

# Decentralized Connectivity Maintenance for Multi-Robot Systems Under Motion and Sensing Uncertainties

Akshay Shetty | Timmy Hussain | Grace Gao

Department of Aeronautics and  
Astronautics  
Stanford University  
Stanford, CA, USA

## Correspondence

Grace Gao,  
Department of Aeronautics and  
Astronautics, Stanford University,  
Stanford, CA, USA, 94305  
Email: [gracegao@stanford.edu](mailto:gracegao@stanford.edu)

## Abstract

Communication connectivity is desirable for the safe and efficient operation of multi-robot systems. While decentralized algorithms for connectivity maintenance have been explored in recent literature, the majority of these works do not account for robot motion and sensing uncertainties. These uncertainties are inherent in practical robots and result in robots deviating from their desired positions which could potentially result in a loss of connectivity. In this paper, we present a decentralized connectivity maintenance algorithm accounting for robot motion and sensing uncertainties (DCMU). We, first, propose a novel weighted graph definition for the multi-robot system that accounts for the aforementioned uncertainties along with realistic connectivity constraints such as line-of-sight connectivity and collision avoidance. We, then, design a decentralized gradient-based controller for connectivity maintenance with which we derive the gradients of the weighted graph edge weights required for computing the control. Finally, we perform multiple simulations to validate the connectivity maintenance performance of our DCMU algorithm under robot motion and sensing uncertainties, showing an improvement compared to previous work.

## Keywords

connectivity maintenance, decentralized algorithms, gradient-based control, motion and sensing uncertainties, multi-robot systems

## 1 | INTRODUCTION

Multi-robot systems are increasingly being used for various tasks such as exploration, target tracking, and search and rescue (Cortés & Egerstedt, 2017; Rizk et al., 2019). One of the primary advantages of multi-robot systems is their ability to coordinate using inter-robot communication, which allows them to execute complex tasks in a safe and efficient manner (Alanwar et al., 2019; Bhamidipati & Gao, 2019; Park & Hutchinson, 2018). Thus, it is highly desirable to maintain communication connectivity within a multi-robot system.

Existing literature has explored decentralized algorithms for connectivity maintenance due to their communication efficiency and scalability properties (Khateri et al., 2019). These works typically represent the system as a weighted graph, where each node represents a robot and each edge represents the communication

connection between two robots (Giordano et al., 2013; Sabattini et al., 2013; Yang et al., 2010). The edge weights between robots are formulated as a function of the robot position subject to constraints such as maximum communication range, line-of-sight communication, and collision avoidance (both inter-robot and obstacles). One commonly used metric for the connectivity of the system is algebraic connectivity, which depends on the graph edge weights as explained later in Section 2. A value greater than zero indicates that the system is connected. Thus, in order to maintain connectivity within the system, previous works (Giordano et al., 2013; Sabattini et al., 2013; Yang et al., 2010) derived a decentralized gradient-based controller to maintain the algebraic connectivity above a specified lower limit.

While previous works present efficient decentralized algorithms for connectivity maintenance, they assume that the robot positions are deterministic and do not explicitly account for robot motion and sensing uncertainties. Here, motion uncertainties refer to the errors between the actual robot motion and a mathematical motion model, whereas sensing uncertainties refer to errors in measurements such as errors in localization measurements. These motion and sensing uncertainties, which are inherent in practical robots, result in robots deviating from their desired nominal positions. These deviations, if not accounted for, can potentially result in a loss of connectivity in the multi-robot system. Thus, it is important to account for robot motion and sensing uncertainties while designing connectivity maintenance algorithms.

In our prior work (Shetty et al., 2020), we defined a weighted graph to represent the connectivity of the multi-robot system while accounting for deviations due to motion and sensing uncertainties. We, then, used a distributed trajectory planner to maintain connectivity within the system. However, our planner in Shetty et al. (2020) required a centralized communication setup (i.e., each robot communicates with all other robots in the system in a single planning iteration). While such communication is possible (perhaps via multi-hop connections), in practice, it introduces additional communication delays and does not scale favorably to large systems. Thus, decentralized algorithms, as proposed in Giordano et al. (2013), Sabattini et al. (2013), and Yang et al. (2010), are desirable since they require a robot to communicate only with its immediate neighboring robots in a single planning iteration. Additionally, the weighted graph in Shetty et al. (2020) assumed a simplistic connectivity model subject to only a maximum communication range constraint, as opposed to the more realistic line-of-sight communication and collision avoidance constraints in Giordano et al. (2013).

In this work, we present a decentralized algorithm (referred to as DCMU) for connectivity maintenance of multi-robot systems while accounting for motion and sensing uncertainties. A decentralized gradient-based controller is implemented in order to maintain system connectivity subject to constraints of maximum communication range, line-of-sight communication, and collision avoidance. Table 1 outlines the contributions of this work in comparison to previous connectivity maintenance works. This paper is based on our work in Shetty et al. (2021). The main contributions of this work are listed as follows:

1. We propose a weighted graph that accounts for deviations in robot positions arising due to motion and sensing uncertainties. Compared to our prior work (Shetty et al., 2020), we include more realistic constraints of line-of-sight communication and collision avoidance (both inter-robot and obstacles) in addition to a maximum communication range.
2. We derive a decentralized gradient-based controller in order to maintain the algebraic connectivity of the proposed weighted graph above a specified lower

limit. Here, we derive the gradients of the edge weights of our weighted graph that are required for computing the control. These edge weights account for deviations in robot positions instead of assuming deterministic robot positions as in Sabattini et al. (2013) and Yang et al. (2010).

3. We present extensive simulation results to evaluate our DCMU algorithm under motion and sensing uncertainties. We compare our connectivity maintenance performance with Sabattini et al. (2013) and validate our algorithm on a high-fidelity simulator (Shah et al., 2018).

The remainder of the paper is organized as follows. We, first, introduce relevant background from graph theory in Section 2. Next, in Section 3, we describe the models used for the robot motion and sensing uncertainties, and formulate the connectivity maintenance problem. Section 4 provides details of our DCMU algorithm including the weighted graph definition and the gradient-based controller. Finally, in Section 5, we present our simulation results validating the connectivity maintenance performance of our algorithm.

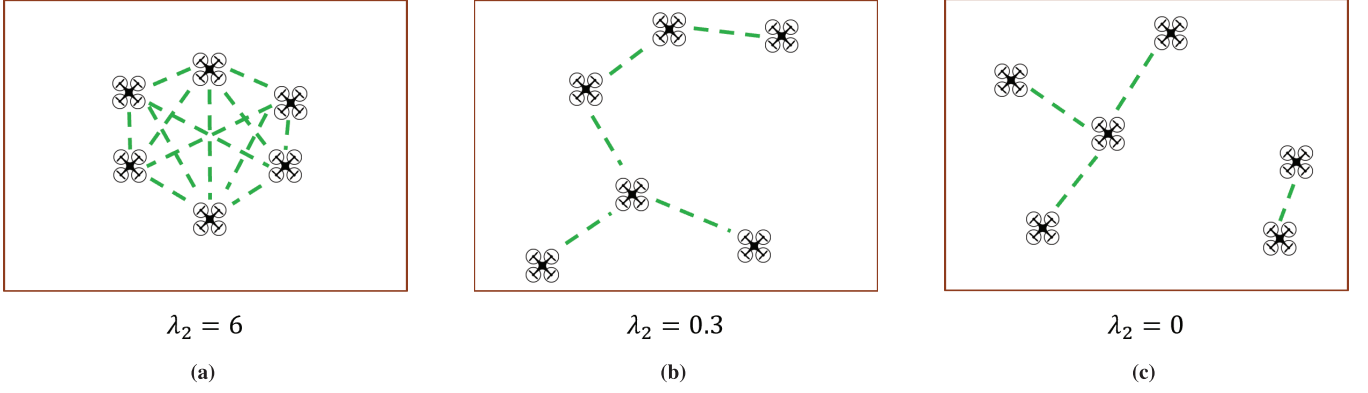
## 2 | PRELIMINARIES: GRAPH THEORY

A multi-robot system can be represented as an undirected graph in which each node represents a robot and each edge represents the communication connection between two robots. Let  $n$  be the number of nodes in the graph. The adjacency matrix,  $A$ , of the graph is defined as an  $n \times n$  matrix where  $a_{ij} \in [0, 1]$  represents the edge weight between two nodes,  $i$  and  $j$ , with  $a_{ii} = 0$  (Grone et al., 1990). The degree of a node is defined as  $d_i = \sum_{j=1}^n a_{ij}$ . The vector of node degrees  $\mathbf{d} = [d_1, \dots, d_n]$  is then used to define the degree matrix,  $D$ , of the graph as  $D = \text{diag}(\mathbf{d})$ . Given matrices  $A$  and  $D$ , the Laplacian matrix,  $L$ , of the graph is defined as  $L = D - A$  (Grone et al., 1990).

Let  $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$  be the eigenvalues of  $L$ . The algebraic connectivity of the graph, also known as the Fiedler value, is defined as the second-smallest eigenvalue of  $L$  (i.e.,  $\lambda_2$ ). As mentioned in Section 1, algebraic connectivity is a commonly used metric to represent the connectivity of multi-robot systems. It varies from zero (if the graph is disconnected) to the number of nodes in the graph (if the graph is fully connected), i.e.  $0 \leq \lambda_2 \leq n$ . Figure 1 illustrates the algebraic connectivity for different configurations of a graph. Thus,  $\lambda_2$  remains greater than zero as long as there exists a (potentially multi-hop) communication path between any two robots in the system.

**TABLE 1**  
Comparison Between This Work and Recent Connectivity Maintenance Works

	Decentralized	Motion and sensing uncertainties	Line-of-sight and collision avoidance
Yang et al. (2010), Sabattini et al. (2013), Gasparri et al. (2017), and Siligardi et al. (2019)	✓		
Giordano et al. (2013)	✓		✓
Shetty et al. (2020)		✓	
<b>This work</b>	✓	✓	✓



**FIGURE 1** The algebraic connectivity  $\lambda_2$  of a graph varies from the number of nodes (fully-connected graph) to zero (disconnected graph).  $\lambda_2 > 0$  implies that the graph is connected.

