMM (MESA)

In: Robots $\mathcal{R}$, communication links $\mathcal{E}$, local estimates $\{\Theta_0^0, \Theta_1^0, ..., \Theta_r^0\}$
Out: Final Variable Estimates $\left\{\Theta_0^{final}, \Theta_1^{final}, ..., \Theta_r^{final}\right\}$
$\lambda_{(i,j)_s}, \lambda_{(j,i)_s} \leftarrow \mathbf{0}\ \forall\ s \in \mathcal{S}_{(i,j)}\ \forall\ (i,j) \in \mathcal{E}$
$z_{(i,j)_s} \leftarrow \theta_{s_i}^0, z_{(j,i)_s} \leftarrow \theta_{s_j}^0\ \forall\ s \in \mathcal{S}_{(i,j)}\ \forall\ (i,j) \in \mathcal{E}$
while Not Converged **do**
  if Communication available between robot i and robot j **then**
    In parallel update Θ_i and Θ_j with (13)
    Between i and j communicate $\theta_{s_i}, \theta_{s_j}\ \forall\ \theta_s \in \mathcal{S}_{(i,j)}$
    In parallel update $z_{(i,j)_s}, z_{(j,i)_s}\ \forall\ \theta_s \in \mathcal{S}_{(i,j)}$ with (14)
    In parallel update $\lambda_{(i,j)_s}, \lambda_{(j,i)_s}\ \forall\ \theta_s \in \mathcal{S}_{(i,j)}$ with (18)
    In parallel update $\beta_{(i,j)}^{k+1} = \alpha\beta_{(i,j)}^k$ and $\beta_{(j,i)}^{k+1} = \alpha\beta_{(j,i)}^k$
  end if
end while

Remark 2 (MESA Theoretical Guarantees)**:** *Due to the non-convex objective and non-convex variable space found in CSLAM, MESA does not share the convergence guarantees of C-ADMM, SOVA, or Edge-based C-ADMM. Recent work has proven that, under certain assumptions, non-convex C-ADMM will converge [24]. However, this proof does not addressed on-manifold decision variables. More importantly, for practical CSLAM it is unclear whether the problems will meet the required assumptions. Therefore, we do not pursue a convergence proof for MESA. Instead we opt to demonstrate convergence of MESA empirically (see Sec. V).*

B. MESA Implementation

Implementation of Alg. 1 requires solving the optimization problems (13) and (14). As originally demonstrated by Choudhary et al. [6] the Augmented Lagrangian in (13) can be represented as a factor-graph by re-writing the dual and penalty terms as "Biased Priors". Specifically, the identity that $\arg\min_a \langle b, a\rangle + (\beta/2)\|a\|^2 = \arg\min_a (\beta/2)\|a + b/\beta\|^2$ when b is constant is used to re-write the terms as (17).

$$\frac{\beta_{(i,j)}}{2}\left\| q_s\left(\theta_s, z_{(i,j)_s}\right) + \frac{\lambda_{(i,j)_s}}{\beta_{(i,j)}}\right\|^2 \tag{17}$$

This allows us to compute (13) using existing sparse nonlinear least squares libraries like `gtsam` [25].

The edge variable update (14) could be implemented similarly and each $z_{(i,j)_s}$ independently solved via a nonlinear optimization. However, to improve efficiency, we would prefer that (14) be solved in closed form. Depending on the selection of constraint function and edge variable space, a closed form solution may exist or may be approximated.

Computing updates to dual variables is straightforward according to (15). In this, dual variables are updated based on the error between a θ_{s_i} and $z_{(i,j)_s}$. However, this error underestimates the magnitude of the constraint satisfaction gap (i.e. the error between θ_{s_i} and θ_{s_j}). We observe that updating dual variables according to (18) is more reflective the proper magnitude and improves convergence speed.

$$\lambda_{(i,j)_s}^{k+1} = \lambda_{(i,j)_s}^k + \beta_{(i,j)}^k\left(q_s\left(\theta_{s_i}^{k+1}, \theta_{s_j}^{k+1}\right)\right) \tag{18}$$

C. Manifold Constraint Functions

To compute (13), (14), and (18) we must also concretely define the constraint function used for each shared variable. The selection of constraint function q_s depends on the type of our shared variables θ_s and the type selected for edge variables $z_{(i,j)_s}$. In the existing C-ADMM literature (see Sec. III) all algorithms assume vector-valued decision variables and thus that all constraints are linear ($q(a,b) \triangleq a - b$) which permits a closed form solution to (14) of $z_{(i,j)_s}^{k+1} = \frac{1}{2}(\theta_{s_i}^{k+1} + \theta_{s_j}^{k+1})$. For linear variables in CLSAM (e.g. landmarks) this approach should be taken.

