Connectivity Maintenance for Multi-Robot Systems Under Motion and Sensing Uncertainties Using Distributed ADMM-based Trajectory Planning

Akshay Shetty, Derek Knowles and Grace Xingxin Gao

Abstract—Inter-robot communication enables multi-robot systems to coordinate and execute complex missions efficiently. Thus, maintaining connectivity of the communication network between robots is essential for many multi-robot systems. In this paper, we present a trajectory planner for connectivity maintenance of a multi-robot system. We first define a weighted undirected graph to represent the connectivity of the system. Unlike previous connectivity maintenance works, we explicitly account for robot motion and sensing uncertainties while formulating the graph edge weights. These uncertainties result in uncertain robot positions which directly affect the connectivity of the system. Next, the algebraic connectivity of the weighted undirected graph is maintained above a specified lower limit using a trajectory planner based on a distributed alternating direction method of multipliers (ADMM) framework. Here we derive an approximation for the Hessian matrices required within the ADMM optimization step to reduce the computational load. Finally, simulation results are presented to statistically validate the connectivity maintenance of our trajectory planner.

Index Terms—multi-robot systems, global connectivity maintenance, motion and sensing uncertainties, distributed trajectory planning, alternating direction method of multipliers (ADMM)

I. INTRODUCTION

There has been growing interest in multi-robot systems for exploration, target tracking, formation control, and cooperative manipulation [1]. Multi-robot systems typically depend on inter-robot communication which enables them to execute complex missions efficiently. Inter-robot communication also adds resilience to malicious attacks [2] and single robot failures [3]. Thus, maintaining connectivity between robots is often a requirement for multi-robot systems.

The topic of connectivity maintenance for multi-robot systems has been widely addressed in literature. The general approach is to synthesize control inputs for each robot in the system such that either local connectivity or global connectivity of the system is maintained [4]. Local connectivity maintenance (LCM) methods focus on keeping the initial topology of connections within the multi-robot system. Thus, if two robots are initially connected, the synthesized control inputs for these robots maintain their connection throughout the mission. LCM methods typically consist of relatively simple computations since the control inputs for each robot depend only on local information of the robots to which it is connected. However, the freedom of motion for each robot is restricted since initial connections between robots are not allowed to be broken. Global connectivity maintenance (GCM) methods allow individual connections to break as long as there exists a (potentially multi-hop) communication path between two robots in the system [5]–[12]. Thus, each robot is afforded a greater freedom of motion compared to LCM methods.

A limitation of the previous connectivity maintenance works is that they do not explicitly account for robot motion and sensing uncertainties in their theoretical formulation. Presence of motion uncertainty causes a robot to deviate from the trajectory desired by the synthesized control inputs. Additionally, sensing uncertainties result in a robot obtaining a noisy estimate of its own position and its neighbors’ positions. These uncertainties, which are inherent in practical robots [13], consequently result in uncertainty in the robot positions and directly affects the system connectivity. Thus, it is important to explicitly account for robot motion and sensing uncertainties while designing connectivity maintenance methods.

Additionally, the majority of previous connectivity maintenance works use a simplified single integrator model to represent the robot motion [5]–[11]. This motion model assumes that the robot can instantaneously change its direction of motion and move towards a desired position for connectivity maintenance. Thus, these works derive control inputs in a myopic fashion, i.e., only for the current time instant. However, for most practical robots, such as unmanned aerial vehicles (UAVs), the direction of motion is not instantaneously changeable. The robot trajectory depends on additional quantities (such as previous velocities [14]) and hence the robot might not be able to change its direction of motion instantaneously. Thus, for connectivity maintenance, it is important to derive control inputs in a non-myopic fashion by considering the trajectory of the robot over multiple future time instants.

In this paper, we present a trajectory planning algorithm that maintains the global connectivity of a multi-robot system while accounting for motion and sensing uncertainties. The main contributions of this paper are listed as follows:

1) We define a weighted undirected graph to represent a multi-robot system with uncertain robot positions. Here the uncertainty in the robot positions are obtained from the robot motion and sensing uncertainties. We show that the algebraic connectivity (explained in Section III-B) of this graph is a probabilistic lower-bound for the true algebraic connectivity of the system.
2) We use the algebraic connectivity of the proposed weighted undirected graph to design a distributed trajectory planning algorithm based on an alternating direction method of multipliers (ADMM) framework [15]. Here we use a general linear motion model for the robots that does not necessarily assume an instantaneously changeable direction of motion. Our algorithm plans trajectories for the multi-robot system in a non-myopic fashion while maintaining the algebraic connectivity of the proposed graph above a specified lower limit.

3) We analyze the computational load of the optimization step within the ADMM framework. An approximation is derived for the required Hessian matrices which significantly reduces the computational load.

4) We simulate multi-UAV missions to statistically validate the global connectivity maintenance of our trajectory planner. We show the convergence of our algorithm and evaluate its performance under real-time constraints.

The remainder of the paper is organized as follows. We begin by discussing related connectivity maintenance works in Section II. In Section III, we formulate the connectivity maintenance and trajectory planning problems for this paper. Section IV defines the weighted undirected graph proposed for uncertain robot positions, which we use in our trajectory planning algorithm detailed in Section V. We present our simulation results in Section VI and conclude in Section VII.

II. RELATED WORK: CONNECTIVITY MAINTENANCE

GCM is widely addressed in literature and common methods are based on branching trees, $k$-connectivity, or algebraic connectivity. In the branching tree method, multiple robots are initially clumped together, one robot moves away from the group to the edge of the connectivity range, and then the preceding robots follow that pattern extending the communication branch to reach a desired goal [16], [17]. The main drawback of the branching tree method is that only a small subset of the robots can move at a given time which limits the amount of area that can be covered by the network. $k$-connectivity GCM methods design controllers that maintain a set level of $k$-connectivity at all times [18]. A robot network is $k$-connected if the network remains connected if fewer than $k$ robots are removed from the network.

Finally, a third group of GCM methods represent the multi-robot system as a weighted undirected graph and use its algebraic connectivity as an indicator of the system connectivity. The algebraic connectivity is defined as the second smallest eigenvalue of the graph Laplacian matrix, as discussed later in Section III-B in [6]. The authors present a decentralized power iteration algorithm for each robot to estimate the algebraic connectivity. This estimate is then used to design a decentralized gradient-based controller for GCM. [7] builds on the method in [6] by defining a decentralized estimation procedure for algebraic connectivity that is formally guaranteed to be stable. Given the estimation error boundedness, they prove that the proposed control law guarantees GCM if the control parameters are chosen appropriately. Further, in [8] the authors extend their previous work of [7] by accounting for an additional (bounded) control input for each robot. [9] extends on [7] by explicitly accounting for additional inter-robot constraints such as a desired relative distance and collision avoidance. In [10], the authors design a GCM method to account for robots with bounded control inputs. They present a theoretical analysis to evaluate the robustness of the controller to bounded errors in estimate of the algebraic connectivity. However, the estimation error bound is heuristically obtained without explicitly accounting for sources of uncertainty such as robot motion and sensing uncertainties.

Another common approach for GCM is to design optimization-based methods without estimating the value of the algebraic connectivity itself. In [5], the authors find optimal positions to maximize the algebraic connectivity and then derive control inputs for a multi-robot system using a decentralized potential field-based method. In [12], the authors present a differential game-theoretic formulation for maximizing the algebraic connectivity in the presence of a malicious jammer. [11] uses control barrier functions to integrate a GCM requirement with an additional control input for each robot.

While many of these methods provide GCM guarantees [5]–[12], [18], they do not explicitly account for robot motion and sensing uncertainties and a majority of them assume a simplified single integrator robot motion model. Thus, in their simulation/experimental setups they make simplifications; for instance, assuming perfect sensing information such as perfect localization measurements and/or using slow-moving robots that can be reasonably modeled as single integrator systems. However, practical robots are typically represented by high-fidelity motion models and use state estimation filters to estimate their positions under motion and sensing uncertainties [13]. In this paper, we primarily address these limitations in previous methods. In the remainder of the paper, we simply refer to global connectivity as connectivity.

III. PROBLEM FORMULATION

A. Robot description

For each robot $i$ in a multi-robot system with $N$ robots, we consider linear discrete-time motion and sensing models:

\[ x_{i,t} = A_{i,t-1} x_{i,t-1} + B_{i,t-1} u_{i,t-1} + w_{i,t}, \]  
\[ z_{i,t} = C_{i,t} x_{i,t} + v_{i,t}, \]

where $t$ is the time instant, $x_{i,t}$ is the state vector, $u_{i,t}$ is the input vector, $z_{i,t}$ is the sensed measurement vector, $A_{i,t}$ is state transition matrix, $B_{i,t}$ is the control-input matrix, $C_{i,t}$ is the system measurement matrix, $w_{i,t} \sim N(0, Q_{i,t})$ is the motion model error and $v_{i,t} \sim N(0, R_{i,t})$ is the sensing model error. Note that throughout the paper we use bold font to represent vectors, and we use the notation $N[\mu, \Gamma]$ to represent a Gaussian-distributed vector with mean $\mu$ and covariance $\Gamma$.