### 3 | PROBLEM FORMULATION

#### 3.1 | Robot Description

For each robot,  $i$ , in the multi-robot system, we consider a discrete-time, single-integrator motion model:

$$\mathbf{x}_{i,t} = \mathbf{x}_{i,t-1} + B\mathbf{u}_{i,t-1} + \mathbf{w}_{i,t} \quad (1)$$

where  $t$  represents the time instant,  $\mathbf{x}_{i,t}$  is the state vector representing the robot position,  $\mathbf{u}_{i,t}$  is the velocity control input,  $B$  is the control input matrix, and  $\mathbf{w}_{i,t}$  is a zero-mean Gaussian-distributed error vector with covariance matrix,  $Q_{i,t}$ , i.e.,  $\mathbf{w}_{i,t} \sim \mathcal{N}(0, Q_{i,t})$ . The control input matrix in Equation (1) is set as  $B = (\Delta t)I$ , where  $\Delta t$  is the discrete time-step. Note that, while this is a simplified motion model, it can still be used to control real robots as demonstrated in previous works on aerial robots (Lee et al., 2013; Soukieh et al., 2009). For the sensing model, we assume each robot,  $i$ , obtains position measurements:

$$\mathbf{z}_{i,t} = \mathbf{x}_{i,t} + \mathbf{v}_{i,t} \quad (2)$$

where  $\mathbf{z}_{i,t}$  is the measurement vector representing position measurements and  $\mathbf{v}_{i,t}$  is a zero-mean, Gaussian-distributed error vector with covariance matrix  $R_{i,t}$ , i.e.,  $\mathbf{v}_{i,t} \sim \mathcal{N}(0, R_{i,t})$ . Given the linear motion and sensing models with Gaussian uncertainties, we set each robot to use a Kalman filter (KF) for state estimation. The prediction step of the KF is performed as:

$$\bar{\mathbf{x}}_{i,t} = \hat{\mathbf{x}}_{i,t-1} + B\mathbf{u}_{i,t-1} \quad (3)$$

$$\bar{P}_{i,t} = P_{i,t-1} + Q_{i,t} \quad (4)$$

where  $P_{i,t}$  is the state estimation covariance matrix such that  $\mathbf{x}_{i,t} \sim \mathcal{N}(\hat{\mathbf{x}}_{i,t}, P_{i,t})$ . The correction step of the KF is performed as:

$$G_{i,t} = \bar{P}_{i,t}(\bar{P}_{i,t} + R_{i,t})^{-1} \quad (5)$$

$$\hat{\mathbf{x}}_{i,t} = \bar{\mathbf{x}}_{i,t} + G_{i,t}(\mathbf{z}_{i,t} - \bar{\mathbf{x}}_{i,t}) \quad (6)$$

$$P_{i,t} = \bar{P}_{i,t} - G_{i,t}\bar{P}_{i,t} \quad (7)$$

where  $G_{i,t}$  is the Kalman gain matrix. In order to track a desired nominal state,  $\check{\mathbf{x}}_{i,t}$ , we assume that each robot uses linear feedback control, where the total control input,  $\mathbf{u}_{i,t}$ , is of the form:

$$\mathbf{u}_{i,t} = \check{\mathbf{u}}_{i,t} - \check{K}_i(\hat{\mathbf{x}}_{i,t} - \check{\mathbf{x}}_{i,t}) \quad (8)$$

where  $\check{K}_i$  is the feedback control gain, which can be designed using methods such as classical control theory or Linear-Quadratic Regulator (LQR) design (Doyle et al., 2013), and  $\check{\mathbf{u}}_{i,t}$  is the nominal control input which relates to the nominal states as:

$$\check{\mathbf{x}}_{i,t} = \check{\mathbf{x}}_{i,t-1} + B\check{\mathbf{u}}_{i,t-1} \quad (9)$$

Later in Section 3.2, we discuss how nominal control input is obtained for different robots in the multi-robot system.

Given the above setup for each robot,  $i$ , the distribution of the robot's true state,  $\mathbf{x}_{i,t}$ , about its nominal state,  $\check{\mathbf{x}}_{i,t}$ , can be obtained using the method described in Bry and Roy (2011). The authors derived the following Gaussian distribution for the robot's true state:

$$\mathbf{x}_{i,t} \sim \mathcal{N}(\check{\mathbf{x}}_{i,t}, \Sigma_{i,t}) \quad (10)$$

where  $\Sigma_{i,t} = P_{i,t} + \Lambda_{i,t}$ . Here,  $P_{i,t}$  is the state estimation covariance matrix from Equation (7), and  $\Lambda_{i,t}$  can be obtained iteratively as follows (Bry & Roy, 2011):

$$\Lambda_{i,t} = (I - B\check{K}_i)\Lambda_{i,t-1}(I - B\check{K}_i)^\top + G_{i,t}\bar{P}_{i,t} \quad (11)$$

where  $I$  is an identity matrix and  $\Lambda_{i,0} = O$ . In essence, the distribution  $\mathbf{x}_{i,t} \sim \mathcal{N}(\check{\mathbf{x}}_{i,t}, \Sigma_{i,t})$  in Equation (10) captures the deviations of the robot's position from its desired nominal position that arise due to the motion and sensing uncertainties.

### 3.2 | Multi-Robot System: Connectivity Maintenance

We consider the multi-robot system to be comprised of two types of robots: leader robots and follower robots. For the leader robots, we assume that the nominal control inputs in Equation (9) are available from a high-level planner, such as from an exploration or a search and rescue strategy (Baxter et al., 2007; Burgard et al., 2005). On the other hand, the objective of follower robots is to maintain connectivity within the multi-robot system. Our DCMU algorithm presented in Section 4 focuses on deriving the nominal control inputs for these follower robots.

In practice, the communication connectivity between two robots can be considered to be binary (i.e., the robots are either connected if certain constraints are satisfied [such as minimum signal strength, bandwidth, etc.] or they are disconnected). We consider any two robots,  $i$  and  $j$ , in the multi-robot system to be connected if the following constraints are satisfied:

1. **Communication range constraint:** The distance between the robots is within a maximum communication range,  $\rho$  (i.e.,  $\|\mathbf{x}_{i,t} - \mathbf{x}_{j,t}\|_2 \leq \rho$ ).
2. **Line-of-sight constraint:** The robots maintain a direct line of sight (i.e., the line segment connecting  $\mathbf{x}_{i,t}$  and  $\mathbf{x}_{j,t}$  is not obstructed by obstacles in the environment).

- Collision avoidance constraint:** The robots are not colliding with any obstacles or other robots in the system; this constraint helps ensure that the communication equipment is not damaged due a collision.

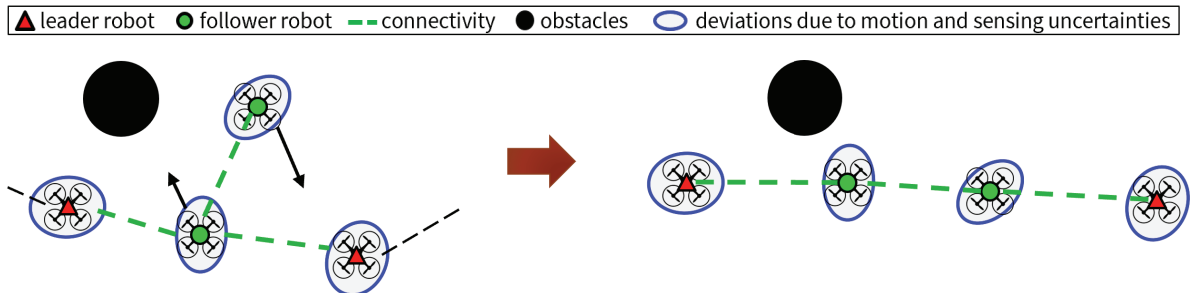
Let  $\bar{a}_{ij} = \{0, 1\}$  represent the binary connectivity between robots  $i$  and  $j$ , i.e.,  $\bar{a}_{ij} = 1$  if the robots are connected (if the above constraints are satisfied) and  $\bar{a}_{ij} = 0$  otherwise (if any of the above constraints is not satisfied). Note that, by definition,  $\bar{a}_{ii} = 0$ . These  $\bar{a}$  values can then be used to obtain the algebraic connectivity,  $\bar{\lambda}_2$  (as explained in Section 2), of the graph which represents the connectivity of the multi-robot system based on the actual robot positions,  $\mathbf{x}_{i,t}$ . Thus,  $\bar{\lambda}_2 > 0$  implies that the multi-robot system is connected.

We define the problem for our DCMU algorithm as: Given a multi-robot system as described in this section, derive nominal control inputs ( $\check{\mathbf{u}}_{i,t}$  in Equation [9]) for the follower robots such that  $\bar{\lambda}_2$  is always above a specified lower limit  $\epsilon$  (i.e.,  $\bar{\lambda}_2 > \epsilon$ ). Figure 2 illustrates the defined problem.

## 4 | PROPOSED CONNECTIVITY MAINTENANCE ALGORITHM: DCMU

During execution, each robot,  $i$ , in the multi-robot system deviates from its desired nominal position,  $\check{\mathbf{x}}_{i,t}$ , due to motion and sensing uncertainties. Previous connectivity maintenance works (Gasparri et al., 2017; Giordano et al., 2013; Sabattini et al., 2013; Siligardi et al., 2019; Yang et al., 2010) that do not account for these uncertainties essentially assume that the robots are at their nominal positions (i.e.,  $\mathbf{x}_{i,t} = \check{\mathbf{x}}_{i,t}$ ). In contrast, our DCMU algorithm accounts for the deviations of  $\mathbf{x}_{i,t}$  from  $\check{\mathbf{x}}_{i,t}$  as modeled by Equation (10).

In this section, we provide the details of our DCMU algorithm. We, first, define a weighted graph to represent the connectivity of the multi-robot system where the edge weights account for deviations due to motion and sensing uncertainties. Next, we implement a decentralized power iteration method for each robot to estimate connectivity information of the weighted graph. Finally, each follower robot uses a gradient-based controller to obtain the nominal control input,  $\check{\mathbf{u}}_{i,t}$ , for maintaining connectivity of the weighted graph. Here, we derive the expressions for the gradients of our weighted graph edge weights, which are required to compute the nominal control input,  $\check{\mathbf{u}}_{i,t}$ . While, for simplicity, we drop the time notation in the remainder of this section, the nominal control input computed by our DCMU algorithm is applicable for all time instants,  $t$ .



**FIGURE 2** The objective of our DCMU algorithm is to derive nominal control inputs (refer to the black arrows) for follower robots in order to maintain communication connectivity within the system.

## 4.1 | Weighted Graph Definition

For the edge weights to represent connectivity between robots, we need to consider the constraints listed in Section 3.2. Thus, for our weighted graph, we define the edge weight between any two robots,  $i$  and  $j$ , as a product of four factors:

$$a_{ij} = \alpha_{ij}\beta_{ij}\gamma_i\gamma_j \quad (12)$$

where  $\alpha_{ij}$  is a factor for the communication range constraint,  $\beta_{ij}$  is a factor for the line-of-sight constraint, and  $\gamma_i$  and  $\gamma_j$  are factors for the collision avoidance constraints of the two robots. Note that Equation (12) ensures that the edge weight,  $a_{ij}$ , is equal to zero (i.e., robots  $i$  and  $j$  are not connected) if any of the factors  $\alpha_{ij}$ ,  $\beta_{ij}$ ,  $\gamma_i$ , or  $\gamma_j$  is equal to zero (which occurs when the corresponding constraint is not satisfied). Next, we define the smooth functions for each of these factors that allow us to compute gradient-based control inputs as discussed later in Section 4.3.