However, in CSLAM we are typically also optimizing over robot poses that are contained on the SE(N) manifold. For such on-manifold variables we identify four unique constraint functions (Geodesic, Approximate-Geodesic, Split, and Chordal). For each we derive the corresponding closed form solution to (14) or an approximate solution if no closed form solution exists. A summary of constraint functions and shared variable updates can be found in Table I.

TABLE I: Constraint functions for SE(N) objects and their corresponding closed form solutions to (14). Where SPLIT interpolates the translation component linearly and the rotation component spherically [26], Vec returns a vector of the objects non-constant matrix elements [27], and p is the dimension of the tangent space for SE(N). Manifold object notation is derived from the work of Solà et al. [28]. The geodesic z update is approximated by the case where $\lambda_{(i,j)_s} = \lambda_{(j,i)_s} = \mathbf{0}$.

Function	$z \in$	$q(\theta, z)$	$z_{(i,j)_s}^{k+1}$
Geodesic	SE(N)	$\mathrm{Log}\left(z^{-1} \circ \theta\right)$	$\mathrm{SPLIT}\left(\theta_{s_i}, \theta_{s_j}, 0.5\right)$
Apx.-Geo.	$\mathbb{R}^p$	$\mathrm{Log}\left(\theta\right) - z$	$\frac{1}{2}\left(\mathrm{Log}\left(\theta_{s_i}\right) + \mathrm{Log}\left(\theta_{s_j}\right)\right)$
Split	SE(N)	$\begin{bmatrix}\mathrm{Log}\left(R_z^{-1}R_\theta\right) \\ t_\theta - t_z\end{bmatrix}$	$\mathrm{SPLIT}\left(\theta_{s_i}, \theta_{s_j}, 0.5\right)$
Chordal	$\mathbb{R}^{N^2+N}$	$\mathrm{Vec}(\theta) - z$	$\frac{1}{2}\left(\mathrm{Vec}\left(\theta_{s_i}\right) + \mathrm{Vec}\left(\theta_{s_j}\right)\right)$

Each constraint function produces a slight variant of the MESA algorithm. We identify each variant by the name of the constraint function used (i.e. Geodesic MESA is the MESA algorithm that uses the Geodesic constraint).

Remark 3 (Hyper-Parameters)**:** *All MESA variants use* $\beta_{(i,j)}^0 = 200$ *and* $\alpha = 1$ *with the exception of Split MESA that uses* $\alpha = 1.2$*. Note that these parameters are modified for some experiments.*

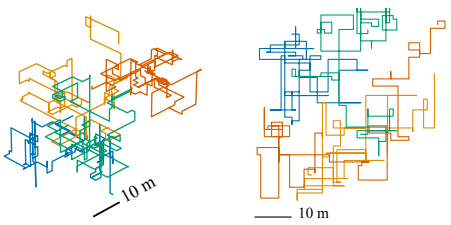
V. Experiments

In this section we evaluate the performance of MESA to solve representative CSLAM problems. We first compare the four variants of the algorithm induced by choice of constraint function. With the best performing variants we then explore the effect of problem parameters (length, initialization quality, number of robots, and types of measurements) to validate its performance across different CSLAM scenarios. Finally, we compare MESA to state of the art works and demonstrate its superior convergence compared to existing methods.

A. Experiment Setup

1) Dataset Generation: To explore different CSLAM conditions we generate datasets by sampling odometry measurements from a categorical distribution over forward motion and a $\pm 90°$ rotation around each axis. At each time-step robots add intra-robot loop closures with probability $p = 0.4$ if the current pose is nearby a previous pose. Every k steps we search for inter-robot loop closures. Unless modified, each experiment consists of 10 datasets generated with 4 robots traversing a 400 pose long trajectory. All measurements are relative poses subject to a noise model of ($\sigma_r = 1°, \sigma_t = 0.05m$). With this general framework we can generate a variety of 2D and 3D datasets (e.g. Fig 2).

