Multi-Point Rendezvous in Multi-Robot Systems

Ramviyas Parasuraman[↑], Jonghoek Kim[↑], Shaocheng Luo, Byung-Cheol Min[*]

***Abstract*—Multi-robot rendezvous control and coordination strategies have garnered significant interest in recent years because of their potential applications in decentralized tasks. In this paper, we introduce a coordinate-free rendezvous control strategy to enable multiple robots to gather at different locations (dynamic leader robots) by tracking their hierarchy in a connected interaction graph. A key novelty in this strategy is the gathering of robots in different groups rather than at a single consensus point, motivated by autonomous multi-point recharging and flocking control problems. We show that the proposed rendezvous strategy guarantees convergence and maintains connectivity while accounting for practical considerations such as robots with limited speeds and an obstacle-rich environment. The algorithm is distributed and handles minor faults such as a broken immobile robot and a sudden link failure. In addition, we propose an approach that determines the locations of rendezvous points based on the connected interaction topology and indirectly optimizes the total energy consumption for rendezvous in all robots. Through extensive experiments with the Robotarium multi-robot testbed, we verified and demonstrated the effectiveness of our approach and its properties.**

***Index Terms*—Networked Robots, Rendezvous Control, Multi-Robot Coordination, Hierarchical Consensus.**

I. Introduction

NETWORKED multi-robot systems can be a potential aid in applications such as search and rescue, autonomous exploration, sensing and communication infrastructure, etc. Coordinating a group of robots involves repetitive tasks of rendezvous, formation control, and flocking of the distributed robots. Here, we focus on the rendezvous problem, in which the distributed robots need to gather at a common location either based on consensus or based on immediate goals [1].

Significant progress has been made in recent works addressing the rendezvous problem, namely by introducing distributed control laws for realizing a consensus to gather robots at a common location and by enabling fault tolerance in such controllers [2], [3], [4]. Most of the studies rely on assumption that the initial interaction topology is densely (or completely) connected or that a global coordinate frame of reference (e.g., a global localization system such as GPS) is available. However, such strong assumptions are difficult to satisfy in challenging GPS-denied large environments and in resource-limited distributed robots.

Encouraged by the reconfigurable coordination mechanism in [5], we show a motivating example scenario in Figure 1,

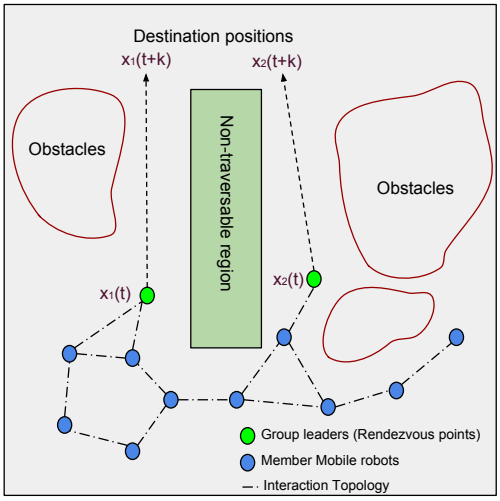


Fig. 1: An illustration of the motivation behind gathering robots in two groups with a leader robot (in green) in each group which has the advanced sensors and computing abilities needed to safely traverse through a cluttered environment. The final goal at time $(t+k)$ is to gather all robots in the upper part of this map, while initially (at time t) all robots share a sparsely-connected interaction topology limited by their sensing range.

in which we assume only a few robots have advanced capabilities (namely to map and analyze the environment with 3D LIDARs) and/or are different from other robots in terms of coverage and visibility (e.g. Unmanned Aerial Vehicles). In such a scenario, it would be appropriate that these advanced robots lead the other member robots to their next target positions while avoiding obstacles in a cluttered environment. To realize this *herding* behavior (inspired from [6]), all the robots need to rendezvous at one of the group-leading robots. Note, the interaction graph (topology) between robots is sparsely connected, reflecting a realistic multi-robot setting.

Furthermore, we take inspirations from the autonomous recharging [7] and coverage planning applications [8], [9] where gathering robots in multiple groups would be of significant interest. Therefore, we address the problem of *distributed multi-point rendezvous without global localization*.

In this paper, we provide a solution to the multi-point rendezvous problem and contribute in the following ways:

- First, we propose a distributed and coordinate-free rendezvous control algorithm based on the hierarchical tracking of a connected graph. We show the theoretical guarantees of the algorithm such as convergence, and link maintenance. The proposed approach is adaptive to changes in network topology and is capable of handling network and mobility faults.

[↑] *R. Parasuraman and J. Kim contributed equally to this work (co-first authors).*

R. Parasuraman, S. Luo, and B.-C. Min are with SMART Lab, Department of Computer and Information Technology, Purdue University, West Lafayette, IN 47907, USA.

J. Kim is with Hongik University, Sejong, South Korea.

[*] Corresponding author email: minb@purdue.edu.