1. **Communication range factor ( $\alpha_{ij}$ ):** This factor indicates how well robots  $i$  and  $j$  are connected in terms of being within a maximum communication range. We begin by defining a conservative measure,  $\bar{l}_{ij}$ , for the range between two robots while accounting for their deviations from their nominal positions up to a desired confidence level.  $\bar{l}_{ij}$  is defined as:

$$\bar{l}_{ij} = \|\check{\mathbf{x}}_i - \check{\mathbf{x}}_j\|_2 + s\sqrt{\bar{\lambda}^{\Sigma_i}} + s\sqrt{\bar{\lambda}^{\Sigma_j}} \quad (13)$$

where  $\check{\mathbf{x}}_i$  and  $\check{\mathbf{x}}_j$  are the nominal positions,  $s$  is a scalar value obtained using the chi-square distribution for a desired confidence level (Hoover, 1984), and  $\bar{\lambda}^{\Sigma_i}$  and  $\bar{\lambda}^{\Sigma_j}$  are the largest eigenvalues of the covariance matrices,  $\Sigma_i$  and  $\Sigma_j$ . Later in Section 5, we demonstrate that a  $3\sigma$  confidence level (reflecting a  $\sim 99.7\%$  confidence level) results in acceptable connectivity maintenance performance. Figure 3(a) illustrates the conservative measure,  $\bar{l}_{ij}$ .

Given  $\bar{l}_{ij}$  and a maximum communication range,  $\rho$ , we define the communication range factor in Equation (12) as:

$$\alpha_{ij} = \begin{cases} 1 & 0 \leq \bar{l}_{ij} \leq \rho_0 \\ \frac{1}{2} + \frac{1}{2} \cos \left[ \frac{\pi(\bar{l}_{ij} - \rho_0)}{\rho - \rho_0} \right] & \rho_0 < \bar{l}_{ij} \leq \rho \\ 0 & \bar{l}_{ij} > \rho \end{cases} \quad (14)$$

where  $\rho_0$  is a parameter such that  $\rho_0 < \rho$ . Figure 4(a) shows how  $\alpha_{ij}$  varies with  $\bar{l}_{ij}$ .

2. **Line-of-sight range factor ( $\beta_{ij}$ ):** This factor indicates how well robots  $i$  and  $j$  are connected in terms of being in line of sight with each other. Similar to  $\bar{l}_{ij}$  in Equation (13), we define a conservative measure,  $\bar{l}_{ij,o}$ , for how close the line-of-sight vector between two robots is to nearby obstacles, as illustrated in Figure 3(b). Thus,  $\bar{l}_{ij,o}$  is defined as:

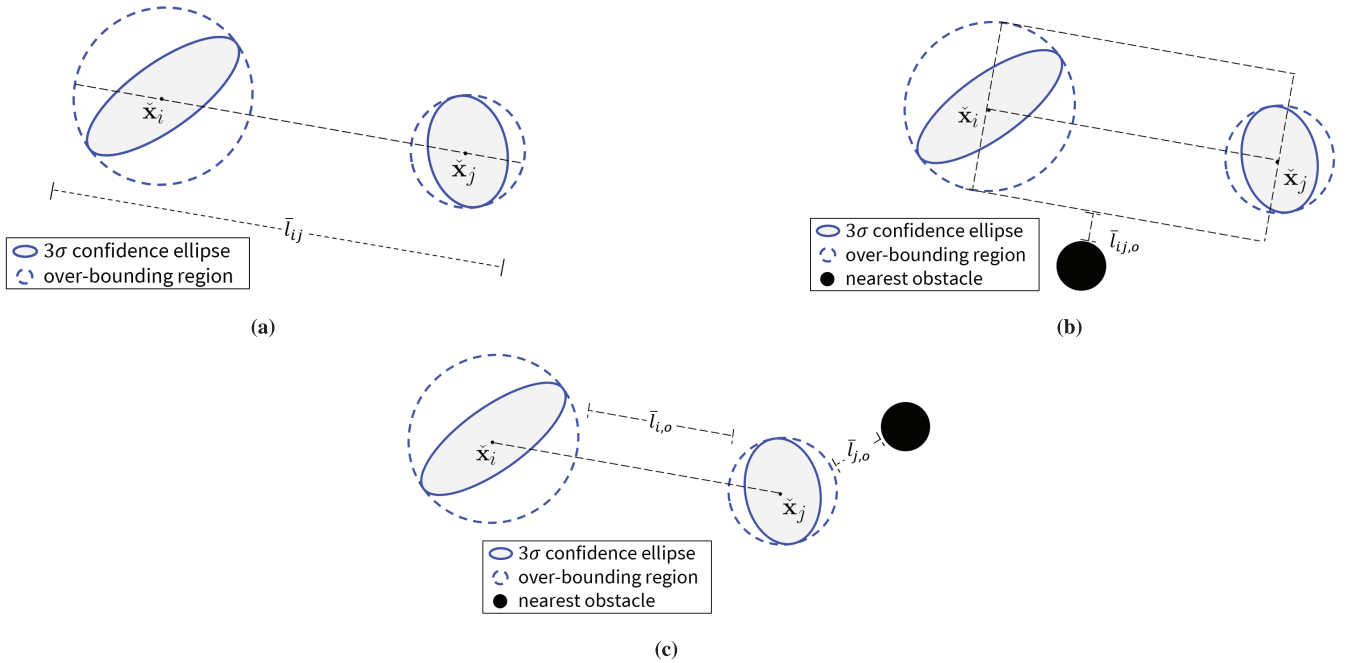
$$\bar{l}_{ij,o} = \|\mathbf{x}_{l,ij} - \mathbf{x}_{\beta,ij}\|_2 - s\sqrt{\max(\bar{\lambda}^{\Sigma_i}, \bar{\lambda}^{\Sigma_j})} \quad (15)$$

where  $\mathbf{x}_{\beta,ij}$  is the closest obstacle point,  $s$  is a scalar value reflecting the desired confidence level (Hoover, 1984; similar to Equation [14]),  $\bar{\lambda}^{\Sigma_i}$  and  $\bar{\lambda}^{\Sigma_j}$

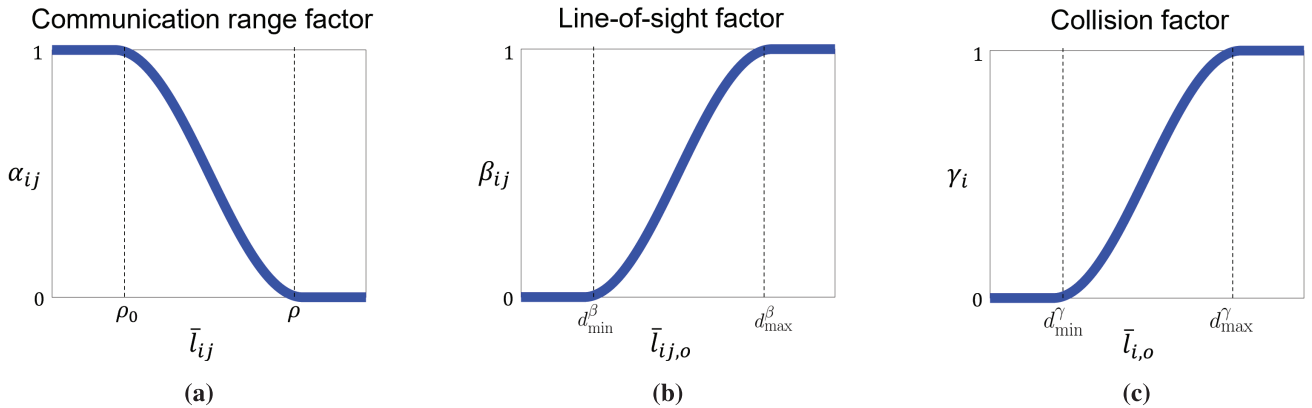
are the largest eigenvalues of the covariance matrices,  $\Sigma_i$  and  $\Sigma_j$ , and  $\mathbf{x}_{l,ij}$  is the point on the line segment connecting  $\check{\mathbf{x}}_i$  and  $\check{\mathbf{x}}_j$  that is closest to  $\mathbf{x}_{\beta,ij}$ , and can be expressed as:

$$\mathbf{x}_{l,ij} = \zeta \check{\mathbf{x}}_i + (1 - \zeta) \check{\mathbf{x}}_j \quad (16)$$

where  $\zeta \in [0, 1]$ . Note that finding the closest obstacle point,  $\mathbf{x}_{\beta,ij}$ , can be computationally challenging in practice, especially in the presence of a large number of obstacles. Techniques such as efficient nearest neighbor searches (Atramentov & LaValle, 2002) have been explored to reduce this computational load and are beyond the scope of this paper.



**FIGURE 3** Conservative measures  $\bar{l}_{ij}$ ,  $\bar{l}_{ij,o}$ ,  $\bar{l}_{i,o}$ , and  $\bar{l}_{j,o}$  used to define our weighted graph in Section 4.1



**FIGURE 4** Factors that define the edge weights of our weighted graph using Equation (12): (a) depicts the communication range factor (see Equation [14]), (b) depicts the line-of-sight factor (see Equation [17]), and (c) depicts the collision factor (see Equation [19]).

Given  $\bar{l}_{ij,o}$  and given parameters for the minimum and maximum desired distances,  $d_{min}^\beta$  and  $d_{max}^\beta$ , from the obstacle, we define the line-of-sight factor in Equation (12) as:

$$\beta_{ij} = \begin{cases} 1 & \bar{l}_{ij,o} > d_{max}^\beta \\ \frac{1}{2} + \frac{1}{2} \cos \left[ \frac{\pi(d_{max}^\beta - \bar{l}_{ij,o})}{d_{max}^\beta - d_{min}^\beta} \right] & d_{min}^\beta < \bar{l}_{ij,o} \leq d_{max}^\beta \\ 0 & \bar{l}_{ij,o} \leq d_{min}^\beta \end{cases} \quad (17)$$

Figure 4(b) shows how  $\beta_{ij}$  varies with  $\bar{l}_{ij,o}$ .