2) Metric: Single robot optimization tasks are often evaluated using the cost function residual (10). In multi-robot



(a) Example 3D Dataset **(b)** Example 2D Dataset

Fig. 2: Ground-truth from example synthetic datasets. Colors represent different robots.

problems, however, we cannot directly apply this metric. For each *shared variable*, there exists a local copy on each robot that observed the variable. We want a *consistent* solution such that all local copies are equal. However, practically local copies will differ slightly and it is ambiguous which copy to use to compute the residual. Moreover, in cases where local copies differ we want a metric that reflects the inconsistency. To meet these goals we introduce the *Mean Residual* (r^2_{mean}). Mean residual is an extension of the SLAM cost function in which, the cost contribution for any factor that affects shared variables is averaged over the cost from all possible combinations of solutions. If all shared variables agree exactly, r^2_{mean} is the same as the SLAM cost function and represents the accuracy of the optimized solution. If shared variables disagree, r^2_{mean} still captures this accuracy but also monotonically increases with any shared variable disagreement. Concretely we define r^2_{mean} as:

$$r^2_{mean} = \sum_{m \in M} \frac{1}{|\mathcal{C}|} \sum_{(i,j,\ldots) \in \mathcal{C}} \left\| h_m\left(\theta_{a_i}, \theta_{b_j}, \ldots\right) - m \right\|^2_{\Sigma_m} \quad (19)$$

Where, for a factor on variables $\{\theta_a, \theta_b \ldots\}$, $\mathcal{C}$ represents the set of combinations of local solutions to these variables. For example if a factor affects variables a and b where a is shared between robots $\{i, j\}$ and b is shared between robots $\{k, l\}$ then $\mathcal{C} = \{(i,k),(i,l),(j,k),(j,l)\}$. If a and b are not shared and owned by robot i then $\mathcal{C}$ is simply $\{(i,i)\}$.

B. MESA Variant Exploration

In our first experiment we explore the performance of different variants of MESA. This experiment is run on 20 random 3D datasets and a summary of r^2_{mean} achieved by each variant are represented by box-plots.

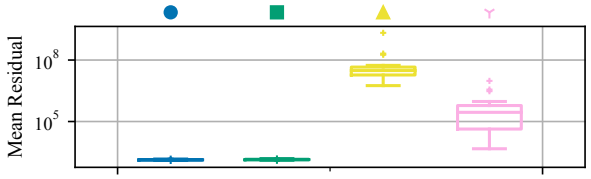


Fig. 3: Comparison of MESA variants derived from Geodesic (●), Split (■), Approximate-Geodesic (▲) and Chordal (ϒ) constraints on 20 synthetic 3D pose graph datasets. Split and Geodesic constraints significantly outperform the alternatives.

As can be seen in Fig. 3 the selection of constraint function has a significant affect on the performance of the algorithm. We can see that the Geodesic and Split constraints significantly outperform the Chordal and Approximate-Geodesic constraints. Throughout the rest of this paper we will present only results from the Geodesic and Split variants.

C. MESA Generalization

Next we look to validate that MESA generalizes to different CSLAM scenarios. For each scenario we present results in comparison to two baselines: a Centralized approach in which all measurements are aggregated and solved using Levenberg-Marquardt and an Independent approach in which robots ignore all inter-robot measurements and solve their local factor graph independently. Specifically, we explore:

1) Trajectory Length (Fig. 4a): First we evaluate how MESA handles different sizes of 3D factor graphs and vary trajectory length from 200 to 1000 poses for all robots.

2) Problem Initialization (Fig. 4b): Next we evaluate how initialization quality affects MESA. The quality of initialization from odometry is proportional to the magnitude of the noise model and thus we vary the measurement noise used to generate each dataset. This trial is run on 20 3D datasets.

3) Problem Scale (Fig. 4c): We also inspect how MESA scales with the size of the robot team. To do so we vary the number of robots in each 3D dataset from 2 to 16.

4) Measurement Models (Fig. 4d): In this sub-experiment we evaluate the ability of MESA to handle generalized 2D CSLAM problems. We explore two types of inter-robot loop-closure measurements (range and bearing+range). These measurements are particularly compelling as they can be made locally and do not require any communication of raw sensor data to support a distributed front-end. For this scenario both MESA variants use $\beta^0_{(i,j)} = 2$ and $\alpha = 1.05$.

As can be seen in Fig. 4, both the Geodesic and Split MESA variants generalize across different conditions found in CSLAM. In all scenarios the algorithms converge toward the centralized solution indicating that the algorithms are both accurate and reliable. Moreover, this experiment validates that collaboration between robots can significantly improve localization performance over independent operation.