We assume that each robot implements a Kalman Filter (KF) [13] on board to obtain an estimate of its state $\hat{x}_i$. The prediction step of the KF is performed as:

\[ \hat{x}_{i,t} = A_{i,t-1} \hat{x}_{i,t-1} + B_{i,t-1} u_{i,t-1}, \]
\[ \hat{P}_{i,t} = A_{i,t-1} \hat{P}_{i,t-1} A_{i,t-1}^\top + Q_{i,t}, \]

where $P_{i,t}$ is the state estimation covariance matrix such that $\hat{x}_{i,t} \sim N[\hat{x}_{i,t}, P_{i,t}]$. The KF correction step is performed as:
summarized as [19]:
Furthermore, the belief dynamics for the robot can be obtained as
\[
\begin{align*}
P_{i,t} &= \bar{P}_{i,t} - L_{i,t}C_{i,t}\bar{P}_{i,t}, \\
\bar{P}_{i,t} &= \sum_{j \neq i} A_{ij,t}C_{j,t}\bar{P}_{j,t},
\end{align*}
\]
where \( L_{i,t} \) is the Kalman gain. Here the second term in (6) is referred to as the innovation term and is distributed according to \( \mathcal{N}[0, L_{i,t}C_{i,t}\bar{P}_{i,t}] \).
Thus, each robot can be represented in the belief space with the belief vector defined as [19]:
\[
b_{i,t} = \begin{bmatrix} \bar{x}_{i,t} \\ \text{vec}[P_{i,t}] \end{bmatrix},
\]
where \( \text{vec}[P_{i,t}] \) denotes a column vector containing the elements of the upper triangle portion of \( P_{i,t} \) (element in first column, appended by elements in second column, and so on). Furthermore, the belief dynamics for the robot can be summarized as [19]:
\[
b_{i,t+1} = g_i[b_{i,t}, u_{i,t}] + M_i[b_{i,t}, u_{i,t}]m_{i,t},
\]
where:
\[
g_i[b_{i,t}, u_{i,t}] = \begin{bmatrix} \bar{x}_{i,t} \\ \text{vec}[P_{i,t} - L_{i,t}C_{i,t}\bar{P}_{i,t}] \end{bmatrix},
\]
\[
M_i[b_{i,t}, u_{i,t}] = \begin{bmatrix} 0 \\ \sqrt{L_{i,t}C_{i,t}\bar{P}_{i,t}} \end{bmatrix},
\]
\[
m_{i,t} \sim \mathcal{N}[0, I],
\]
where \( I \) represents an identity matrix.

Given that we assume linear models in (1)-(2), the KF exactly represents the uncertainty in the true state as \( \bar{x}_{i,t} \sim \mathcal{N}[x_{i,t}, P_{i,t}] \) [13], and consequently the belief dynamics in (4) exactly captures the state uncertainty. While the belief dynamics can be derived for nonlinear models with an Extended Kalman Filter (EKF) (as done in [19]), the EKF only provides an approximation of the state uncertainty. Thus, designing the trajectory planner based on an approximation of the state uncertainty could lead to undesirable loss of connectivity. While the linear model in (1) is more restrictive (as opposed to a nonlinear model), it represents the motion of robotic systems more realistically [20], [21] compared to a single-integrator motion model assumed in a majority of related work (see Section IV). The linear sensing model in (2) is commonly used to represent measurements from on-board sensors, such as localization measurements from Global Navigation Satellite System (GNSS) receivers or cameras. For our simulations in Section VI we use a double-integrator motion model (state vector contains robot position and velocity; input vector contains accelerations) along with localization measurements.

**B. Connectivity maintenance**

Similar to most previous connectivity maintenance works [6]–[10], we assume a disk communication model. Thus, two robots are considered to be connected only if the distance between them is smaller than a specified communication range \( \Delta \). Let the multi-robot system be represented as an undirected graph, where each node represents a robot and each edge represents the connection between two robots. The adjacency matrix of the graph at any time-step \( t \) can be obtained as:

\[
A_{ij,t} = \begin{cases} 1 & 0 \leq l_{ij,t} \leq \Delta \\ 0 & l_{ij,t} > \Delta \end{cases}
\]
where \( A_{ij,t} \) is the \((i, j)\)th element of adjacency matrix \( A_t \) and \( l_{ij,t} \) is the distance between the robots. The distance \( l_{ij,t} \) can be computed as \( l_{ij,t} = ||p_{i,t} - p_{j,t}||_2 \), where \( p_i \) and \( p_j \) are the true positions of the robots and \( ||\cdot||_2 \) represents the L2-norm. Here we assume that the robot positions \( p_i \) are contained in the robot state vectors \( x_i \) (defined in (1)), which is generally true for most mobile robot systems such as UAVs.

Given the adjacency matrix, the degree of each node can be obtained as \( d_i = \sum_{j=1}^{N} A_{ij,t} \). The vector of node degrees \( d_i \) is then used to define the degree matrix \( D_t \) of the graph as \( D_t = \text{diag}(d_i) \). Using matrices \( A_t \) and \( D_t \), the Laplacian matrix \( L_t \) of the graph is defined as \( L_t = D_t - A_t \) [22]. The second-smallest eigenvalue of the Laplacian matrix, \( \lambda_2^L \), is defined as the algebraic connectivity of the graph, which is a commonly used indicator for connectivity as discussed in Section II. The value of \( \lambda_2^L \) varies from zero (if the graph is disconnected) to the number nodes in the graph (if the graph is fully connected), i.e., \( 0 \leq \lambda_2^L \leq N \). Thus, \( \lambda_2^L > 0 \) implies that multi-robot system is connected, i.e., there exists a (potentially multi-hop) communication path between any two robots.

Note that the value of \( \lambda_2^L \) depends on the robot positions \( p_i \) which are contained in the state vectors \( x_i \). Since the state vectors are stochastic in nature (as discussed in Section III-A), the value of \( \lambda_2^L \) is also stochastic. Thus, given a desired lower limit \( \epsilon \) for the algebraic connectivity of the system, we state the following connectivity maintenance requirement for our trajectory planning algorithm:

\[
\Pr[\lambda_2^L > \epsilon] \geq 1 - \delta \forall t \in [0, T],
\]

i.e., the planner should maintain \( \lambda_2^L \) above \( \epsilon \) with a minimum probability value of \((1 - \delta)\) for the planning time horizon \( T \). We specify the values of \( \epsilon \) and \( \delta \) chosen for our simulations later in Section VI-A.

**C. Trajectory planning**

The objective of the trajectory planner is to plan nominal trajectories for each robot such that they perform local tasks while maintaining connectivity within the multi-robot system. Here the local tasks can represent objectives such as tracking a target, minimizing the control input effort, avoiding collisions, reaching a desired position for exploration, coverage or formation control, etc. We assume that the following information is available to each robot in the system:

1) The number of robots in the system \( N \), and the initial beliefs of all robots, i.e., \( b_{i,\text{init}} \forall i \in [1, N] \). As defined in (8), the initial belief vector consists of the initial state estimate and the initial estimation covariance.
2) The belief dynamics associated with all robots in the system as defined in (9).
3) The cost functions representing the local tasks for all robots in the system, i.e., \( J_{i,t}[b_{i,t}, u_{i,t}] \forall i \in [1, N], \forall t \in [0, T] \).

The nominal trajectory for each robot \( i \) can be represented as a series of nominal beliefs and nominal control inputs \( \{b_{i,0}, u_{i,0}, \ldots, b_{i,T-1}, u_{i,T-1}, b_{i,T}\} \) [19], such that:
\[ \mathbf{b}_{i,t+1} = g_i[\mathbf{b}_{i,t}, \mathbf{u}_{i,t}] \quad \forall t \in [0, T-1]. \]  

We define a concatenated nominal input matrix \( \hat{U} \), consisting of nominal input vectors of all robots in the system, over the entire planning time horizon:

\[ \hat{U} = \begin{bmatrix} \mathbf{u}_{1,0} & \cdots & \mathbf{u}_{1,T-1} \\ \vdots & \ddots & \vdots \\ \mathbf{u}_{N,0} & \cdots & \mathbf{u}_{N,T-1} \end{bmatrix}. \]  

Note that given the initial beliefs \( \mathbf{b}_{i,\text{init}} \forall i \in [1, N] \), it is sufficient to represent the nominal trajectories for the multi-robot system by \( \hat{U} \) since the nominal beliefs for each robot can be calculated recursively using (12). Thus, in the remainder of the paper we simply refer to the concatenated nominal input matrix \( \hat{U} \) as the nominal trajectory for the multi-robot system.

Finally, the overall objective of the planner is stated as:

\[ \hat{U} = \arg\min_{\mathbf{u}} \sum_{t=0}^{T-1} \sum_{i=1}^{N} J_{i,t}[\mathbf{b}_{i,t}, \mathbf{u}_{i,t}], \]

subject to:

\[ \Pr[\lambda_{E}^t > \epsilon] \geq 1 - \delta \quad \forall t \in [0, T], \]

\[ \mathbf{b}_{i,0} = \mathbf{b}_{i,\text{init}} \forall i \in [1, N], \]

\[ \mathbf{b}_{i,t+1} = g_i[\mathbf{b}_{i,t}, \mathbf{u}_{i,t}] \quad \forall i \in [1, N], \forall t \in [0, T-1], \]

where the first constraint is the connectivity maintenance requirement stated in (11). Common examples for the cost functions \( J_{i,t} \) include distance to a desired position, proximity to a unsafe set of states (such as collisions), amount of required control input, etc. In order to solve the above planning problem, we first define a weighted undirected graph in Section IV that accounts for uncertain robot positions, and then propose a distributed ADMM-based trajectory planning algorithm in Section V.