- Collision factor ( $\gamma_i$ ):** This factor indicates how close robot  $i$  is to a collision. Here, again, we define a conservative measure,  $\bar{l}_{i,o}$ , of the distance from the robot to the nearest possible collision point. This nearest collision point can be a neighboring robot or a nearby obstacle. As mentioned above, techniques to reduce the computational load in obtaining the nearest collision point are beyond the scope of this paper. Figure 3(c) illustrates the conservative measure for two robots,  $i$  and  $j$  (i.e.,  $\bar{l}_{i,o}$  and  $\bar{l}_{j,o}$ ). We define the measure  $\bar{l}_{i,o}$  as:

$$\bar{l}_{i,o} = \|\check{\mathbf{x}}_i - \mathbf{x}_{\gamma,i}\|_2 - s\sqrt{\lambda^{\Sigma_i}} - b_{i,o} \quad (18)$$

where  $\check{\mathbf{x}}_i$  is the nominal position and  $s$  is a scalar value reflecting the desired confidence level (Hoover, 1984; similar to Equation [14]). Here, if the nearest collision point for robot  $i$  is robot  $j$ , then  $\mathbf{x}_{\gamma,i} = \check{\mathbf{x}}_j$  and  $b_{i,o} = s\sqrt{\lambda^{\Sigma_j}}$ . Otherwise, if the nearest collision point for robot  $i$  is an obstacle, then  $\mathbf{x}_{\gamma,i}$  is the center of the obstacle and  $b_{i,o}$  represents the width of the obstacle. Given  $\bar{l}_{i,o}$  and given parameters for the minimum and maximum desired distances,  $d_{min}^\gamma$  and  $d_{max}^\gamma$ , we define the collision factor in Equation (12) as:

$$\gamma_i = \begin{cases} 1 & \bar{l}_{i,o} > d_{max}^\gamma \\ \frac{1}{2} + \frac{1}{2} \cos \left[ \frac{\pi(d_{max}^\gamma - \bar{l}_{i,o})}{d_{max}^\gamma - d_{min}^\gamma} \right] & d_{min}^\gamma < \bar{l}_{i,o} \leq d_{max}^\gamma \\ 0 & \bar{l}_{i,o} \leq d_{min}^\gamma \end{cases} \quad (19)$$

Figure 4(c) shows how  $\gamma_i$  varies with  $\bar{l}_{i,o}$ . Note that, similar to  $\gamma_i$ , the collision factor,  $\gamma_j$ , indicates how close robot  $j$  is to a collision.

Thus, given the nominal position,  $\check{\mathbf{x}}_i$ , and covariance matrix,  $\Sigma_i$ , for all robots in the system, Equations (12)–(19) can be used to obtain the edge weights for our weighted graph. The main objective of these edge weights is to model the connectivity constraints listed in Section 3.2. If desired, alternate functions (such as exponential-based functions presented in Sabattini et al. [2013] and Yang et al. [2010]) can be used to model these constraints as long as the edge weights remain in the range [0, 1] (as described in Section 2) and are differentiable (required by our gradient-based controller described later in Section 4.3). The gradients of these edge weights with respect to the robot nominal positions are eventually used to compute robot control inputs.

Note that the factors  $\alpha$  (Equation [14]),  $\beta$  (Equation [17]), and  $\gamma$  (Equation [19]) conservatively represent the connectivity constraints listed in Section 3.2 since these factors depend on the conservative measures defined in Equations (13), (15), and (18). For instance, the conservative measure for the range between two robots

(shown in Figure 3[a]) is typically larger than the true distance (greater than 99.7% of the time for a  $3\sigma$  confidence level), consequently leading the communication range factor,  $\alpha$ , to have a conservative representation of the communication range constraint. Thus, the algebraic connectivity of our weighted graph,  $\lambda_2$ , conservatively measures the true algebraic connectivity of the system,  $\bar{\lambda}_2$ . A detailed analysis comparing  $\lambda_2$  and  $\bar{\lambda}_2$  can be found in our prior work (Shetty et al., 2020). Hence, in order to maintain  $\bar{\lambda}_2 > \epsilon$  according to the connectivity maintenance requirement stated in Section 3, our DCMU algorithm attempts to maintain  $\lambda_2 > \epsilon$ , detailed later in Section 4.3.

## 4.2 | Decentralized Power Iteration

In order to compute the nominal control input for connectivity maintenance, each robot needs to obtain information on how well connected the weighted graph is. To obtain this information in a decentralized manner, we implement a decentralized power iteration method similar to previous works (Sabattini et al., 2013; Yang et al., 2010). Here, each robot communicates in a decentralized manner (i.e., only with other robots it is connected to) and shares information related to its estimate of the system connectivity. Based on the information received from connected robots, each robot is able to gradually improve its estimates of system connectivity. Further details of the method can be found in Sabattini et al. (2013). At the end of each iteration of the power method, each robot  $i$  estimates the following two quantities:

1. The  $i$ -th component of the Fiedler vector (Section 4.1) of the weighted graph,  $\tilde{\mathbf{e}}_{2,i}^{(i)}$ , contains information on how well robot  $i$  is connected to the system.
2. The algebraic connectivity (Section 4.1) of the weighted graph,  $\tilde{\lambda}_{2,i}$ , contains information of how well connected the entire multi-robot system is.

For additional details of the decentralized power iteration method, we refer our readers to Sabattini et al. (2013).

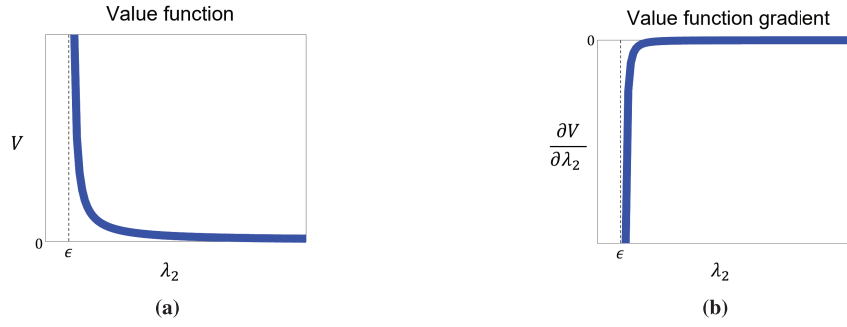
## 4.3 | Decentralized Gradient-Based Controller

As mentioned in Section 3.2, the objective of the controller is to derive the nominal control input,  $\tilde{\mathbf{u}}_{i,t}$ , for each follower robot,  $i$ , to maintain the algebraic connectivity,  $\bar{\lambda}_2$ , above a specified lower limit,  $\epsilon$ . In order to achieve this, we instead maintain the algebraic connectivity of our weighted graph,  $\lambda_2$ , above  $\epsilon$ , which is a conservative representation of the multi-robot system connectivity as explained at the end of Section 4.1.

We use a similar approach to Sabattini et al. (2013), where we first define a value function,  $V$ , that depends on algebraic connectivity of our weighted graph,  $\lambda_2$ , and then design a gradient-based controller for the follower robots to increase  $\lambda_2$ . The value function is defined as:

$$V(\lambda_2) = \begin{cases} \coth(\lambda_2 - \epsilon) & \lambda_2 > \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (20)$$

Figure 5(a) shows the variation of the value function with  $\lambda_2$ . To maintain connectivity in the system, we seek to adjust the robots' nominal positions in order



**FIGURE 5** (a) The value function (Sabattini et al., 2013) defined in Equation (20), along with (b) the gradient of the value function required to compute the nominal control input in Equation (22)

to perform a gradient descent of the value function. This consequently results in increasing the value of  $\lambda_2$  as seen in Figure 5(a). Thus, after simplification of the value function gradient (Sabattini et al., 2013; Yang et al., 2010), the desired change in the robot's nominal position can be expressed as:

$$\Delta \check{\mathbf{x}}_i = \left( -\frac{\partial V(\lambda_2)}{\partial \check{\mathbf{x}}_i} \right) = \left( -\frac{\partial V(\lambda_2)}{\partial \lambda_2} \right) \frac{\partial \lambda_2}{\partial \check{\mathbf{x}}_i} = \left( -\frac{\partial V(\lambda_2)}{\partial \lambda_2} \right) \sum_{j=1}^n \frac{\partial a_{ij}}{\partial \check{\mathbf{x}}_i} \left( \tilde{\mathbf{e}}_{2,i}^{(i)} - \tilde{\mathbf{e}}_{2,j}^{(j)} \right)^2 \quad (21)$$

where the gradient of the value function,  $\frac{\partial V(\lambda_2)}{\partial \lambda_2}$ , is evaluated at  $\tilde{\lambda}_{2,i}$ . Here  $\tilde{\lambda}_{2,i}$ ,  $\tilde{\mathbf{e}}_{2,i}^{(i)}$ , and  $\tilde{\mathbf{e}}_{2,j}^{(j)}$  are obtained from the decentralized power iteration method as explained in Section 4.2. Figure 5(b) shows the gradient of the value function. Note that, as  $\lambda_2$  approaches  $\epsilon$ , the magnitude of the gradient sharply increases, resulting in a large  $\Delta \check{\mathbf{x}}_i$ . This consequently leads to the follower robots moving faster (limited by their maximum velocity) to maintain connectivity when  $\lambda_2$  approaches  $\epsilon$ .

Note that, if desired, alternate value functions can be used instead of Equation (20). Given that the gradient of the value function affects the change in the robots' nominal positions (as shown in Equation [22]; Sabattini et al., 2013) mentions the required properties for this function. It needs to be continuously differentiable, non-negative, and non-increasing  $\forall \lambda_2 > \epsilon$ . Additionally, it needs to suddenly increase as  $\lambda_2$  approaches  $\epsilon$  (resulting in large changes in positions when the system is near loss of connectivity) and it needs to approach a constant value for large values of  $\lambda_2$  (resulting in small changes in positions when the system is well connected).

Next, using Equations (9) and (21), the nominal control input can be obtained as:

$$\check{\mathbf{u}}_i = B^{-1} \Delta \check{\mathbf{x}}_i = \frac{1}{\Delta t} \left( -\frac{\partial V(\lambda_2)}{\partial \lambda_2} \right) \sum_{j=1}^n \frac{\partial a_{ij}}{\partial \check{\mathbf{x}}_i} \left( \tilde{\mathbf{e}}_{2,i}^{(i)} - \tilde{\mathbf{e}}_{2,j}^{(j)} \right)^2 \quad (22)$$

where the control input matrix,  $B = (\Delta t)I$ , as defined in Equation (1). In order to compute the nominal control input,  $\check{\mathbf{u}}_i$ , using the above equation, we require the gradients of the edge weights,  $a_{ij}$ , with respect to the robot nominal position,  $\check{\mathbf{x}}_i$  (i.e.,  $\frac{\partial a_{ij}}{\partial \check{\mathbf{x}}_i}$ ). Given the definition of the edge weight in Equation (12), the required gradient can be expressed as:

$$\frac{\partial a_{ij}}{\partial \check{\mathbf{x}}_i} = \frac{\partial \alpha_{ij}}{\partial \check{\mathbf{x}}_i} \beta_{ij} \gamma_i \gamma_j + \alpha_{ij} \frac{\partial \beta_{ij}}{\partial \check{\mathbf{x}}_i} \gamma_i \gamma_j + \alpha_{ij} \beta_{ij} \frac{\partial \gamma_i}{\partial \check{\mathbf{x}}_i} \gamma_j + \alpha_{ij} \beta_{ij} \gamma_i \frac{\partial \gamma_j}{\partial \check{\mathbf{x}}_i} \quad (23)$$