- Secondly, we extend the above algorithm to perform rendezvous at multiple points (leader robots). We also propose to choose the leaders in each group by optimizing the distance traveled to realize the rendezvous task.
- Finally, we demonstrate the proposed algorithm in terms of scalability, efficiency, robustness to failures, and herding along with its dynamic nature through extensive simulation experiments in the Robotarium multi-robot testbed [10]. To validate the performance of our hierarchical tracking algorithm, we compare it against two algorithms available in the literature: the standard consensus in [11] and the circumcenter-based consensus in [12].

The paper is organized as follows. In Sec. II, we analyze the literature and show how we depart from others. Sec. III provides some background on graph theory, assumptions, and definitions. Specifically, the multi-point rendezvous problem is formulated in Sec. III-C. Then in Sec. IV, we introduce the proposed algorithm to realize a multi-point rendezvous of robots with limited speeds. We present the experiments and results in Sec. V and conclude the paper in Sec. VI.

II. Related Work

Distributed rendezvous control laws have been well-studied in the literature [4], [13].Ando et al. [12] proposed a control law to drive a robot towards the circumcenter of the relative positions of all neighboring robots, thereby achieving rendezvous in a distributed manner. In [14], a decentralized control is proposed that preserves global connectivity by identifying and limiting control to critical robots that risk being disconnected from the global interaction graph. The authors of [15] presented a distributed method for navigating a team of robots in formation through static and dynamic obstacles by integrating the local and global motion planners.

Rendezvous control laws can be classified depending on the sensing modality used: 1) coordinate-based (using relative positions) [4], [12]; or 2) bearings-only or range-only (for example using vision sensors) [3], [16]. Although the above methods are able to perform rendezvous efficiently, they have high sensing costs because the robots need to keep track of the coordinates or bearings of all their neighbors in every iteration. Thus, in this paper, we aim to minimize the sensing load of each robot by tracking only one of the neighboring robots (designated as its parent robot) using a hierarchical graph-tracking algorithm that is compatible with both coordinate- and bearing-based local movement controllers.

Fault-tolerant rendezvous control methods are also proposed in the literature to enable multiple robots rendezvous even in the presence of some unidentified faulty robots [2], [17]. Note that [2] required a densely connected initial graph and [17] assumed that the probabilities associated to robot failures are available to realize stochastic optimal control strategies tolerant to faults in the system. However, we do not make such assumptions and we find that there is a need for an efficient rendezvous algorithm that can handle both dense and sparse graphs, which is addressed by our proposed hierarchical tracking approach. In practice, the network topology changes in the event of faulty robots and the proposed algorithm is capable of handling such changes in the network graph.

Inspired by the graph-based approaches in applications such as multi-robot recharging [7], Leader-Follower tactics [18], and herding in groups [6], [19], [20], we propose a unique strategy to achieve multi-point rendezvous by optimally selecting leader robots (to reduce the combined traversal distance of each robot while obeying certain bounds) and assigning other mobile robots to one of the leader robots as their rendezvous points. In addition, a leader robot can freely maneuver to perform its own task as long as it is able to maintain connectivity with its group members. In this way, our control algorithm is flexible and suitable for realistic scenarios where a robot has multiple tasks to perform. As far as we know, we are the first to introduce multiple rendezvous points and intentionally rendezvousing in different groups.

Additionally, literature studies [21] suggested that locomotion is the main contributor to energy consumption in larger robots (the longer a robot travels, the higher the energy consumed). In [22], [23], rendezvous control laws included optimization of motion energy, whereas in our work we minimize the distance traveled by all robots (similar to [24]). Thus, the proposed optimization (grouping) strategy indirectly optimizes energy consumed during the rendezvous task.

Multi-group rendezvous is first studied in [25], where a theoretically sound approach based on Glowworm Swarm Optimization (GSO) is proposed to localize and rendezvous at multiple peaks of a multi-modal source profile. The robots split into multiple groups by following the neighbors with the highest *luciferin* value of the sensed objective function at spatially distributed locations of the robots. However, it fundamentally differs from our work in terms of a sensing strategy and the goal of rendezvous. In [25], the rendezvous happens because all the robots in the group aim to reach the position of the local optimum of a correlated objective function of a physical process in the field (e.g., temperature, radiation, etc.), whereas in our work, the robots split and rendezvous at the dynamic leader robots following the network graph with distance (energy) optimization and herding goals without depending on sensing/measurement of a global physical process.

We depart from existing related work in two important ways. First, we introduce a novel strategy for rendezvous in multiple groups and with dynamic rendezvous points, whereas works in the literature focused on either a static consensus point or rendezvous at multiple pre-determined locations. Second, distributed controllers typically use a network graph without a leader robot for achieving consensus (rendezvous) among multiple robots. In such scenarios, meeting points are decided by the algorithm and the dynamics and spatial distribution of all robots in the network. However, in our work, we employ a hierarchical tracking algorithm using a rooted tree (leader-based rendezvous). Here, meeting points are decided by the dynamics of the leader robots and are not affected by how other non-leader agents are spread out in the space.

Moreover, the presence of dynamic moving leader robots enables practical applications such as herding and multi-point recharging, which is a key novelty of the proposed solution. The advantages of the proposed solution in fault-tolerant rendezvous and multi-point herding applications are demonstrated in Sec. V-G and Sec. V-F, respectively.