D. Prior Work Comparison

In our final experiment we evaluate the performance of MESA relative to existing prior works namely DDF-SAM2 [9], MB-ADMM [6], DGS [12], and ASAPP [16]. Since ASAPP and DGS are distributed PGO algorithms we limit our experiments in this section to pose graph CSLAM.

Remark 4 (Prior Work Implementation Details)**:** *DDF-SAM2 is implemented by the authors. MB-ADMM is provided by the original authors*[1] *and uses parameters from [6]. DGS is provided by the original authors*[2] *and uses default parameters. ASAPP is provided by the original authors*[3] *and we use* 100 *RGD steps per iteration with step size* 0.001*. A small step size was used as larger step sizes sometimes caused divergence and we ran too many trials to permit tuning the step size for each dataset as done in [16]. All methods were limited to a maximum of* $500 * |\mathcal{E}| * |\mathcal{R}|$ *communications.*

1) Benchmark Datasets: We first compare the performance of these methods on partitioned versions of benchmark SLAM datasets. Each dataset is partitioned into 5 subgraphs using METIS partitioning [6]. Table II compares the achieved r^2_{mean}. This sub-experiment omits Independent and

[1] https://github.com/itzsid/admm-slam
[2] https://github.com/CogRob/distributed-mapper
[3] https://github.com/mit-acl/dpgo

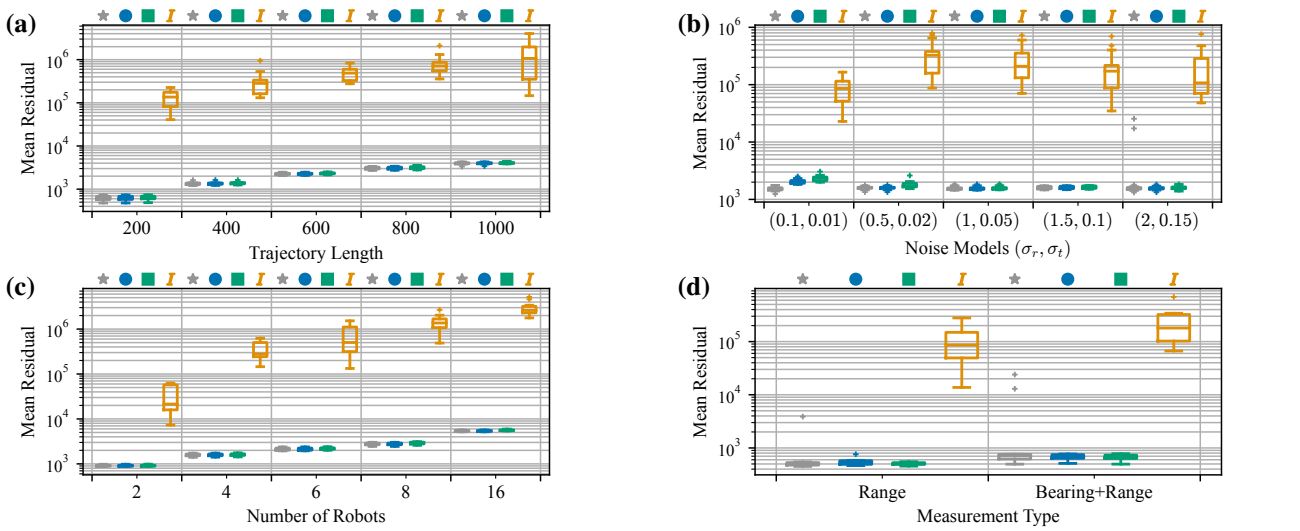


Fig. 4: Performance of Geodesic MESA (●) and Split MESA (■) compared to Centralized (★) and Independent (*I*) baselines under a variety of conditions. The MESA variants consistently converge to the centralized solution indicating they are accurate and generalize across different conditions found in CSLAM problems.

DDF-SAM2 results as not all partitions have priors and these methods will be under-determined. In this sub-experiment both MESA variants use $\beta^0_{(i,j)} = 2$ and $\alpha = 1.05$.

TABLE II: Mean residuals achieved by MESA and prior works on benchmark datasets: Sphere2500 [29], Parking Garage [13], and Torus [13].