IV. WEIGHTED UNDIRECTED GRAPH FOR UNCERTAIN ROBOT POSITIONS

In order to address the connectivity maintenance requirement from (11), we first define a weighted undirected graph that accounts for uncertain robot positions arising due to the presence of motion and sensing uncertainties. The algebraic connectivity of this graph is then used in our trajectory planning algorithm in Section V. Since the graph definition is applicable for any time instant \( t \in [0, T] \), for simplicity we omit the time notations in this section. As mentioned in Section III-B we assume that the robot positions \( \mathbf{p}_i \) are contained in the state vectors \( \mathbf{x}_i \). Thus, given that the state vector is distributed as \( \mathbf{x}_i \sim \mathcal{N}[\hat{\mathbf{x}}_i, \Sigma_i] \) (see Section III-A), the robot positions can be represented as \( \mathbf{p}_i \sim \mathcal{N}[\hat{\mathbf{p}}_i, \Sigma_i] \). Here \( \hat{\mathbf{p}}_i \) is the estimated position contained in the estimated state \( \hat{\mathbf{x}}_i \), and the covariance matrix \( \Sigma_i \) is a submatrix of \( \Sigma_i \).

We begin defining our weighted graph by considering a confidence ellipse \( E_i \) centered at \( \hat{\mathbf{p}}_i \) such that:

\[ \Pr[\mathbf{p}_i \in E_i] = 1 - \delta_{E}, \]

where \( \delta_{E} \) is a probability value that decides the size of the confidence ellipse. We derive the value for \( \delta_{E} \) used in our algorithm later in (25). Let \( \lambda_{E_i} \) represent the largest eigenvalue of the covariance matrix \( \Sigma_i \). Thus, the length of the semi-major axis of \( E_i \) is \( s \sqrt{\lambda_{E_i}} \), where \( s \) is a scalar factor that follows a chi-square distribution (23) based on the value of \( \delta_{E} \). We then define a circular region \( S_i \) centered at \( \hat{\mathbf{p}}_i \) with radius \( s \sqrt{\lambda_{E_i}} \). This circular region overbounds \( E_i \) and thus, contains \( \mathbf{p}_i \) with a probability greater than or equal to \( \delta_{E} \), i.e.:

\[ \Pr[\mathbf{p}_i \in S_i] \geq 1 - \delta_{E}. \]

We then define a distance measure between the boundaries of the overbounding circular regions of two robots \( i \) and \( j \) as:

\[ \hat{l}_{ij} = \|\hat{\mathbf{p}}_i - \hat{\mathbf{p}}_j\|_2 + s \sqrt{\lambda_{E_i}} + s \sqrt{\lambda_{E_j}}. \]

Fig. [1] illustrates the confidence ellipses, the overbounding circular regions and the distance measure between two robots.

Given the communication range of \( \Delta \) between two robots, we introduce a new parameter \( \Delta_0 \), such that \( 0 < \Delta_0 < \Delta \). Based on the edge weight defined in (9), we use \( \Delta_0 \) to define a non-binary edge weight between two robots \( i \) and \( j \) as:

\[ \bar{A}_{ij} = \begin{cases} 1 & \text{if } \hat{l}_{ij} \leq \Delta_0, \\ \frac{1}{2} + \frac{1}{2} \cos \left[ \pi \frac{(\hat{l}_{ij} - \Delta_0)}{\Delta - \Delta_0} \right] & \Delta_0 < \hat{l}_{ij} \leq \Delta, \\ 0 & \hat{l}_{ij} > \Delta. \end{cases} \]

Fig. [2] compares \( \bar{A}_{ij} \) with \( A_{ij} \) from (10). We then proceed to define the corresponding degree matrix \( \bar{D} = \text{diag}[d_i] \) with \( d_i = \sum_{j=1}^{n} \bar{A}_{ij} \) and the corresponding Laplacian matrix \( \bar{L} = \bar{D} - \bar{A} \). Finally, we use the algebraic connectivity of this weighted undirected graph \( \lambda_{E_i}^{\text{adj}} \) as an indicator for the connectivity of the system with uncertain robot positions.

Next, we proceed to derive the value of \( \delta_{E} \) required in (15).

We define the following events:
\[ E_1 : \ p_i \in S_i \ \forall \ i \in [1, N], \]
\[ E_2 : \ \bar{t}_{ij} \geq t_{ij} \ \forall \ \{i, j\} \in [1, N] \times [1, N] \ | \ i \neq j, \]
\[ E_3 : \ A_{ij} \leq \mathbb{A}_{ij} \ \forall \ \{i, j\} \in [1, N] \times [1, N] \ | \ i \neq j, \]
\[ E_4 : \ \lambda_2^E \geq \Lambda_2^E. \]

Here \( E_1 \) represents the event that all robot positions \( p_i \) lie within their corresponding circular regions. We assume that the true positions \( p_i \) of the robots in the system are independent of each other. Thus, using (16), we express the probability of event \( E_1 \) as:
\[
\Pr[E_1] = \prod_{i=1}^{N} \Pr[p_i \in S_i] \geq \prod_{i=1}^{N} (1 - \delta) = (1 - \delta)^N, \tag{19}
\]
where \( N \) is the number of robots in the system.

\( E_2 \) represents the event that the distance measures \( \bar{t}_{ij} \) between any two robots \( i \) and \( j \) will always be greater than or equal to the true distance \( t_{ij} \). We proceed to derive the probability of event \( E_2 \) as:
\[
\Pr[E_2] = \Pr[E_2 | E_1] \cdot \Pr[E_1] \geq \Pr[E_2 | E_1] = \Pr[E_1], \tag{20}
\]
where \( \Pr[E_2 | E_1] = 1 \) since for any two robots \( i \) and \( j \) if \( p_i \in S_i \) and \( p_j \in S_j \), then \( \bar{t}_{ij} \geq t_{ij} \) as shown in Fig. 1.

\( E_3 \) represents the event that the non-binary edge weight \( \Delta_{ij} \) from (18) is less than the edge weight \( A_{ij} \) from (10) for any two robots \( i \) and \( j \). Similar to (20), we derive the probability of event \( E_3 \) as:
\[
\Pr[E_3] = \Pr[E_3 | E_2] \cdot \Pr[E_2] + \Pr[E_3 | E_2'] \cdot \Pr[E_2'],
\]
\[
\geq \Pr[E_3 | E_2] = \Pr[E_2], \tag{21}
\]
where \( \Pr[E_3 | E_2] = 1 \) since for any two robots \( i \) and \( j \) if \( \bar{t}_{ij} \geq t_{ij} \), then \( \Delta_{ij} \leq A_{ij} \) as shown in Fig. 2.

Finally, \( E_4 \) is derived as:
\[
\Pr[E_4] = \Pr[E_4 | E_3] \cdot \Pr[E_3] \geq \Pr[E_4 | E_3] = \Pr[E_3], \tag{22}
\]
where \( \Pr[E_4 | E_3] = 1 \) since by definition the algebraic connectivity monotonically increases as the edge weights increase \([6, 9]\). Thus, from (19)-(22) we have:
\[
\Pr[\lambda_2^E \geq \Lambda_2^E] \geq (1 - \delta)^N, \tag{23}
\]
which shows that \( \lambda_2^E \) lower-bounds \( \Lambda_2^E \) with a minimum probability value of \( (1 - \delta)^N \). If the value of \( \lambda_2^E \) is maintained above the specified lower limit \( \epsilon \) from (14), i.e., if \( \lambda_2^E > \epsilon \), then from (23) we get:
\[
\Pr[\lambda_2^E > \epsilon] \geq (1 - \delta)^N. \tag{24}
\]

In order to satisfy the connectivity maintenance requirement described in (11), we set \( (1 - \delta)^N = 1 - \delta \), which finally gives us the following value for \( \delta \):
\[
\delta = 1 - (1 - \delta)^{1/N}, \tag{25}
\]
where \( \delta \) is the probability value representing a desired confidence level in (11). Thus, setting the value of \( \delta \) as shown in (25) and ensuring that \( \lambda_2^E \) is maintained above \( \epsilon \) results in satisfying the connectivity maintenance requirement from (11).

Note that the weighted undirected graph is a conservative representation of the system connectivity since it measures the connectivity based on overbounding circular regions \( S_i \) as opposed to true robot positions \( p_i \). Thus, while maintaining \( \lambda_2^E \) above \( \epsilon \) satisfies the connectivity maintenance requirement, it can result in restricting the mobility of the system. Here the amount of restriction depends on the size of the overbounding circular regions \( S_i \), which depend on the amount of uncertainty in the robot motion and sensing models in (1)-(2).

While in this paper we evaluate our algorithm in two-dimensions, the weighted undirected graph defined in this section can be directly extended to three-dimensions. In the three-dimensional case \( E_i \) in (15) would represent a confidence ellipsoid for robot \( i \) and \( S_i \) in (16) would represent the corresponding overlapping spherical region.

V. Trajectory Planning Algorithm

In this section, we present the details of our trajectory planning algorithm for solving the problem stated in (14). First, we define a connectivity cost function based on the algebraic connectivity \( \Lambda_2^E \) of the weighted undirected graph defined in Section IV. We incorporate this cost function with the cost in (14) to obtain a transformed planning problem. Next, we present a distributed ADMM setup in order to solve the transformed problem and plan nominal trajectories for the multi-robot system. We then describe the method used for performing the optimization step within the ADMM setup and analyze the complexity of its computational bottleneck. Finally, we present an approach to reduce the computational load of this optimization step by deriving an approximation for the required Hessian matrices. Later in Section VII we demonstrate how the proposed planning algorithm can be utilized for connectivity maintenance under real-time constraints.