In the remainder of this section, we proceed to derive the expressions for the gradient of the factors  $\alpha_{ij}$ ,  $\beta_{ij}$ ,  $\gamma_i$ , and  $\gamma_j$  with respect to  $\check{\mathbf{x}}_i$ . From Equation (14), the gradient of  $\alpha_{ij}$  can be expressed as:

$$\frac{\partial \alpha_{ij}}{\partial \check{\mathbf{x}}_i} = -\frac{\pi}{2(\rho - \rho_0)} \sin \left[ \frac{\pi(\bar{l}_{ij} - \rho_0)}{\rho - \rho_0} \right] \frac{\partial \bar{l}_{ij}}{\partial \check{\mathbf{x}}_i} \quad (24)$$

where using Equation (13), the gradient of  $\bar{l}_{ij}$  with respect to the  $m$ -th element of  $\check{\mathbf{x}}_i$  is:

$$\frac{\partial \bar{l}_{ij}}{\partial \check{\mathbf{x}}_i^{(m)}} = \frac{\check{\mathbf{x}}_i^{(m)} - \check{\mathbf{x}}_j^{(m)}}{\|\check{\mathbf{x}}_i - \check{\mathbf{x}}_j\|_2} + \frac{s}{2\sqrt{\lambda^{\Sigma_i}}} \frac{\partial \bar{\lambda}^{\Sigma_i}}{\partial \check{\mathbf{x}}_i^{(m)}} \quad (25)$$

Next, for the gradient of  $\beta_{ij}$ , using Equation (17) we get:

$$\frac{\partial \beta_{ij}}{\partial \check{\mathbf{x}}_i} = \frac{\pi}{2(d_{max}^\beta - d_{min}^\beta)} \sin \left[ \frac{\pi(d_{max}^\beta - \bar{l}_{ij,o})}{d_{max}^\beta - d_{min}^\beta} \right] \frac{\partial \bar{l}_{ij,o}}{\partial \check{\mathbf{x}}_i} \quad (26)$$

where, using Equation (15), the gradient of  $\bar{l}_{ij,o}$  with respect to the  $m$ -th element of  $\check{\mathbf{x}}_i$  is:

$$\frac{\partial \bar{l}_{ij,o}}{\partial \check{\mathbf{x}}_i^{(m)}} = \zeta \frac{(\check{\mathbf{x}}_i^{(m)} - \mathbf{x}_o^{(m)})}{\|\check{\mathbf{x}}_i - \mathbf{x}_o\|_2} - \frac{s}{2\sqrt{\max(\bar{\lambda}^{\Sigma_i}, \bar{\lambda}^{\Sigma_j})}} \frac{\partial \max(\bar{\lambda}^{\Sigma_i}, \bar{\lambda}^{\Sigma_j})}{\partial \check{\mathbf{x}}_i^{(m)}} \quad (27)$$

Using Equation (19), the gradient of  $\gamma_i$  can be expressed as:

$$\frac{\partial \gamma_i}{\partial \check{\mathbf{x}}_i} = \frac{\pi}{2(d_{max}^\gamma - d_{min}^\gamma)} \sin \left[ \frac{\pi(d_{max}^\gamma - \bar{l}_{i,o})}{d_{max}^\gamma - d_{min}^\gamma} \right] \frac{\partial \bar{l}_{i,o}}{\partial \check{\mathbf{x}}_i} \quad (28)$$

where, using Equation (18), the gradient of  $\bar{l}_{i,o}$  with respect to the  $m$ -th element of  $\check{\mathbf{x}}_i$  is:

$$\frac{\partial \bar{l}_{i,o}}{\partial \check{\mathbf{x}}_i^{(m)}} = \frac{(\check{\mathbf{x}}_i^{(m)} - \mathbf{x}_o^{(m)})}{\|\check{\mathbf{x}}_i - \mathbf{x}_o\|_2} - \frac{s}{2\sqrt{\lambda^{\Sigma_i}}} \frac{\partial \bar{\lambda}^{\Sigma_i}}{\partial \check{\mathbf{x}}_i^{(m)}} \quad (29)$$

Since  $\gamma_j$  indicates how close robot  $j$  is to a collision, its gradient with respect to  $\check{\mathbf{x}}_i$  is non-zero only when the closest collision point for robot  $j$  is robot  $i$ . As explained below Equation (18), this implies  $\mathbf{x}_{\gamma,j} = \check{\mathbf{x}}_i$  and  $b_{j,o} = s\sqrt{\lambda^{\Sigma_i}}$ . Thus, if the closest collision point for robot  $j$  is robot  $i$ , then, using Equation (19), the gradient of  $\gamma_j$  can be expressed as:

$$\frac{\partial \gamma_j}{\partial \check{\mathbf{x}}_i} = \frac{\pi}{2(d_{max}^\gamma - d_{min}^\gamma)} \sin \left[ \frac{\pi(d_{max}^\gamma - \bar{l}_{j,o})}{d_{max}^\gamma - d_{min}^\gamma} \right] \frac{\partial \bar{l}_{j,o}}{\partial \check{\mathbf{x}}_i} \quad (30)$$

where, using Equation (18), the gradient of  $\bar{l}_{j,o}$  with respect to the  $m$ -th element of  $\check{\mathbf{x}}_i$  is:

$$\frac{\partial \bar{l}_{j,o}}{\partial \check{\mathbf{x}}_i^{(m)}} = -\frac{(\check{\mathbf{x}}_j^{(m)} - \check{\mathbf{x}}_i^{(m)})}{\|\check{\mathbf{x}}_j - \check{\mathbf{x}}_i\|_2} - \frac{s}{2\sqrt{\lambda^{\Sigma_i}}} \frac{\partial \bar{\lambda}^{\Sigma_i}}{\partial \check{\mathbf{x}}_i^{(m)}} \quad (31)$$

In order to compute the gradients of the factors using Equations (24), (26), (28), and (30), we need to derive the gradients of the eigenvalues of covariance matrices  $\Sigma_i$  and  $\Sigma_j$  required in Equations (25), (27), (29), and (31). The equations in Section 3.1 show how the covariance matrix for a robot,  $\Sigma_i$ , varies with its position,  $\check{\mathbf{x}}_i$ . From Equation (10), note that  $\Sigma_i = P_i + \Lambda_i$ . Thus, as the robot nominal position,  $\check{\mathbf{x}}_i$ , changes from time instant  $t$  to  $t+1$ , the covariance matrix updates to  $\Sigma_{i,t+1}$  as:

$$\Sigma_{i,t+1} = P_{i,t+1} + \Lambda_{i,t+1} \quad (32)$$

$$= \bar{P}_{i,t+1} - G_{i,t+1} \bar{P}_{i,t+1} + (I - BK_{i,t+1}) \Lambda_{i,t} (I - BK_{i,t+1})^\top + G_{i,t+1} \bar{P}_{i,t+1} \quad (33)$$

$$= P_{i,t} + Q_{i,t+1} + (I - BK_{i,t+1}) \Lambda_{i,t} (I - BK_{i,t+1})^\top \quad (34)$$

Note that the updated covariance matrix,  $\Sigma_{i,t+1}$ , only depends on  $P_{i,t}$ ,  $Q_{i,t+1}$ ,  $B$ ,  $K_{i,t+1}$ , and  $\Lambda_{i,t}$ , and does not depend on the new nominal position,  $\check{\mathbf{x}}_{i,t+1}$ . Consequently, the eigenvalue  $\bar{\lambda}^{\Sigma_i}$  also does not depend on  $\check{\mathbf{x}}_i$ , which implies  $\frac{\partial \bar{\lambda}^{\Sigma_i}}{\partial \check{\mathbf{x}}_i^{(m)}} = 0$ . Thus, the terms involving gradients of the eigenvalues in Equations (25), (27), (29), and (31) are equal to zero.

In summary, our DCMU algorithm obtains nominal control inputs,  $\check{\mathbf{u}}_i$ , for the follower robots using Equation (22), which relies on quantities estimated using the decentralized power iteration method mentioned in Section 4.2 and relies on gradients of our weighted graph edge weights derived in Equations (23)–(32). In Equation (22), note that the magnitude of the nominal control input,  $\check{\mathbf{u}}_i$ , depends directly on the magnitude of the value function gradient (shown in Figure 5[b]), which grows large as the system algebraic connectivity decreases. Thus, when the system is close to losing connectivity, Equation (22) obtains large nominal control inputs,  $\check{\mathbf{u}}_i$ , for the follower robots to maintain connectivity.

Note that the connectivity maintenance performance of our DCMU algorithm depends on the nominal trajectories of the leader robots, which are assumed to be obtained from a high-level planner as mentioned in Section 3.2. While our algorithm attempts to move the follower robots such that connectivity is maintained, in some cases, the nominal trajectories of the leader robots might be such that connectivity maintenance is not possible—for example, a system with two leader robots and one follower robot in which the leader robots move indefinitely in opposite directions. On the other hand, our algorithm does guarantee collision avoidance for the robots with the desired confidence level (greater than 99.7% for a  $3\sigma$  confidence level). If a robot,  $i$ , gets close to a collision point, the corresponding collision factor,  $\gamma_i$ , decreases, consequently decreasing the weighted graph algebraic connectivity. The nominal control input,  $\check{\mathbf{u}}_i$ , obtained using Equation (22) then attempts to move the robot away from the collision point. We validate the connectivity maintenance and collision avoidance performance of our DCMU algorithm on a few multi-robot systems below in Section 5.

## 5 | SIMULATION RESULTS

In this section, we validate the performance of our DCMU algorithm in two 2D simulation setups: MATLAB and AirSim (Shah et al., 2018). We, first, present the MATLAB simulations that allowed us to quantitatively compare our algorithm with related previous work in Sabattini et al. (2013), which we refer to as *the Sabattini algorithm* for simplicity. We, then, discuss the AirSim simulations that validate our algorithm on a high-fidelity simulator in which the motion of the robots was

more realistically simulated. For both simulation setups, we first demonstrated our DCMU algorithm on a simple two-robot setup with one leader robot and one follower robot, and then proceeded to more complex multi-robot configurations. As mentioned in Section 3.2, the leader robots were given predefined nominal trajectories (which we assume are available from some high-level planner such as an exploration strategy), and the follower robots implement nominal control inputs from our DCMU algorithm (i.e., from Equation [22]). A video containing the simulation results can be viewed online at [https://youtu.be/SbE-ejQ\\_zm8](https://youtu.be/SbE-ejQ_zm8).

For both our simulation setups, we set the discrete time-step  $\Delta t = 0.2$  s. We assumed an inter-robot communication rate of 1,000 Hz for the decentralized power iteration method. The motion and sensing model covariance matrices were set as  $Q_{i,t} = (0.02 \text{ m}^2)I$  and  $R_{i,t} = (5 \text{ m}^2)I$ , respectively. These covariance matrices were used to add zero-mean Gaussian-distributed errors to the state and measurement vectors (Equations [1] and [2]) for both the leader and the follower robots in the system. The initial state estimation covariance matrix was set as  $P_{i,t} = (0.1 \text{ m}^2)I$  and the control feedback gain used in Equation (8) was set as  $\check{K}_i = (0.14)I$ .

For a  $3\sigma$  confidence level (reflecting a  $\sim 99.7\%$  confidence level) in our weighted graph, we obtained the scalar value  $s = 3.494$  using the chi-square distribution (Hoover, 1984). For the MATLAB simulations, we limited the maximum velocity control input (in each direction) in the motion model (Equation [1]) to be  $2 \text{ ms}^{-1}$  in order to reflect the physical constraints of practical robots. The AirSim simulator inherently implements maximum velocity constraints while simulating the robot motion. We specified the lower limit for the algebraic connectivity of the system to be  $\epsilon = 0.01$ , where the connectivity maintenance objective was to maintain  $\bar{\lambda}_2 > \epsilon$ .