Dataset	Sphere2500	Parking Garage	Torus
Centralized	675.7	0.634	29981
MB-ADMM [6]	1.7e+22	0.638	2.5e+14
DGS [12]	1204.9	0.650	12254
ASAPP [16]	940.2	0.651	**12196**
Geodesic MESA	**676.2**	**0.636**	30002
Split MESA	1054.7	0.645	30014

Table II illustrates that MESA is able to converge very closely to the centralized solution across all datasets. It also highlights that, like the centralized solution, MESA is a local solver and can fall into local optima like on the Torus dataset.

2) Empirical Convergence: We also explore the performance of each algorithm with respect to its convergence speed. As each algorithm's implementation differs, we propose a normalized measure of runtime – the number of communications. We define one communication as when a pair of robots pass bidirectionally any amount of information. For example an iteration of Alg. 1 results in one communication.

Communication is expected to be the largest bottleneck in the CSLAM pipeline and, for all algorithms, the number of communications is proportional to both computational cost and communication burden. Therefore, the number of communications provides a holistic metric to represent the complexity of the algorithm. To account for varying convergence criteria, we run all algorithms with a tight stopping condition and report results (r^2_{mean} and communications) when the algorithm reaches within 1% of its final r^2_{mean}. To reduce variance, this experiment is run on 200 3D datasets.

As we can see in Fig. 5 both MESA variants exhibit significantly faster convergence than the prior works while providing superior accuracy. Though difficult to see in Fig. 5, Split MESA converges more quickly than Geodesic MESA at the cost of reduced accuracy. ASAPP, despite being

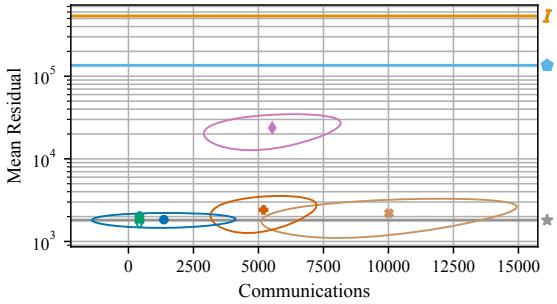


Fig. 5: Accuracy vs. Communications of Geodesic MESA (●) and Split MESA (■) compared to prior works [DGS (✤), ASAPP (✖), MB-ADMM (⧫), DDF-SAM2 (⬟)] and baselines [Centralized (★), Independent (*I*)] on 200 synthetic 3D pose graph datasets. Ellipses depict 3σ uncertainty bounds. Methods that require only one round of communication are shown as horizontal lines. MESA outperforms prior works both with respect to accuracy and convergence speed.

certifiably correct, exhibits worse performance than MESA as it often reached the communication limit before fully converging. This experiment also shows that, as expected, DGS provides less accurate results due to its chordal approximation of the PGO problem, MB-ADMM struggles to converge, and DDF-SAM2's performance lags other methods as it does not produce consistent solutions.

VI. Conclusion and Future Work

In this work we presented MESA, a distributed algorithm for solving general CSLAM problems. MESA exhibits a superior convergence rate and final accuracy compare to prior works and accomplishes this for generalized CSLAM problems while permitting asynchronous communication.

While MESA demonstrates excellent performance across experiments, the quality of its convergence can be dependent on hyper-parameters β^0 and α. Determining these parameters automatically would be a good extension to MESA. Additionally, like most prior works, MESA is a batch solver. Therefore, despite its fast convergence it still requires many rounds of communication to compute a solution making it difficult to apply to real-time applications. Future work should include extending MESA to operate incrementally.

REFERENCES

[1] D. Drew, "Multi-agent systems for search and rescue applications," *Current Robotics Reports*, vol. 2, pp. 189–200, 2021.

[2] NASA Jet Propulsion Laboratory, "Cooperative Autonomous Distributed Robotic Exploration (CADRE)," https://www.jpl.nasa.gov/missions/cadre, accessed on 2023-08-24.

[3] M. Schuster, M. Müller, S. Brunner, H. Lehner, P. Lehner, R. Sakagami, A. Dömel, L. Meyer, B. Vodermayer, E. Giubilato, M. Vayugundla, J. Reill, F. Steidle, I. von Bargen, K. Bussmann, R. Belder, P. Lutz, W. Stürzl, M. Smíšek, M. Maier, S. Stoneman, A. Prince, B. Rebele, M. Durner, E. Staudinger, S. Zhang, R. Pöhlmann, E. Bischoff, C. Braun, S. Schröder, E. Dietz, S. Frohmann, A. Börner, H. Hübers, B. Foing, R. Triebel, A. Albu-Schäffer, and A. Wedler, "The ARCHES space-analogue demonstration mission: Towards heterogeneous teams of autonomous robots for collaborative scientific sampling in planetary exploration," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 4, pp. 5315–5322, 2020.