A. Connectivity cost and transformed planning problem

As discussed earlier in Section IV, maintaining \( \lambda_2^E \) above the specified lower limit \( \epsilon \) enables us to satisfy the connectivity maintenance requirement described in (11). Thus, in order to maintain \( \Lambda_2^E \) above \( \epsilon \), we define a connectivity cost function that grows to infinity as \( \lambda_2^E \) approaches \( \epsilon \). Various cost functions with the above property have been proposed in related work \([7,9]\). For a distributed ADMM setup, it has been shown that the ADMM iteration complexity is inversely proportional to the algebraic connectivity of the system \([24]\). Thus, we choose to define the connectivity cost function for any time instant \( t \) as following:
\[
J_t^E = \frac{k_c}{\lambda_2^E - \epsilon}, \tag{26}
\]
where \( k_c \) is a parameter that determines the magnitude of the cost function. In order to incorporate the connectivity cost function with (14), we update original planning problem as follows:
\[ \bar{U} = \arg \min \left( \sum_{t=0}^{T} \sum_{i=1}^{N} J_{i,t} \right) \]

\[ \bar{U} = \arg \min \sum_{t=0}^{T} \sum_{i=1}^{N} J_{i,t}, \]  

where:
\[ J_{i,t} = J_{i,t}(\tilde{b}_{i,t}, \tilde{\mathbf{u}}_{i,t}) + \frac{1}{N} \tilde{J}_{i}. \]

Thus, we write the transformed planning problem as:
\[ \bar{U} = \arg \min \sum_{t=0}^{T} \sum_{i=1}^{N} \tilde{J}_{i,t} \]

subject to:
\[ \tilde{b}_{i,0} = \mathbf{b}_{i,\text{init}} \forall i \in [1, N], \]
\[ \tilde{b}_{i,t+1} = g_i[\tilde{b}_{i,t}, \tilde{\mathbf{u}}_{i,t}] \forall i \in [1, N], \forall t \in [0, T - 1]. \]

The difference between (14) and the transformed planning problem is that the connectivity maintenance constraint has been incorporated in the cost function. The main reason behind transforming the planning problem from (14) to (29) is to allow us to use existing optimization tools [19] within the ADMM setup, as will be discussed in Section V-C. Note that \( \tilde{J}_{i} \) in (26) and consequently \( \tilde{J}_{i,t} \) in (29) are undefined for \( \tilde{\lambda}_i^0 \leq \epsilon \). In order to avoid numerical instability, we use a line-search method (discussed later in (32)) that ensures \( \tilde{\lambda}_i^0 > \epsilon \) for the initial guess of the trajectory optimization. If \( \tilde{\lambda}_i^0 \) gets close to \( \epsilon \) during the optimization process, the nature of the cost function in (26) results in gradients that drive \( \tilde{\lambda}_i^0 \) away from \( \epsilon \).

B. Distributed ADMM setup

In order to solve the transformed planning problem from (29), we implement a distributed ADMM setup [15] that iteratively plans nominal trajectories for the multi-robot system. In each ADMM iteration, each robot optimizes only a subset of the robot trajectories in order to reduce the computational load of the optimization step. The optimized trajectories are then communicated with the rest of the system. After the communication step, each robot updates its local ADMM consensus and dual variables before moving on to the next ADMM iteration. When the stopping criteria is satisfied, the last updated local ADMM consensus variable is used as the planned nominal trajectories for the multi-robot system.

Each robot begins by generating an initial guess for the nominal trajectories of the multi-robot system \( \bar{U}_{(i,1)} \), where the superscript denotes that the variable is stored locally on robot \( i \) and is for the first ADMM iteration. The initial guess is typically generated based on the local tasks for each robot. We assume that the process used to generate the initial guess maintains \( \tilde{\lambda}_i^0 \) above \( \epsilon \forall t \in [0, T] \). Later in Section VI-A we describe our method for obtaining the initial guess when the local task for each robot involves reaching a desired position. Once the initial guess has been generated, the robot proceeds to initialize its local copy of the consensus variable as \( \bar{U}_{(i,1)} = \bar{U}_{(1,1)} \). The ADMM dual variable \( \lambda_{(i,1)} \) is initialized as a zero matrix.

Next, the robot begins the ADMM iterations. In each ADMM iteration \( k \), the robot first obtains a subset \( \mathcal{V}_{(i,k)} \) containing indices of the robot trajectories to optimize. Different strategies can be deployed for obtaining \( \mathcal{V}_{(i,k)} \). For example, setting \( \mathcal{V}_{(i,k)} = \{ i \} \) results in a greedy optimization where the robot optimizes its own trajectory; setting \( \mathcal{V}_{(i,k)} \) to contain neighboring robot indices focuses more on the local connectivity rather than the global connectivity of the system. In our algorithm, we obtain \( \mathcal{V}_{(i,k)} \) such that it contains \( i \) and cycles through the indices of the other \( (N - 1) \) robots. As mentioned in Section IV-C, we assume that each robot knows there are \( N \) number of robots in the system. Let \( \eta \) represent the number of elements in \( \mathcal{V}_{(i,k)} \). Table I shows an example of the subsets \( \mathcal{V}_{(i,k)} \) for four ADMM iterations in a system with four robots and with \( \eta = 3 \). We observe that this strategy for obtaining \( \mathcal{V}_{(i,k)} \) avoids the problem of greedy optimizations and eventually results in nominal trajectories for the system with lower overall costs as shown in Section VI. The value of \( \eta \) can be chosen based on the computation power of each robot. While choosing a larger \( \eta \) would result in lower overall costs, the computational load would be higher.

| k | \( \mathcal{V}_{(1,k)} \) | \( \mathcal{V}_{(2,k)} \) | \( \mathcal{V}_{(3,k)} \) | \( \mathcal{V}_{(4,k)} \) |
|---|---|---|---|---|
| 1 | \{1, 2, 3\} | \{2, 3, 4\} | \{3, 4, 1\} | \{4, 1, 2\} |
| 2 | \{1, 3, 4\} | \{2, 4, 1\} | \{3, 1, 2\} | \{4, 2, 3\} |
| 3 | \{1, 4, 2\} | \{1, 2, 0\} | \{2, 3, 4\} | \{3, 4, 1\} | \{4, 1, 2\} |

TABLE I: Example of subsets \( \mathcal{V} \) for a system with four robots, where \( \eta = 3 \) and up to \( k = 4 \) ADMM iterations are considered.

Once the subset \( \mathcal{V}_{(i,k)} \) has been obtained, the robot performs the optimization step. In this step, we first initialize \( \tilde{U}_{(i,k+1)} = U_{(i,k)} \) and then only update the trajectories for robots in subset \( \mathcal{V}_{(i,k)} \). Based on the cost function in (29), the robot optimizes the following augmented cost [15] to obtain optimized trajectories for robots in \( \mathcal{V}_{(i,k)} \):

\[ \tilde{U}_{(i,k+1)} \mathcal{V}_{(i,k)} = \arg \min \tilde{U}_{(i,k)} \sum_{t=0}^{T} \sum_{j \in \mathcal{V}_{(i,k)}} \tilde{J}_{j,t} \]

\[ + \mathbf{y}^{(i,k)}_{(i,k)} \left( \tilde{\mathbf{u}}_{(i,k),t} - \tilde{\mathbf{u}}_{(i,k),t} \right) \]

\[ + \left( \rho / 2 \right) \left\| \tilde{\mathbf{u}}_{(i,k),t} - \tilde{\mathbf{u}}_{(i,k),t} \right\|_2^2 \]  

subject to:
\[ \tilde{b}_{i,0} = \mathbf{b}_{i,\text{init}} \forall i \in [1, N], \]
\[ \tilde{b}_{i,t+1} = g_i[\tilde{b}_{i,t}, \tilde{\mathbf{u}}_{i,t}] \forall i \in [1, N], \forall t \in [0, T - 1], \]  

where \( \rho > 0 \) is the ADMM penalty weight, \( \tilde{\mathbf{u}} \) and \( \tilde{\mathbf{u}}_{(i,k)} \) represent the corresponding vectors from matrices \( \tilde{U} \) and \( \tilde{U}_{(i,k)} \) respectively, and \( \mathbf{y}^{(i,k)}_{(i,k)} \) represents the corresponding vector from dual variable matrix \( \mathbf{Y}^{(i,k)} \). We discuss the method used to solve this optimization step later in Section V-C. Here the cost accompanied by \( \rho \) is commonly referred to as the consensus constraint [15]. Since each robot optimizes trajectories for a subset of robots, each robot might have a different idea of the planned nominal trajectories for the complete system. Thus, enforcing this constraint allows the robots reach a consensus on the planned nominal trajectories.

After the optimization step, the robot proceeds to the communication step where each robot \( i \) shares the optimized...
trajectories $\bar{U}^{(i,k+1)}_{(i,k)}$ obtained using (30). In this step, each robot receives the optimized trajectories from all other robots in the system, potentially via multi-hop communication. Note that our planner is distributed since each robot optimizes with respect to a subset of trajectories, as opposed to trajectories from all robots in the system. However, the communication architecture is centralized since each robot shares information with all other robots in the system. Thus, the communication load does not scale well with the number of robots and could lead to delays during execution. Later in Section VI we account for a communication delay while evaluating our planner under real-time constraints.