For the weighted graph in Section 4.1, the parameters  $\rho$ ,  $d_{min}^\beta$ , and  $d_{min}^\gamma$  represent the maximum desired communication range between robots, the minimum desired distance between robot line-of-sight vectors and obstacles, and the minimum desired distance between robots and their nearest collision points, respectively. The parameters  $\rho_0$ ,  $d_{max}^\beta$ , and  $d_{max}^\gamma$  represent the distances at which the communication range factor, the line-of-sight factor, and the collision factors, respectively, start affecting the nominal control input.

For instance, the communication range factor between two robots affects their nominal control inputs only when the conservative measure in Equation (13) is greater than  $\rho_0$ . Thus, setting the value of  $\rho_0$  close to  $\rho$  results in the communication range factor affecting the nominal control inputs only when the conservative measure (Equation [13]) is close to  $\rho$ . Setting  $\rho_0$  to be much smaller than  $\rho$  would result in the nominal control inputs being affected much earlier. In practice, these parameters can be set based on the desired performance of the multi-robot system. For our simulations, we heuristically set these parameter values as  $\rho = 20$  m,  $\rho_0 = 18$  m,  $d_{max}^\beta = 3$  m,  $d_{min}^\beta = 1$  m,  $d_{max}^\gamma = 3$  m, and  $d_{min}^\gamma = 1$  m.

## 5.1 | MATLAB Simulations

For the MATLAB simulations, our primary objective was to compare the connectivity maintenance performance of our algorithm with the Sabattini algorithm (Sabattini et al., 2013) for various multi-robot system configurations. As mentioned in the beginning of Section 4, previous works (Gasparri et al., 2017; Giordano et al., 2013; Sabattini et al., 2013; Siligardi et al., 2019; Yang et al., 2010) in essence assumed that the robots were already at their desired nominal positions. In contrast, our DCMU algorithm accounts for the deviations arising due to motion and sensing uncertainties.

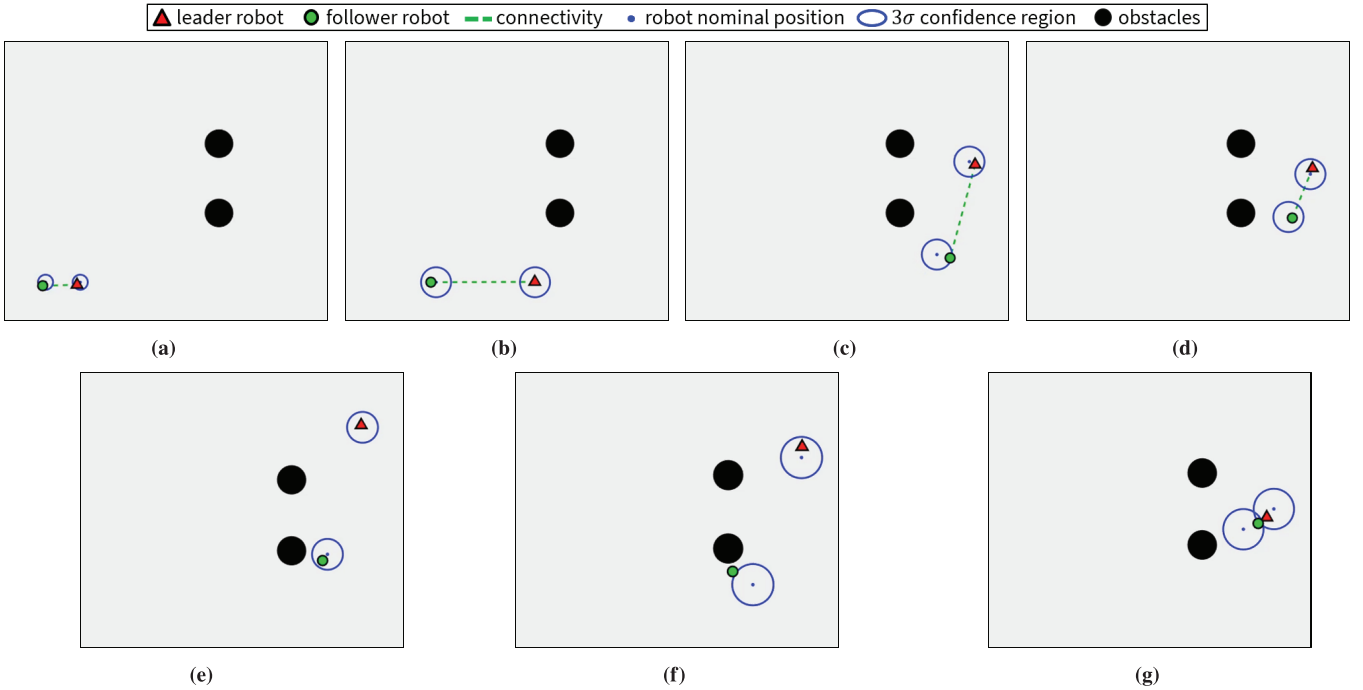


FIGURE 6 Snapshots of two-robot simulation runs: (a)–(d) represent single runs in which the follower robot implements the DCMU algorithm and maintains connectivity, while (e)–(g) represent different simulation runs in which the follower robot implements the Sabattini algorithm and connectivity is lost.

As shown in Figure 6, we first analyzed a simple two-robot setup with one leader robot and one follower. The leader robot was provided a nominal trajectory that covered a distance of 120 m in a duration of 120 s. Figure 6(a)–(d) show snapshots of a single simulation run in which the follower robot implemented our DCMU algorithm to maintain connectivity with the leader robot. Starting from an initial position (Figure 6[a]), the follower robot follows the leader robot to stay within communication range (Figure 6[b]). As the leader robot moves upwards, the follower robot maintains line-of-sight connectivity while maintaining a safe distance from the obstacles (Figure 6[c]), while also maintaining a safe distance from the leader robot, itself (Figure 6[d]). Figure 6(e)–(g) show snapshots of different simulation runs where the follower robot implements the Sabattini algorithm (i.e., not accounting for the motion and sensing uncertainties [Sabattini et al., 2013]). We observed that not accounting for these uncertainties could potentially lead to a loss of connection due to the robots getting further away from the communication range (Figure 6[e]), the robots losing line-of-sight communication (Figure 6[f]), or due to a collision (Figure 6[g]).

In Figure 7, we quantitatively compare the connectivity maintenance performance for the above two-robot setup with the Sabattini algorithm (Sabattini et al., 2013). Here, we varied the amount of motion and sensing uncertainties in the multi-robot system and performed 1,000 simulation runs for each scenario for both our DCMU algorithm and the Sabattini algorithm. For both of these algorithms, we compared the ratio of runs for which the algebraic connectivity based on the actual robot positions,  $\bar{\lambda}_2$ , was maintained above the specified lower limit  $\epsilon = 0.01$  throughout the run, as stated in the problem formulation in Section 3. Note that, for a given set of nominal trajectories for the leader robots, the nominal trajectories for the follower robots obtained using our DCMU algorithm

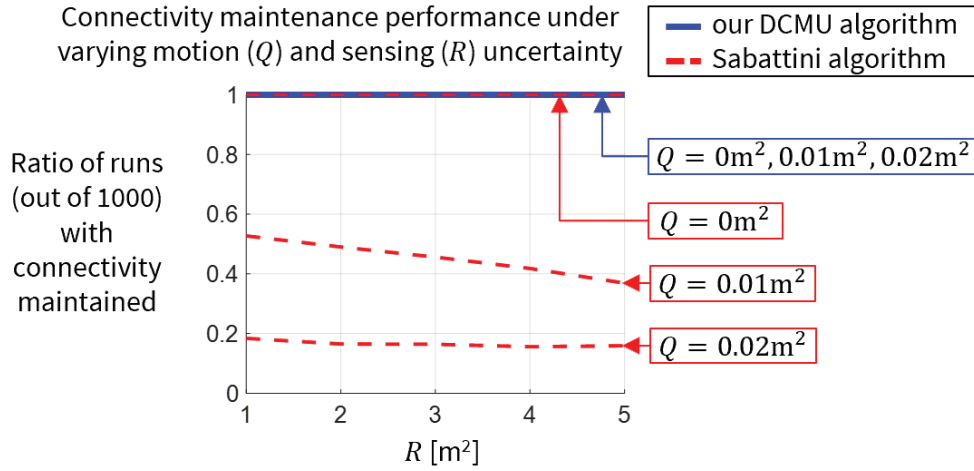


FIGURE 7 Quantitative comparison for the two-robot MATLAB simulation shown in Figure 6; our DCMU algorithm performs better than the Sabattini algorithm (Sabattini et al., 2013) under increasing motion and sensing uncertainties.

remained the same. However, the actual robot trajectories varied across the 1,000 simulation runs due to the presence of Gaussian-distributed motion and sensing uncertainties. We began with the trivial case of zero motion model uncertainty, i.e.,  $Q_{i,t} = (0 \text{ m}^2)I$  where the robots follow the nominal trajectories exactly. Here, we could observe that our DCMU algorithm performed the same as the Sabattini algorithm and maintained  $\bar{\lambda}_2 > \epsilon$  throughout all 1,000 simulation runs. We, then, compared the connectivity maintenance performance for the following motion and sensing model covariance matrices:  $Q_{i,t} = (0.01 \text{ m}^2)I$  and  $Q_{i,t} = (0.02 \text{ m}^2)I$ ;  $R_{i,t} = (1 \text{ m}^2)I$ ,  $R_{i,t} = (2 \text{ m}^2)I$ ,  $R_{i,t} = (3 \text{ m}^2)I$ ,  $R_{i,t} = (4 \text{ m}^2)I$ , and  $R_{i,t} = (5 \text{ m}^2)I$ . Our DCMU algorithm maintained  $\bar{\lambda}_2 > \epsilon$  throughout all 1,000 simulation runs for the different motion and sensing uncertainties. However, as the amount of uncertainty increased, we observed that the connectivity maintenance performance of the Sabattini algorithm decreased sharply.

We, then, compared our algorithm with the Sabattini algorithm on different multi-robot system configurations as shown in Figure 8. Snapshots of a single simulation run with follower robots using our DCMU algorithm are shown for three different configurations: Figure 8(a)–(c), Figure 8(d)–(f), and Figure 8(g)–(i). The nominal trajectories for the leader robots covered distances of 80 m in 80 s, 60 m in 60 s, and 60 m in 60 s, respectively, for the three configurations. Note that the follower robots were not assigned to follow specific leader robots. Instead, our DCMU algorithm computed nominal control inputs for the follower robots directly based on each robot's estimate of the system connectivity. Thus, in Figure 8, we can observe that the follower robots rearranged themselves in order to maintain connectivity within the system while accounting for the deviations arising due to motion and sensing uncertainties. Figure 9 quantitatively compares the connectivity maintenance performance of our DCMU algorithm and the Sabattini algorithm for 1,000 simulation runs of each of the three multi-robot configurations. Similar to the two-robot case, we observed that our algorithm was able to maintain  $\bar{\lambda}_2 > \epsilon$  throughout all 1,000 runs, whereas the ratio of runs for which Sabattini algorithm maintained  $\bar{\lambda}_2 > \epsilon$  throughout the runs sharply decreased as the amount of motion and sensing uncertainties increased. These simulations validate the applicability of our algorithm towards maintaining connectivity in the system under robot motion and sensing uncertainties.