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. on Robotics (TRO)*, vol. 32, no. 6, pp. 1309–1332, 2016.

[5] T. Halsted, O. Shorinwa, J. Yu, and M. Schwager, "A survey of distributed optimization methods for multi-robot systems," arXiv preprint, arXiv:2103.12840 [cs.RO], 2021.

[6] S. Choudhary, L. Carlone, H. Christensen, and F. Dellaert, "Exactly sparse memory efficient SLAM using the multi-block alternating direction method of multipliers," in *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Hamburg, DE, Oct. 2015, pp. 1349–1356.

[7] C. Chen, B. He, Y. Ye, and X. Yuan, "The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent," *Mathematical Programming*, vol. 155, no. 1-2, pp. 57–79, 2016.

[8] A. Cunningham, M. Paluri, and F. Dellaert, "DDF-SAM: Fully distributed slam using constrained factor graphs," in *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Taipei, TW, Oct. 2010, pp. 3025–3030.

[9] A. Cunningham, V. Indelman, and F. Dellaert, "DDF-SAM 2.0: Consistent distributed smoothing and mapping," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, DE, May 2013, pp. 5220–5227.

[10] R. Murai, J. Ortiz, S. Saeedi, P. Kelly, and A. Davison, "A robot web for distributed many-device localisation," arXiv preprint, arXiv:2202.03314[cs.RO], 2022.

[11] K. Murphy, Y. Weiss, and M. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proc. Fifteenth Conference on Uncertainty in Artificial Intelligence*, Stockholm, SE, 1999, p. 467–475.

[12] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. Christensen, and F. Dellaert, "Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models," *Intl. J. of Robotics Research (IJRR)*, vol. 36, no. 12, pp. 1286–1311, 2017.

[13] 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 *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Seattle, USA, 2015, pp. 4597–4604.

[14] E. Cristofalo, E. Montijano, and M. Schwager, "Geod: Consensus-based geodesic distributed pose graph optimization," arXiv preprint, arXiv:2010.00156 [cs.RO], 2020.

[15] Y. Tian, K. Khosoussi, D. Rosen, and J. How, "Distributed certifiably correct pose-graph optimization," *IEEE Trans. on Robotics (TRO)*, vol. 37, no. 6, pp. 2137–2156, 2021.

[16] Y. Tian, A. Koppel, A. Bedi, and J. How, "Asynchronous and parallel distributed pose graph optimization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 4, pp. 5819–5826, 2020.

[17] G. Mateos, J. Bazerque, and G. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262–5276, 2010.

[18] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[19] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[20] O. Shorinwa, T. Halsted, and M. Schwager, "Scalable distributed optimization with separable variables in multi-agent networks," in *Proc. American Control Conference (ACC)*, Denver, USA, Jul. 2020, pp. 3619–3626.

[21] E. Wei, "Distributed optimization and market analysis of networked systems," PhD thesis, Massachusetts Institute of Technology, Boston, MA, September 2014.

[22] E. Wei and A. Ozdaglar, "On the O(1/k) convergence of asynchronous distributed alternating direction method of multipliers," in *Proc. IEEE Global Conference on Signal and Information Processing*, Austin, USA, 2013, pp. 551–554.

[23] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations and Trends in Robotics (FNT)*, vol. 6, no. 1-2, pp. 1–139, 2017.

[24] M. Hong, Z. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 337–364, 2016.

[25] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Institute of Technology, Technical Report, 2012.

[26] A. Haarbach, T. Birdal, and S. Ilic, "Survey of higher order rigid body motion interpolation methods for keyframe animation and continuous-time trajectory estimation," in *Proc. International Conference on 3D Vision (3DV)*, Verona, IT, 2018, pp. 381–389.

[27] A. Kovnatsky, K. Glashoff, and M. Bronstein, "MADMM: a generic algorithm for non-smooth optimization on manifolds," in *Proc. Eur. Conf. on Computer Vision (ECCV)*, Amsterdam, NL, 2016, pp. 680–696.

[28] J. Solà, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," arXiv, 1812.01537 [cs.RO], 2021.

[29] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Fast incremental smoothing and mapping with efficient data association," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Rome, Italy, Apr. 2007, pp. 1670–1677.