Once the communication step is complete, each robot $i$ receives the optimized trajectories from all other robots. Note that the trajectory for each robot has been optimized $\eta$ times across the system. For example, in Table I at ADMM iteration $k = 3$, the trajectory for robot 4 was optimized by robots 1, 3, and 4, i.e., $\eta = 3$ times. Thus, based on the consensus update step in (15), the robot calculates an average optimized trajectory for each robot $j$ as follows:

$$\bar{U}^{(i,k+1)}_j = \frac{1}{\eta} \sum_{l=1}^{\eta} U^{(i,k+1)}_j \cdot \mathbf{1}_{Y^{(i,k)}}[j],$$  \hspace{1cm} (31)

where $\mathbf{1}_{Y^{(i,k)}}[j]$ is an indicator function equal to 1 if $j \in Y^{(i,k)}$ and equal to 0 otherwise. Note that in (31) the robot does not need to keep track of which robot it received the optimized trajectory $\bar{U}^{(i,k+1)}_j$ from during the communication step.

After the averaging step, it is possible that $\bar{U}^{(i,k+1)}$ might result in a trajectory for the multi-robot system that does not maintain $\bar{L}_t^{\rho}$ above the specified lower limit of $\epsilon$. Thus, in order to ensure that the consensus variable $\bar{U}$ always results in trajectories that maintain $\bar{L}_t^{\rho} > \epsilon$, we use a line search algorithm [25] to update $\bar{U}$. We limit the change to $\bar{U}$ between iterations as follows:

$$\bar{U}^{(i,k+1)} = \bar{U}^{(i,k)} + \beta \cdot (\bar{U}^{(i,k+1)} - \bar{U}^{(i,k)}),$$  \hspace{1cm} (32)

where $\beta$ is a parameter that determines the amount of change in $\bar{U}$. We begin with $\beta = 1$ and check if the corresponding $\bar{U}^{(i,k+1)}$ results in trajectories that maintain $\bar{L}_t^{\rho} > \epsilon$. If $\bar{L}_t^{\rho}$ is not maintained above $\epsilon$, we reduce $\beta$ by a factor $\gamma$ as: $\beta = \gamma \cdot \beta$, where $0 < \gamma < 1$. Then we calculate the new $\bar{U}^{(i,k+1)}$ using (32) and repeat the process until $\bar{U}^{(i,k+1)}$ results in trajectories that maintain $\bar{L}_t^{\rho} > \epsilon$. Thus, if $\bar{U}^{(i,k+1)}$ obtained from the averaging step does not maintain $\bar{L}_t^{\rho} > \epsilon$, the line search algorithm gradually scales the trajectory to $\bar{U}^{(i,k)}$.

Finally, each robot $i$ updates its ADMM dual variable as follows:

$$Y^{(i,k+1)} = Y^{(i,k)} + \rho \cdot (\bar{U}^{(i,k+1)} - \bar{U}^{(i,k+1)}),$$  \hspace{1cm} (33)

where $\rho$ is the ADMM penalty weight defined in (30).

Before beginning the next ADMM iteration, the robot checks if the stopping criterion has been satisfied. The stopping criteria can be either convergence-based or time-based. Later in Section VII we evaluate our planner under a time-based stopping criterion. If the stopping criteria is satisfied, the robots set the last updated value of $\bar{U}$ as the planned nominal trajectories for the multi-robot system. Since $\bar{U}$ always results in trajectories that maintain $\bar{L}_t^{\rho} > \epsilon$, the output from our trajectory planning algorithm always results in trajectories that maintain $\bar{L}_t^{\rho} > \epsilon$. Thus, our algorithm satisfies the connectivity maintenance requirement from (11). Algorithm I summarizes our trajectory planning algorithm.

C. ADMM trajectory optimization and complexity analysis

In order to obtain the optimized nominal trajectories $\bar{U}^{(i,k+1)}_{(i,k)}$ in (30), we use the belief-space iterative Linear Quadratic Gaussian (belief-space iLQG) method [19]. Since the analysis in the remainder of this section is applicable for a general subset of robot trajectories, we simply represent $Y^{(i,k)}$ as $\mathcal{V}$. We first extend [2] to define concatenated belief dynamics for the subset $\mathcal{V}$ as follows:

$$b_{\mathcal{V},t+1} = g_{\mathcal{V}}[b_{\mathcal{V},t}, u_{\mathcal{V},t}, t] + M_{\mathcal{V}}[b_{\mathcal{V},t}, u_{\mathcal{V},t}] m_{\mathcal{V},t},$$  \hspace{1cm} (34)

where $b_{\mathcal{V}}$ is the concatenated belief vector of robots in the subset $\mathcal{V}$. Next, similar to (12), the concatenated nominal trajectory for the subset $\mathcal{V}$ is represented as $(b_{\mathcal{V},0}, u_{\mathcal{V},0}, \ldots, b_{\mathcal{V},T-1}, u_{\mathcal{V},T-1}, b_{\mathcal{V},T})$, such that:

$$b_{\mathcal{V},t+1} = g_{\mathcal{V}}[b_{\mathcal{V},t}, u_{\mathcal{V},t}] \quad \forall \ t \in [0, T - 1].$$  \hspace{1cm} (35)

Additionally, we rewrite the ADMM optimization step in (30) as:

$$\bar{U}^{(i,k+1)}_{\mathcal{V}} = \arg\min_{U_{\mathcal{V}}} \sum_{t=0}^{T} c_t[b_{\mathcal{V},t}, u_{\mathcal{V},t}],$$

subject to:

$$b_{\mathcal{V},0} = b_{\mathcal{V},\text{init}},$$

$$b_{\mathcal{V},t+1} = g_{\mathcal{V}}[b_{\mathcal{V},t}, u_{\mathcal{V},t}] \quad \forall \ t \in [0, T - 1],$$
where \(c_t\) is the cost at time instant \(t\). While \(c_t\) depends on the belief and input vectors of the entire multi-robot system, for simplicity we write \(c_t\) to be a function of only \(\mathbf{b}_V\) and \(\mathbf{u}_V\), since only the trajectories for subset \(V\) are being optimized. The belief-space iLQG method [19] begins with an initial guess for the nominal trajectory and computes a locally optimal solution for \(\mathbf{x}_{\text{est}}\) by performing backward value iteration. The value iteration process involves quadratizing the cost function \(c_t\) along the nominal trajectory as follows:

\[
c_t \approx \frac{1}{2} \left[ \begin{array}{c} \delta \mathbf{b} \\ \delta \mathbf{u} \\ \delta \mathbf{b}_{b,t} \\ \delta \mathbf{b}_{u,t} \\ \delta \mathbf{u}_{b,t} \\ \delta \mathbf{u}_{u,t} \end{array} \right] \begin{bmatrix} \mathbf{c}_{bb,t} & \mathbf{c}_{bu,t} & \mathbf{c}_{bu,t} & \mathbf{c}_{uu,t} \end{bmatrix} \begin{bmatrix} \delta \mathbf{b} \\ \delta \mathbf{u} \\ \delta \mathbf{b}_{b,t} \\ \delta \mathbf{b}_{u,t} \\ \delta \mathbf{u}_{b,t} \\ \delta \mathbf{u}_{u,t} \end{bmatrix} + \delta c_t,
\]

where:

\[
\begin{align*}
\delta \mathbf{b} &= \mathbf{b}_{V,t} - \mathbf{b}_{V,t}, & \delta \mathbf{u} &= \mathbf{u}_{V,t} - \mathbf{u}_{V,t}, \\
\delta \mathbf{b}_{b,t} &= \frac{\partial^2 c_t}{\partial \mathbf{b}_V \partial \mathbf{b}_V} [\mathbf{b}_{V,t}, \mathbf{u}_{V,t}], & \delta \mathbf{b}_{u,t} &= \frac{\partial^2 c_t}{\partial \mathbf{u}_V \partial \mathbf{b}_V} [\mathbf{b}_{V,t}, \mathbf{u}_{V,t}], \\
\delta \mathbf{u}_{b,t} &= \frac{\partial^2 c_t}{\partial \mathbf{b}_V \partial \mathbf{u}_V} [\mathbf{b}_{V,t}, \mathbf{u}_{V,t}], & \delta \mathbf{u}_{u,t} &= \frac{\partial^2 c_t}{\partial \mathbf{u}_V \partial \mathbf{u}_V} [\mathbf{b}_{V,t}, \mathbf{u}_{V,t}], \\
\delta c_t &= \frac{\partial c_t}{\partial \mathbf{b}_V} [\mathbf{b}_{V,t}, \mathbf{u}_{V,t}],
\end{align*}
\]

As discussed in previous works related to belief-space iLQG [19], [21], one of the primary sources of a computational bottleneck lies in the computation of the Hessian \(\mathbf{c}_{bb,t}\). In our trajectory planner, from (27) to (30) and (36), note that the cost function \(c_t\) consists of the connectivity cost function \(J_F\). Thus, computation of \(\mathbf{c}_{bb,t}\) involves computing the Hessian of the connectivity cost function \(J_F^{bb,t} = \frac{\partial^2 J_F}{\partial \mathbf{b}_V \partial \mathbf{b}_V} [\mathbf{b}_{V,t}, \mathbf{u}_{V,t}]\). For our belief-space iLQG implementation, we observe that computing \(J_F^{bb,t}\) is the primary computational bottleneck. Thus, we analyze its complexity below.