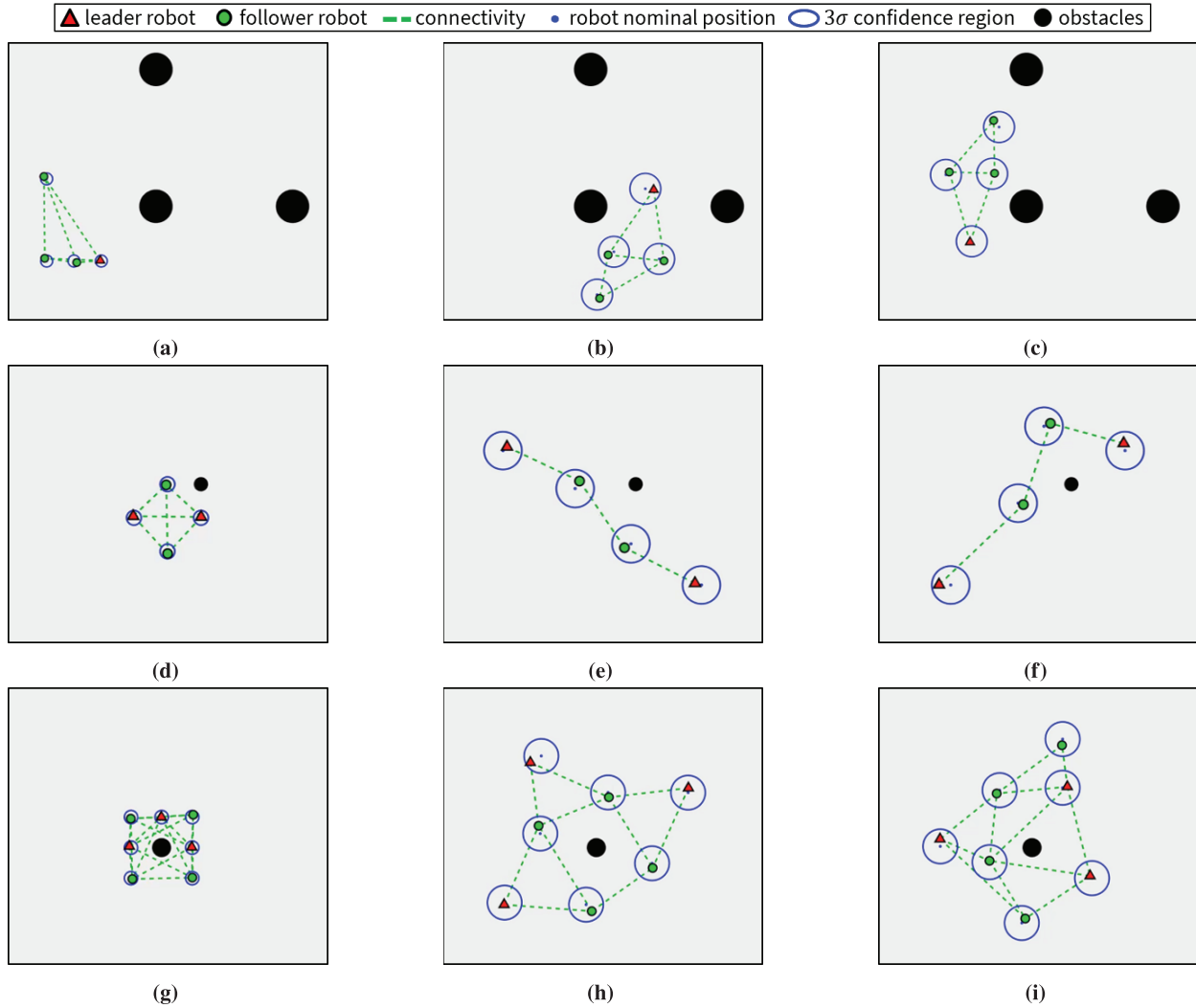


FIGURE 8 MATLAB simulations for various multi-robot system configurations via snapshots of simulation runs in which the follower robots implement the DCMU algorithm in three different configurations: (a)–(c), (d)–(f), and (g)–(i); the simulation videos can be viewed online at [https://www.youtube.com/watch?v=SbE-ejQ\\_zm8](https://www.youtube.com/watch?v=SbE-ejQ_zm8)

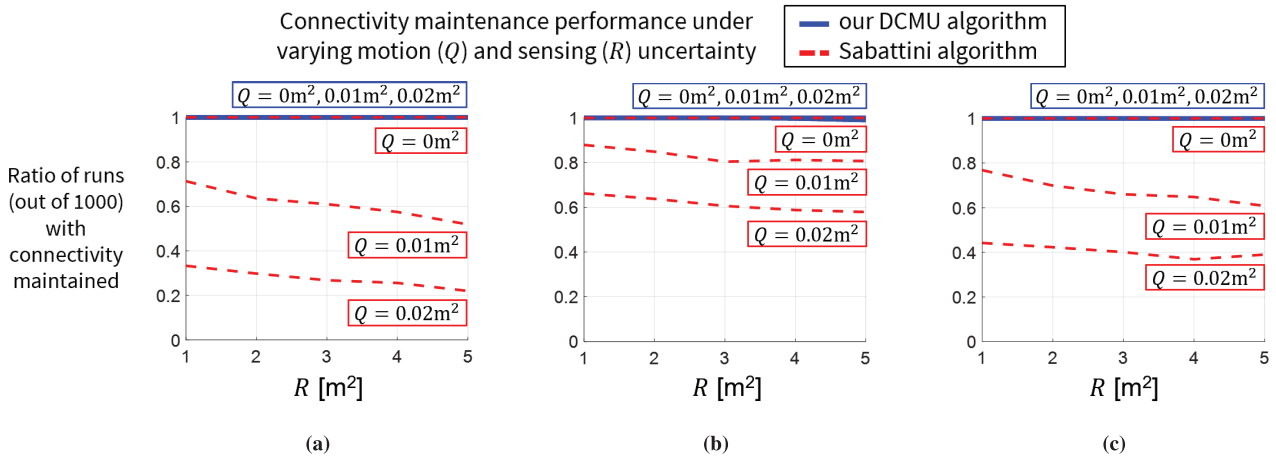


FIGURE 9 Quantitative comparisons for the multi-robot MATLAB simulations shown in Figure 8; the DCMU algorithm performs better than the Sabattini algorithm (Sabattini et al., 2013) under increasing motion and sensing uncertainties for the three multi-robot configurations.

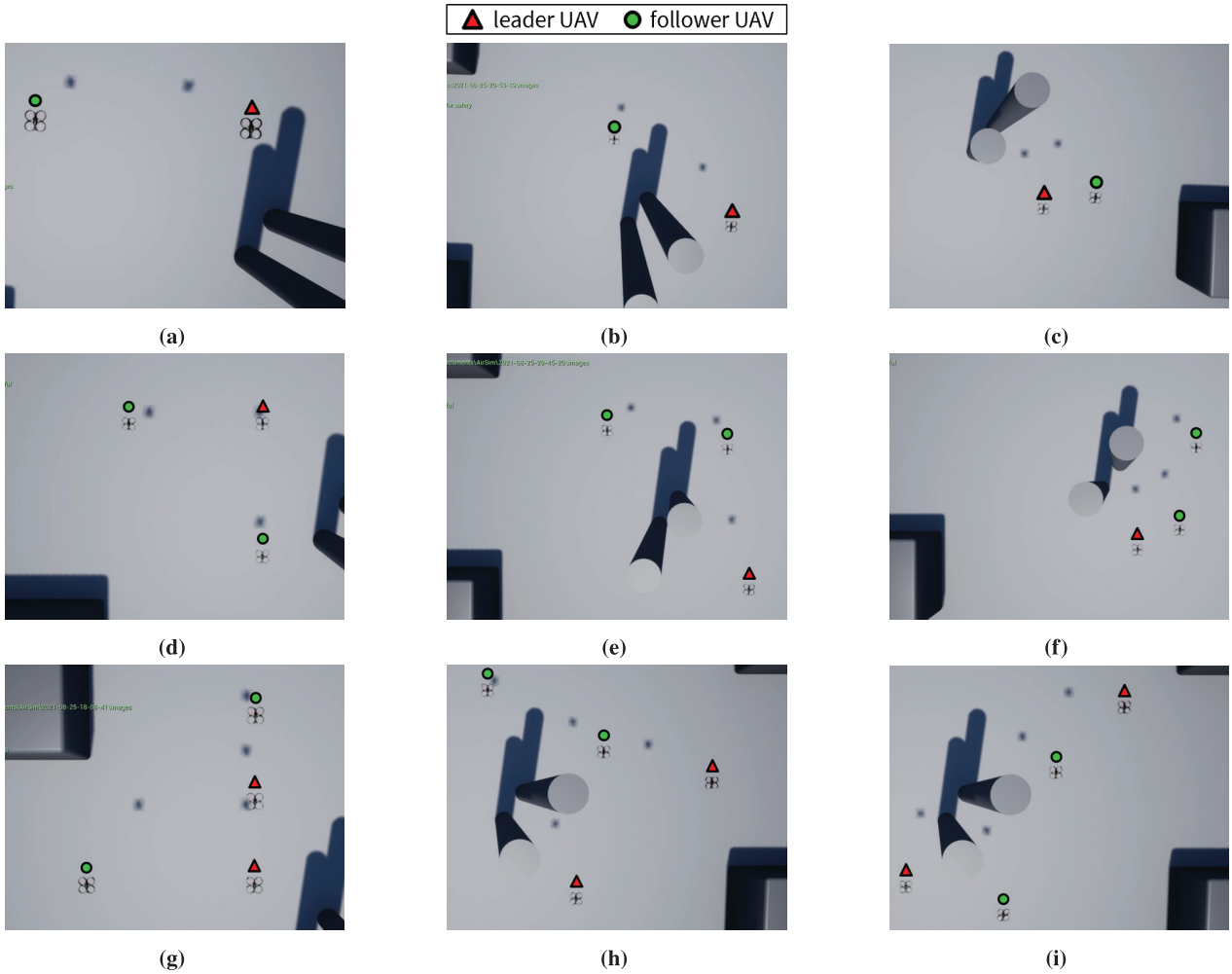


FIGURE 10 AirSim simulations for various multi-UAV system configurations via snapshots of simulation runs in which the follower UAVs implement the DCMU algorithm in three different configurations: (a)–(c), (d)–(f), and (g)–(i); the simulation videos can be viewed online at [https://www.youtube.com/watch?v=SbE-ejQ\\_zm8](https://www.youtube.com/watch?v=SbE-ejQ_zm8)

## 5.2 | AirSim Simulations

For the AirSim simulations, we validated our algorithm on multi-robot systems comprised of unmanned aerial vehicles (UAVs). The motion of the UAVs was modeled using Equation (1), while sensing uncertainties were introduced according to Equation (2). We show snapshots with the follower UAVs using our DCMU algorithm for three different configurations: Figure 10(a)–(c), Figure 10(d)–(f), and Figure 10(g)–(i). The nominal trajectories for the leader robots covered distances of 105 m in 105 s, 180 m in 180 s, and 45 m in 90 s, respectively, for the three configurations. The main objective of the snapshots shown in Figure 10 was to demonstrate that our DCMU algorithm is able to compute nominal trajectories for the follower UAVs such that connectivity is maintained for different multi-UAV configurations while the leader UAVs follow their predefined nominal trajectories. Figure 11 shows how the algebraic connectivity,  $\bar{\lambda}_2$ , is maintained above the lower limit,  $\epsilon$ , throughout the simulation for the different configurations. Note that the AirSim simulations were executed in real time, thus validating the real-time applicability of our DCMU algorithm.