For simplicity, we assume that the state vector \(\mathbf{x}_{\text{est},t}\) has the same dimension \(n\) for all robots in the system throughout the planning horizon. In this case, the dimension of the belief vector for each robot, as defined in [8], is \(O[\eta^2]\) since it contains elements from the state estimation covariance matrix. Thus, the dimension of the concatenated belief vector \(\mathbf{b}_V\) is \(O[\eta m^2]\), since \(V\) contains \(\eta\) elements. Given the dimension of \(\mathbf{b}_V\), the Hessian \(J_F^{bb,t}\) contains \(O[\eta^2 n^4]\) entries. The typical approach to compute the required Hessian in previous belief-space iLQG implementations is to use numerical differentiation (central differences) [19], [21]. Using numerical differentiation would require \(O[\eta^2 n^4]\) evaluations of \(J_F^t\), and consequently \(O[\eta^2 n^4]\) evaluations of \(\mathbf{L}_t^{\mathcal{C}}\). Considering the entire planning horizon, this results in \(O[\eta^2 n^4 T]\) evaluations of \(\mathbf{L}_t^{\mathcal{C}}\) per iteration of belief-space iLQG.

Evaluating \(\mathbf{L}_t^{\mathcal{C}}\) requires obtaining the Laplacian matrix \(\mathbf{L}_t\) whose elements depend on the distance measure as shown in (18). For obtaining the distance measures between all robots in the system, we need to perform eigendecompositions of the covariance matrices \(\Sigma_i, \forall i \in [1, N]\) as shown in (17). Assuming the dimension of the position vector \(\mathbf{p}_t\) to be \(n\) each eigendecomposition can be evaluated in \(O[\theta^3]\) time [26]. Thus, the complexity to obtain \(\mathbf{L}_t\) is of \(O[\theta^3 N]\). Once we obtain \(\mathbf{L}_t\), we need to perform another eigendecomposition with complexity of \(O[N^3]\) to obtain \(\mathbf{L}_t^{\mathcal{C}}\). Thus, the complexity of a single evaluation of \(\mathbf{L}_t^{\mathcal{C}}\) is of \(O[\max[\theta^3 N, N^3]]\). Given that we need \(O[\eta^2 n^4 T]\) evaluations of \(\mathbf{L}_t^{\mathcal{C}}\), we finally have a complexity of \(O[\eta^2 n^4 T \cdot \max[\theta^3 N, N^3]]\) per iteration of belief-space iLQG.

Since the belief-space iLQG method is used for the optimization step within each ADMM iteration, using numerical differentiation to compute \(J_F^{bb,t}\) results in a prohibitively large computational load. Thus, in the next subsection we present an approach to approximate \(J_F^{bb,t}\) and consequently reduce the required computational load for the belief-space iLQG method.

D. Hessian approximation for complexity reduction

In this subsection, we drop the time notation for simplicity since the presented approximation is applicable \(\forall t \in [0, T]\). As discussed in Section V-C, the primary computational bottleneck in our implementation of belief-space iLQG arises in computing \(J_F^{bb,t}\). Thus, in this subsection we derive an analytical expression to approximate \(J_F^{bb,t}\) and show that it significantly reduces the required computational load.

We begin by obtaining the gradient of our metric \(\mathcal{L}_t\) with respect to the belief vector \(\mathbf{b}_t\) as follows [6]:

\[
\frac{\partial \mathcal{L}_t}{\partial \mathbf{b}_t} = (\mathbf{e}_t^C)^\top \frac{\partial \mathcal{L}_t}{\partial \mathbf{b}_1} (\mathbf{e}_t^C) = \sum_{j=1}^{N} \frac{\partial \mathcal{L}_t}{\partial \mathbf{b}_1} \left( \mathbf{e}_2^{(i)} - \mathbf{e}_2^{(j)} \right)^2,
\]

where \(\mathbf{e}_t^C\) is the eigenvector of \(\mathcal{L}_t\) corresponding to the eigenvalue \(\lambda_t^{(i)}\), and \(\mathbf{e}_2^{(i)}\) is the \(i\)th element of \(\mathbf{e}_t^C\). From (38), we obtain the gradient of \(\mathcal{L}_t\) with respect to the belief vector \(\mathbf{b}_t\) as:

\[
\frac{\partial \mathcal{L}_t}{\partial \mathbf{b}_1} = -\frac{\pi}{2(\Delta - \Delta_0)} \sin \left( \pi(\mathbf{1}_{ij} - \Delta_0) \right) \frac{\partial \mathcal{L}_t}{\partial \mathbf{b}_1} \left( \mathbf{e}_2^{(i)} - \mathbf{e}_2^{(j)} \right)^2,
\]

Note that while the belief vector \(\mathbf{b}_t\) in (38) contains the state estimate and the estimation covariance, the distance measure \(\mathbf{1}_{ij}\) in (17) depends only on the position estimate \(\mathbf{p}_i\) and the position estimation covariance \(\Sigma_i\). Thus, in order to obtain \(\frac{\partial \mathcal{L}_t}{\partial \mathbf{b}_1}\) in (39), we only need the gradient of \(\mathbf{1}_{ij}\) with respect to each element of \(\mathbf{p}_i\) and with respect to each element of \(\Sigma_i\).

From (17), the gradient of \(\mathbf{1}_{ij}\) with respect to \(\mathbf{p}_i\) is (39), i.e., the \(m\)th element of \(\mathbf{p}_i\), is computed as:

\[
\frac{\partial \mathbf{1}_{ij}}{\partial \mathbf{p}_i^{(m)}} = \frac{\left( \hat{\mathbf{p}}_i^{(m)} - \hat{\mathbf{p}}_j^{(m)} \right)}{\| \mathbf{p}_i - \mathbf{p}_j \|^2},
\]

and the gradient of \(\mathbf{1}_{ij}\) with respect to element \((m, b)\) of \(\Sigma_i\) is computed as (27):

\[
\frac{\partial \mathbf{1}_{ij}}{\partial \Sigma_i^{(m, b)}} = \left( \frac{s}{2\sqrt{\lambda_i}} \mathbf{e}_i^{(b)} \right) \mathbf{e}_i^{(m)} \left( \frac{\partial \Sigma_i}{\partial \Sigma_i^{(m, b)}} \right),
\]

where \(\mathbf{e}_i^{(b)}\) is the eigenvector of \(\Sigma_i\) corresponding to the largest eigenvalue \(\lambda_i\). Thus, (41) simplifies to:

\[
\frac{\partial \mathbf{1}_{ij}}{\partial \Sigma_i^{(m, b)}} = \left( \frac{s}{2\sqrt{\lambda_i}} \mathbf{e}_i^{(b)} \right) \left( \hat{\mathbf{e}}_i^{(m)} \hat{\mathbf{e}}_i^{(b)} \right),
\]
Equations (40) and (43) allow us to construct the gradient \( \frac{\partial J_{cb}}{\partial b} \), which is required to obtain \( \frac{\partial J_{cb}}{\partial b} \) using (38) and (39).

By concatenating the gradients \( \frac{\partial J_{cb}}{\partial b} \) \( \forall i \in \mathcal{V} \), we obtain the gradient \( \frac{\partial J_{cb}}{\partial b} \).

Next, in order to approximate \( \bar{J}_{cb} \), we begin by writing the second-order Taylor expansion of \( J_c \) about \( B_V \):

\[
J_c[\Delta_2 [b_V]] \approx \frac{1}{2} J_{cb}(\Delta_2 [b_V] - \Delta_2 [b_V])^2 + \bar{J}_{cb}(\Delta_2 [b_V] - \Delta_2 [b_V]) + J_c, \tag{44}
\]

where:

\[
\bar{J}_{cb} = \frac{\partial^2 J_c}{\partial \Delta_2^2} \Delta_2 [b_V], \quad J_c = \frac{\partial J_c}{\partial \Delta_2} \Delta_2 [b_V],
\]

\[
J_c = J_c[\Delta_2 [b_V]].
\]

Here \( \bar{J}_{cb} \) is obtained from (26) as:

\[
\bar{J}_{cb} = \frac{2k_c}{(\Delta_2 [b_V] - c)^3}. \tag{45}
\]

We then approximate the term \( (\Delta_2 [b_V] - \Delta_2 [b_V]) \) in (44) using a first-order Taylor expansion about \( b_V \) as follows:

\[
(\Delta_2 [b_V] - \Delta_2 [b_V]) \approx (b_V - b_V)^T a,
\]

where \( a = \left( \frac{\partial \Delta_2}{\partial b_V} \right)^T \). By substituting (46) in (44), we get:

\[
J_c[\Delta_2 [b_V]] \approx \frac{1}{2} (b_V - b_V)^T (\bar{J}_{cb} a^T) (b_V - b_V) + (b_V - b_V)^T (\bar{J}_{cb} a) + J_c, \tag{47}
\]

where \( (\bar{J}_{cb} a a^T) \) is an approximation for \( J_{cb} \). Note that in order to compute the above approximation for \( J_{cb} \), we require only a single evaluation of \( \Delta_2 \) in (45). Considering the entire planning horizon, this results in only \( T \) evaluations of \( \Delta_2 \) per iteration of belief-space iLQG. This is in contrast to using numerical differentiation which requires \( O(n^3 T) \) evaluations as discussed in Section V-C. Thus, approximating \( J_{cb} \) with \( (\bar{J}_{cb} a a^T) \) significantly reduces the computational cost of the belief-space iLQG method.

In summary, this section presented our distributed ADMM-based trajectory planning algorithm for connectivity maintenance. Note that the ADMM optimization step in (30) is non-convex and that our algorithm involves additional components such as the line search algorithm and the approximation of the Hessian. Thus, it is non-trivial to provide guarantees of whether our planning algorithm converges to the optimal set of trajectories. However, as discussed in Section V-B, our planning algorithm ensures that the planned trajectories \( \bar{U} \) satisfy the connectivity maintenance requirement from (11).

**VI. SIMULATIONS**

In this section, we demonstrate the applicability of our trajectory planning algorithm for a multi-UAV mission under real-time constraints. We first describe details of the multi-UAV setup including the motion and sensing models. Next, we provide details of the simulated mission including the local tasks for the UAVs, the method used for generating initial trajectory guesses, and the values used for parameters within the planner. Finally, we discuss the performance of our planner across the simulated mission. Here we validate the connectivity maintenance of our algorithm by simulating 1000 trajectory rollouts in MATLAB, where each rollout shows a possible realization of the multi-UAV system’s trajectory under motion and sensing uncertainties. Additionally, we evaluate our planner on AirSim [28], a high-fidelity simulator that represents UAV motion more realistically. Fig. 3 shows a snapshot for the multi-UAV setup in AirSim. All simulations in this section are performed on a 2.80 GHz Quad-core IntelTM i7 machine.

**A. Multi-UAV system setup**

For each UAV in the multi-UAV system, we consider a 2-dimensional (2D) double integrator model as the motion model. The UAV state vector contains the 2D position and velocity, i.e., \( x_{i,t} = [p_{i,t}^T \dot{p}_{i,t}^T]^T \), and the input vector is the UAV accelerations. The motion model for the UAV can be written in the form of (1) as:

\[
x_{i,t} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_{i,t-1} + \begin{bmatrix} \frac{dt^2}{2} & 0 & \frac{dt^2}{2} & 0 \\ 0 & \frac{dt^2}{2} & 0 & \frac{dt^2}{2} \\ \frac{dt^2}{2} & 0 & \frac{dt^2}{2} & 0 \\ 0 & \frac{dt^2}{2} & 0 & \frac{dt^2}{2} \end{bmatrix} u_{i,t-1} + w_{i,t}, \tag{48}
\]

where \( dt \) is the time-step between two time instants and

\[
w_{i,t} \sim \mathcal{N} \left( 0, 0.1 \text{m}^2\text{s}^{-3} \begin{bmatrix} \frac{dt^3}{3} & 0 & \frac{dt^2}{2} & 0 \\ 0 & \frac{dt^3}{3} & 0 & \frac{dt^2}{2} \\ \frac{dt^3}{3} & 0 & \frac{dt^2}{2} & 0 \\ 0 & \frac{dt^3}{3} & 0 & \frac{dt^2}{2} \end{bmatrix} \right).
\]

As discussed in Section III-A, using a linear model to represent the UAV motion can be restrictive. However, a double-integrator model as in (48) has been previously used to represent UAVs [29, 21] and is more realistic than the single-integrator model used in the majority of related work. Note that it is important to choose an appropriately small time-step \( dt \) such that the system connectivity along the discretized trajectory represents the system connectivity in continuous time. For our simulations we choose \( dt = 0.2 \text{s} \) which also reflects the rate of common localization measurements from GNSS or camera. Additionally, we set a maximum limit of \( 5 \text{m}^2\text{s}^{-2} \) on the magnitude of input accelerations. For the sensing model in (5), we consider position measurements:

\[
z_{i,t} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x_{i,t} + v_{i,t}, \tag{49}
\]

where \( v_{i,t} \sim \mathcal{N} \left( 0, \text{diag}(1 \text{m}^2, 1 \text{m}^2) \right) \).

![Fig. 3: Snapshot of the AirSim simulator [28] used to evaluate our trajectory planning algorithm on a high-fidelity simulator.](image)
B. Simulated mission details

We consider the mission to be comprised of multiple segments and allow a maximum planning time (stopping criterion for the planner) for each segment. This resembles real-time applications such as exploration, coverage, or formation control, where only a limited amount of computation time is available for planning trajectories while the remaining time is required for other purposes such as analyzing sensor data or decision making. Fig. 4 shows how our trajectory planning algorithm is used across multiple segments of the mission. The trajectories for the first segment are planned while the system remains stationary. For the remaining segments, the trajectories for the upcoming segment are planned while executing the planned trajectory for the current segment.

We consider the multi-UAV system to consist of ten UAVs: six primary and four bridge. The goal of the primary robots is to reach desired positions, whereas the bridge robots focus on maintaining connectivity in the system. Here we assume that the desired positions are available from a high-level planning strategy (such as exploration or formation control), and pick them randomly for our simulation. Additionally, we assume that each UAV operates at a different altitude, similar to [21]. We make this assumption in order to alleviate inter-UAV collision constraints and focus on the connectivity maintenance of the system. For each segment, we consider the local task of reaching a desired position along with minimizing the control input effort. Thus, \( J_{i,t} \) for \( t \in [0, T] \) in Section III-C are set as:

\[
J_{i,t} = u_i^T W_i u_i, \quad \forall t \in [0, T-1]
\]

\[
J_{i,T} = (\hat{x}_{i,T} - x_{i,des})^T W_i^x (\hat{x}_{i,T} - x_{i,des}),
\]

where \( x_{i,des} = [p_{i,des}^T \; p_{i,des}^T]^T \) contains the desired position \( p_{i,des} \) and desired velocity \( p_{i,des} \) for the UAV, and \( W_i^x \) and \( W_i^w \) are used to set the relative importance of the different costs. \( p_{i,des} \) and \( p_{i,des} \) are the randomly picked desired positions and \( p_{i,des} \) is set to \( 0 \). We specify an initial position \( p_{i,init} \) for each UAV and set their initial velocities to be zeros. The initial state estimation covariance is set as \( P_{i,init} = \text{diag}(0.1 \text{m}^2, 0.1 \text{m}^2, 0.001 \text{m}^2 \text{s}^{-2}, 0.001 \text{m}^2 \text{s}^{-2}) \).

For the initial trajectory guess, we require a computationally inexpensive method of generating a trajectory based on the local tasks for each UAV. For example, sampling-based planners such as rapidly-exploring random trees (RRTs) can be used in order to quickly plan trajectories around obstacles [19], [29]; in a multiple target tracking application, each UAV can be randomly assigned to track a separate target [21]. In our algorithm, we use a linear-quadratic-regulator (LQR) to obtain an initial trajectory guess for each primary UAV from \( x_{i,init} \) to \( x_{i,des} \). For bridge UAVs, we simply set the initial trajectory guess to be hovering at the initial position. If \( \lambda^+ \) is not maintained above \( \epsilon \) for the resulting trajectory guess, we consider new desired states (only for the initial trajectory guess and not for the rest of the planner) midway between \( x_{i,init} \) and \( x_{i,des} \) for all primary UAVs and repeat the process.

We consider the mission to consist of six segments. For each segment, we set a maximum planning time of 25 s and a trajectory duration of 50 s (\( T = 250 \)). Additionally, since our planner uses a centralized communication architecture, we account for a delay of 0.2 s (by simply pausing the code execution) in each ADMM iteration of our planning algorithm. We consider a communication range of \( \Delta = 40 \text{ m} \) and set the parameter \( \Delta_0 = 35 \text{ m} \) in (18). For the connectivity maintenance requirement in (11), we specify the lower algebraic connectivity limit \( \epsilon = 0.1 \) and the corresponding probability value \( \delta = 0.003 \) to reflect a 3σ confidence level. In (25), we set the parameter for the magnitude of the connectivity cost as \( k_c = 0.001 \). We set \( \eta = 2 \) for the number of elements in subsets \( \mathcal{V} \) obtained in the trajectory planner. Finally, for the line search algorithm used to update the ADMM consensus variable in (32), we set \( \gamma = 0.8 \).

C. Planning results under real-time constraints

Fig. 4 shows the results of our planner for the simulated real-time mission. In Figs. 5(a)-(f) we show the planned trajectories for the six mission segments (blue for primary UAVs and black for bridge UAVs), along with 1000 rollouts (gray) showing possible trajectory realizations of the system. Additionally, we show the trajectories of the system simulated in AirSim (magenta). For all the segments we observe that the planner attempts to drive the primary UAVs to their desired positions while rearranging the bridge UAVs for maintaining connectivity. For cases when the desired positions might lead to a loss of connectivity (such as the leftmost desired position in Fig. 5(a)), our planner attempts to bring the primary UAVs as close as possible while maintaining connectivity. Also, note that the planned trajectories closely match the UAV motion from AirSim, thus validating the use of a double-integrator motion model in (48).

Figs. 5(g)-(l) show the convergence of the cost from (29) for the six mission segments. We observe that our planner is able to find lower cost trajectories within the 25 s planning time for various desired system configurations. In Fig. 5(m) we analyze the connectivity maintenance throughout the mission. Our planner maintains the algebraic connectivity of our weighted graph \( \Lambda^C \) (blue) above the specified lower limit \( \epsilon = 0.1 \) (blue-dashed). In order to validate that the connectivity maintenance requirement from (11) is satisfied, we plot the algebraic connectivity of the 1000 trajectory rollouts (gray), which are obtained using (10). Note that since the edge weights in (10) are binary, the corresponding algebraic connectivity contains jumps when an edge weight changes from 0 to 1, or from 1 to 0. We observe that the algebraic connectivity for only one rollout (red) drops below \( \Lambda^C \), while none drop below \( \epsilon \). This validates that our planner satisfies the connectivity
Fig. 5: Trajectory planning and connectivity maintenance for the simulated mission with six primary and four bridge UAVs. (a)-(f) The final planned trajectories, 1000 trajectory rollouts and AirSim trajectory for each mission segment. (g)-(l) Convergence of the cost in the transformed optimization problem for a maximum planning time of 25 s. (m) Connectivity maintenance performance throughout the mission. Only one (red) of the 1000 trajectory rollouts results in system algebraic connectivity $\lambda_2^L$ less than $\lambda_2^L$, whereas none drop below the specified lower limit $\epsilon = 0.1$.