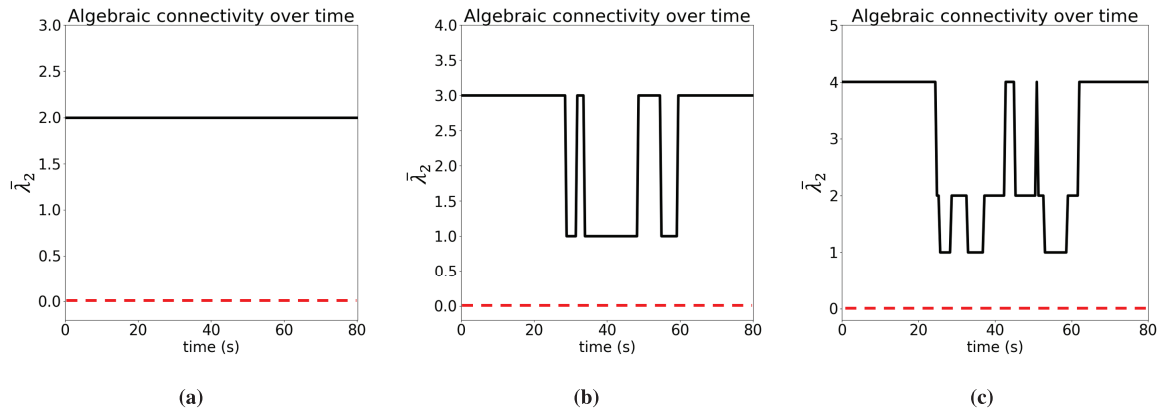


FIGURE 11 Algebraic connectivity of the multi-UAV AirSim simulations shown in Figure 10; the DCMU algorithm maintains  $\bar{\lambda}_2$  (black lines) above the lower limit  $\epsilon$  (red-dashed lines).

## 6 | CONCLUSION

In this paper, we presented the decentralized connectivity maintenance algorithm that accounts for robot motion and sensing uncertainties (DCMU). We, first, defined a weighted graph to account for these uncertainties along with constraints such as a maximum communication range, line-of-sight communication, and collision avoidance. Next, we designed a decentralized gradient-based controller by deriving the gradients of our weighted graph edge weights. Finally, we validated the connectivity maintenance performance of our algorithm in two simulation setups: MATLAB and AirSim. We quantitatively compared our DCMU algorithm with previous related work (Sabattini et al., 2013) and demonstrated that our algorithm performed better under motion and sensing uncertainties. Additionally, we demonstrated the connectivity maintenance performance of our algorithm on AirSim, which is a higher-fidelity simulator compared to MATLAB.

Possible directions for future work include exploring alternate geometric representations that reduce the conservatism in our weighted graph, evaluating connectivity maintenance performance for a range of graph parameter values, and accounting for more complex robot motion models such as ground robots with nonholonomic constraints. Additionally, we plan to evaluate our DCMU algorithm on real hardware and analyze the effects of communication latency on system connectivity maintenance.

## REFERENCES

- Alanwar, A., Said, H., & Althoff, M. (2019). Distributed secure state estimation using diffusion Kalman filters and reachability analysis. *IEEE 58th Conference on Decision and Control (CDC)*, Nice, France. <https://doi.org/10.1109/CDC40024.2019.9029929>
- Atramentov, A., & LaValle, S. M. (2002). Efficient nearest neighbor searching for motion planning. *Proceedings 2002 IEEE International Conference on Robotics and Automation*, 632–637. <https://doi.org/10.1109/ROBOT.2002.1013429>
- Baxter, J. L., Burke, E. K., Garibaldi, J. M., & Norman, M. (2007). Multi-robot search and rescue: a potential field based approach. In S. C. Mukhopadhyay & G. S. Gupta (Eds.), *Autonomous robots and agents* (pp. 9–16). Springer. [https://doi.org/10.1007/978-3-540-73424-6\\_2](https://doi.org/10.1007/978-3-540-73424-6_2)
- Bhamidipati, S., & Gao, G. X. (2019). Locating multiple GPS jammers using networked UAVs. *IEEE Internet of Things Journal*, 6(2), 1816–1828. <https://doi.org/10.1109/JIOT.2019.2896262>
- Bry, A., & Roy, N. (2011). Rapidly-exploring random belief trees for motion planning under uncertainty. *IEEE International Conference on Robotics and Automation*, Shanghai, China. <http://doi.org/10.1109/ICRA.2011.5980508>
- Burgard, W., Moors, M., Stachniss, C., & Schneider, F. E. (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3), 376–386. <https://doi.org/10.1109/TRO.2004.839232>

- Cortés, J., & Egerstedt, M. (2017). Coordinated control of multi-robot systems: a survey. *SICE Journal of Control, Measurement, and System Integration*, 10(6), 495–503. <https://doi.org/10.9746/jcmsi.10.495>
- Doyle, J. C., Francis, B. A., & Tannenbaum, A. R. (2013). *Feedback control theory*. Courier Corporation.
- Gasparri, A., Sabattini, L., & Ulivi, G. (2017). Bounded control law for global connectivity maintenance in cooperative multirobot systems. *IEEE Transactions on Robotics*, 33(3), 700–717. <https://doi.org/10.1109/TRO.2017.2664883>
- Giordano, P. R., Franchi, A., Secchi, C., & Bühlhoff, H. H. (2013). A passivity-based decentralized strategy for generalized connectivity maintenance. *The International Journal of Robotics Research*, 32(3), 299–323. <https://doi.org/10.1177/0278364912469671>
- Grone, R., Merris, R., & Sunder, V. S. (1990). The Laplacian spectrum of a graph. *SIAM Journal on Matrix Analysis and Applications*, 11(2), 218–238. <https://www.math.ucdavis.edu/~saito/data/graphlap/grone-merris-sunder-lapeig.pdf>
- Hoover, W. E. (1984). *Algorithms for confidence circles and ellipses* [Technical report]. National Oceanic and Atmospheric Administration (NOAA). [https://www.ngs.noaa.gov/PUBS\\_LIB/AlgorithmsForConfidenceCirclesAndEllipses\\_TR\\_NOS107\\_CGS3.pdf](https://www.ngs.noaa.gov/PUBS_LIB/AlgorithmsForConfidenceCirclesAndEllipses_TR_NOS107_CGS3.pdf)
- Khateri, K., Pourgholi, M., Montazeri, M., & Sabattini, L. (2019). A comparison between decentralized local and global methods for connectivity maintenance of multi-robot networks. *IEEE Robotics and Automation Letters*, 4(2), 633–640. <https://doi.org/10.1109/LRA.2019.2892552>
- Lee, D., Franchi, A., Son, H. I., Ha, C., Bühlhoff, H. H., & Giordano, P. R. (2013). Semiautonomous haptic teleoperation control architecture of multiple unmanned aerial vehicles. *IEEE/ASME Transactions on Mechatronics*, 18(4), 1334–1345. <https://doi.org/10.1109/TMECH.2013.2263963>
- Park, H., & Hutchinson, S. (2018). Robust rendezvous for multi-robot system with random node failures: an optimization approach. *Autonomous Robots*, 42(8), 1807–1818. <https://doi.org/10.1007/s10514-018-9715-8>
- Rizk, Y., Awad, M., & Tunstel, E. W. (2019). Cooperative heterogeneous multi-robot systems: a survey. *ACM Computing Surveys*, 52(2), 1–31. <https://doi.org/10.1145/3303848>
- Sabattini, L., Chopra, N., & Secchi, C. (2013). Decentralized connectivity maintenance for cooperative control of mobile robotic systems. *The International Journal of Robotics Research*, 32(12), 1411–1423. <https://doi.org/10.1177/0278364913499085>
- Shah, S., Dey, D., Lovett, C., & Kapoor, A. (2018). Airsim: high-fidelity visual and physical simulation for autonomous vehicles. In M. Hutter & R. Siegwart (Eds.), *Field and service robotics* (pp. 621–635). Springer. [https://doi.org/10.1007/978-3-319-67361-5\\_40](https://doi.org/10.1007/978-3-319-67361-5_40)
- Shetty, A., Hussain, T., & Gao, G. (2021). Decentralized connectivity maintenance for multi-robot systems under motion and sensing uncertainties. *Proc. of the 34th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2021)*, 2445–2458. <https://www.ion.org/publications/abstract.cfm?articleID=17943>
- Shetty, A., Knowles, D., & Gao, G. X. (2020). Connectivity maintenance for multi-robot systems under motion and sensing uncertainties using distributed ADMM-based trajectory planning. <https://doi.org/10.48550/arXiv.2012.09808>
- Siligardi, L., Panerati, J., Kaufmann, M., Minelli, M., Ghedini, C., Beltrame, G., & Sabattini, L. (2019). Robust area coverage with connectivity maintenance. *International Conference on Robotics and Automation (ICRA)*, Montreal, Canada. <https://doi.org/10.1109/ICRA.2019.8793555>
- Soukieh, R., Shames, I., & Fidan, B. (2009). Obstacle avoidance of robotic formations based on fluid mechanical modeling. *European Control Conference (ECC)*, Budapest, Hungary. <https://doi.org/10.23919/ECC.2009.7074908>
- Yang, P., Freeman, R. A., Gordon, G. J., Lynch, K. M., Srinivasa, S. S., & Sukthankar, R. (2010). Decentralized estimation and control of graph connectivity for mobile sensor networks. *Automatica*, 46(2), 390–396. <https://doi.org/10.1016/j.automatica.2009.11.012>

**How to cite this article:** Shetty, A., Hussain, T., & Gao, G. (2023). Decentralized connectivity maintenance for multi-robot systems under motion and sensing uncertainties. *NAVIGATION*, 70(1). <https://doi.org/10.33012/navi.552>