VII. CONCLUSIONS

We have presented a trajectory planning algorithm for global connectivity maintenance of multi-robot systems that addresses two limitations in related work: it accounts for robot motion and sensing uncertainties, and it considers general linear robot motion models which do not necessarily have an instantaneously changeable direction of motion. For connectivity maintenance, we first define a weighted undirected graph to represent connectivity of a system with uncertain robot positions. The algebraic connectivity of this graph is then used to define a transformed trajectory planning problem which is solved by a distributed ADMM setup. We present an approach to reduce the computational load of the ADMM optimization step by approximating the required Hessian matrices. Finally, we evaluate the planner under real-time constraints on a simulated multi-UAV mission with multiple segments. Our planner plans trajectories attempting the complete the local UAV tasks...
while satisfying the connectivity maintenance requirement.

While we have demonstrated the utility of our planner in addressing the aforementioned limitations in related work, multiple future directions of work exist. First, a natural extension includes exploring decentralized architectures in order to improve the scalability with respect to the communication load. Second, we plan to extend our algorithm to more general robot motion models such as feedback linearizable or differentially flat systems. We also plan to evaluate our planner for nonlinear systems using an approximation of the state uncertainty provided by the EKF. Third, it is desirable to include additional realistic constraints for the multi-robot system such as line-of-sight communication and collision avoidance. Finally, we also plan to test the planner on a real-world hardware multi-robot platform.

REFERENCES

[1] Y. Rizk, M. Awad, and E. W. Tunstel, “Cooperative Heterogeneous Multi-robot Systems: A Survey,” ACM Computing Surveys (CSUR), vol. 52, no. 2, pp. 1–31, 2019.

[2] A. Alanwar, H. Said, and M. Aithoff, “Distributed Secure State Estimation Using Diffusion Kalman Filters and Reachability Analysis,” in 2019 IEEE 58th Conference on Decision and Control (CDC). IEEE, 2019, pp. 4133–4139.

[3] H. Park and S. Hutchinson, “Robust Rendezvous for Multi-robot System with Random Node Failures: An Optimization Approach,” Autonomous Robots, vol. 42, no. 8, pp. 1807–1818, 2018.

[4] K. Khatari, M. Pourgholi, M. Montazeri, and L. Sabattini, “A Comparison Between Decentralized Local and Global Methods for Connectivity Maintenance of Multi-robot Networks.” IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 633–640, 2019.

[5] M. C. De Gennaro and A. Jadbabaie, “Decentralized Control of Connectivity for Multi-agent Systems,” in Proceedings of the 45th IEEE Conference on Decision and Control. IEEE, 2006, pp. 3628–3633.

[6] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, and R. Sukthankar, “Decentralized Estimation and Control of Graph Connectivity for Mobile Sensor Networks,” Automatica, vol. 46, no. 2, pp. 390–396, 2010.

[7] L. Sabattini, N. Chopra, and C. Scacci, “Decentralized Connectivity Maintenance for Cooperative Control of Mobile Robotic Systems,” The International Journal of Robotics Research, vol. 32, no. 12, pp. 1411–1422, oct 2013. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364913499085

[8] L. Sabattini, C. Scacci, N. Chopra, and A. Gasparri, “Distributed Control of Multirobot Systems with Global Connectivity Maintenance,” IEEE Transactions on Robotics, vol. 29, no. 5, pp. 1326–1332, 2013.

[9] P. Robuffo Giordano, A. Franchi, C. Scacci, and H. H. Bülthoff, “A Passivity-based Decentralized Strategy for Generalized Connectivity Maintenance,” The International Journal of Robotics Research, vol. 32, no. 3, pp. 299–323, 2013.

[10] A. Gasparri, L. Sabattini, and G. Ulivi, “Bounded Control Law for Global Connectivity Maintenance in Cooperative Multirobot Systems,” IEEE Transactions on Robotics, vol. 33, no. 3, pp. 700–717, 2017.

[11] B. Capelli and L. Sabattini, “Connectivity Maintenance: Global and Optimized approach through Control Barrier Functions,” Proceedings - IEEE International Conference on Robotics and Automation, pp. 5590–5596, may 2020.

[12] S. Bhattacharya and T. Başar, “Graph-theoretic Approach for Connectivity Maintenance in Mobile Networks in the Presence of a Jammer,” in Proceedings of the IEEE Conference on Decision and Control, 2010, pp. 3560–3565.

[13] S. Thrun, W. Burgard, and D. Fox, “Probabilistic Robotics (Intelligent Robots and Autonomous Agents series), ser. Intelligent Robotics and Autonomous Agents,” 2005.

[14] M. Owen, R. Beard, and T. McLain, “Implementing Dubins Airplane Paths on Fixed-wing UAVs,” Contributed chapter to the Handbook of Unmanned Aerial Vehicles, pp. 1677–1701, 2014.

[15] S. Boyd, N. Parikh, E. Chu, J. Eckstein, S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Duality, Optimization and Statistical Learning via the Alternating Direction Method of Multipliers,” Foundations and Trends R in Machine Learning, vol. 3, no. 1, pp. 1–122, 2010.

[16] W. Luo, S. Yi, and K. Sycara, “Behavior Mixing with Minimum Global and Subgroup Connectivity Maintenance for Large-Scale Multi-Robot Systems,” Proceedings - IEEE International Conference on Robotics and Automation, pp. 9845–9851, May 2020.

[17] J. Scherer and B. Rinne, “Multi-Robot Persistent Surveillance with Connectivity Constrained,” IEEE Access, vol. 8, pp. 15 093–15 109, 2020.

[18] W. Luo and K. Sycara, “Minimum k-connectivity Maintenance for Robust Multi-Robot Systems,” in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019, pp. 7370–7377.

[19] J. Van Den Berg, S. Patil, and R. Alterovitz, “Motion Planning Under Uncertainty using Iterative Local Optimization in Belief Space,” The International Journal of Robotics Research, vol. 31, no. 11, pp. 1263–1278, 2012.

[20] Z. Hou and I. Fantoni, “Distributed Leader-Follower Formation Control for Multiple Quadrotors With Weighted Topology,” in 2015 10th System of Systems Engineering Conference (SoSE). IEEE, 2015, pp. 256–261.

[21] S.-S. Park, Y. Min, J.-S. Ha, D.-H. Cho, and H.-L. Choi, “A Distributed ADMM Approach to Non-Monopath Path Planning for Multi-Target Tracking,” IEEE Access, vol. 7, pp. 163 589–163 613, 2019.

[22] R. Grone, R. Merris, and V. S. Sunder, “The Laplacian Spectrum of a Graph,” SIAM Journal on matrix analysis and applications, vol. 11, no. 2, pp. 218–228, 1990.

[23] W. E. Hoover, “Algorithms for Confidence Circles and Ellipses,” NOAA Technical Report, 1984.

[24] A. Makkhoudi and A. Ozdaglar, “Convergence Rate of Distributed ADMM over Networks,” IEEE Transactions on Automatic Control, vol. 62, no. 10, pp. 5082–5095, 2017.

[25] J. J. Moré and D. J. Thuente, “Line Search Algorithms with Guaranteed Sufficient Decrease,” ACM Transactions on Mathematical Software (TOMS), vol. 20, no. 3, pp. 386–397, 1994.

[26] V. Y. Pan and Z. Q. Chen, “The Complexity of the Matrix Eigenproblem,” in Proceedings of the thirty-first annual ACM symposium on Theory of computing, 1999, pp. 507–516.

[27] E. Stump, A. Jadbabaie, and V. Kumar, “Connectivity Management in Mobile Robot Teams,” in 2008 IEEE International Conference on Robotics and Automation. IEEE, 2008, pp. 1525–1530.

[28] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “AirSim: High-fidelity Visual and Physical Simulation for Autonomous Vehicles,” in Field and service robotics. Springer, 2018, pp. 621–635.

[29] G. Goretkin, A. Perez, R. Platt, and G. Konidaris, “Optimal Sampling-based Planning for Linear-quadratic Kinodynamic Systems,” in 2013 IEEE International Conference on Robotics and Automation. IEEE, 2013, pp. 2429–2436.

Akshay Shetty is a postdoctoral researcher in the Department of Aerospace Engineering at Stanford University. He received his Ph.D. degree in Aerospace Engineering from University of Illinois at Urbana-Champaign in 2021. His research interests include autonomous robotic navigation, robust perception, and safe control.

Derek Knowles received the B.S. degree in mechanical engineering from Brigham Young University in 2019. He is currently pursuing the Ph.D. degree in mechanical engineering from Stanford University. His research interests include autonomous robotic navigation, robust perception, and safe control.

Grace Xingxin Gao is an assistant professor in the Department of Aeronautics and Astronautics at Stanford University. Before joining Stanford University, she was an assistant professor at University of Illinois at Urbana-Champaign. She obtained her Ph.D. degree at Stanford University. Her research is on robust and secure positioning, navigation and timing with applications to manned and unmanned aerial vehicles, robotics and power systems.