III. Background

This section reviews conventions and definitions used in graph theory, introduces the assumptions and definitions used, and formalizes the problem statement.

A. Graph Theory

We provide some general notations used in graph theory [26]. Let $A = (V, E, W)$ denote a undirected weighted graph with vertex set V and edge set E. Each edge, say $e \in E$ is weighted by a non-negative value and is defined by the matrix $W : [w_{ij}]$. Two vertices are *connected* if there is a path between the two vertices, $\{(i,j) \in E \,|\, j \in N_i\}$, where N_i represents the neighbors set for robot i. A *subgraph* of A induced by a set of vertices $S \subseteq V$ is the graph (S, E_S, W_S), where $E_S = \{(i,j) \in E : i,j \in S\}$. We assume that the graph A is symmetric ($w_{ij} = w_{ji}$), there is no self-loop or repetitions of edges ($w_{ii} = 0$), and A is bidirectional ($i \in N_j \leftrightarrow j \in N_i$).

Tree: Assuming the graph is simple, non-cyclic, we define a *rooted tree* $T = (V_T, E_T, W_T)$ where one of the vertices is designated as the *root*, and every vertex in the tree has a hierarchical structure and establishes a parent-child relationship with its neighboring vertices. We use the following terms to define elements of a tree: $p(u)$ denotes the parent robot of u, $c(u)$ denotes the children set of u, and $R(u)$ denotes the root node of the tree containing the node u. A *leaf* in a tree is of the lowest hierarchy (it has no child node $c(u_{leaf}) = \emptyset$). A *shortest-path tree* rooted at a vertex $v \in V$ in the graph A is a minimum spanning tree T of A.

B. Assumptions and Definitions

We consider a set of N mobile robots in a 2D workspace with their positions denoted as $\mathbf{q}_i = [x_i, y_i]^T \in \mathbb{R}^2$. We use the Single Integrator model to describe their dynamics[1].

$$\dot{\mathbf{q}}_i(t) = \mathbf{v}_i(t),\ i \in V = [1, ..., N] \tag{1}$$

where $\mathbf{v}_i(t) \in \mathbb{R}^2$ denotes the velocity control input for robot i at time instant t. We define the following definitions and assumptions that will be used in this work.

First, we presume that there is no global localization or coordinate reference system available, and the robots can only use their local sensors to detect and identify neighboring robots. Each robot is equipped with local onboard sensors to estimate the relative positions or bearings of the neighboring robots. We presume that the maximum speed of any robot u is bounded by S_m ($\|\dot{\mathbf{q}}_u\| \leq S_m$).

Let us define S_R as the maximum sensing range of the local sensor, thus resulting in a S_R-disk proximity graph [27]. Let us define SSR_u as the Safe Sensing Range of a robot u beyond which the robot movement may greatly affect the sensing capabilities. $SSR_u = kS_R, k \leq 1$. This limit is used to let the robots freely maneuver as long as the neighbors are within its SSR.

[1]Without losing generality, the problem can be extended to Double Integrator and higher order dynamics. Also, the problem can be adapted to higher dimensions for use in aerial and underwater vehicles.

We say that a robot *detects* another robot (using range or bearing sensors such as RADAR, LIDAR, SONAR, etc.) when the relative distance between the two robots is within S_R. We say that a robot *meets* or *merges* into another robot when the distance between the two robots is less than $\epsilon \ll S_R$, a small positive threshold accounting for physical proximity and kinematic constraints. We say that two robots are *neighbors* if and only if two robots detect each other as we assume a bidirectional graph. Generally, range or bearing sensors require line of sight with the target objects for accurate measurements.Therefore, we assume that there is a collision-free path (i.e., no obstacles blocking the line of sight) in every edge. Nevertheless, this constraint can be overcome by integrating an obstacle avoidance algorithm at the local controller of the robot.

Let $A(k) = (V_{A(k)}, E_{A(k)}, W_{A(k)})$ denote the connectivity (proximity) graph at time step k. Each vertex in $A(k)$ is associated to each robot, and each edge (i,j) in $A(k)$ implies that two robots associated to i and j are neighbors. Each edge in $A(k)$ is weighted as follows. The weight of each edge, say $(i,j) \in E_{A(k)}$, is directly proportional to the distance between the two robots connected by the edge, $w_{ij}(k) \propto \|\mathbf{q}_i(k) - \mathbf{q}_j(k)\|$, where $\mathbf{q}_i$ and $\mathbf{q}_j$ are the positions of the robots i and j respectively.

A graph is said to be *connected* if there is a *path* (sequence of connected edges with finite total weight) between every pair of distinct vertices. A collection of graphs is *jointly connected* if the union of the set of vertices and edges across all the graphs remains connected. We assume that the initial graph $A(0)$ is a connected graph.

C. Problem Statement

We consider the problem where the robots are needed to assemble (rendezvous) into M groups, where the number of rendezvous points (one for each group) is fixed and predetermined. Let us assign M robots as the leader (root) robots of each group. The subset representing the leader robots is denoted as $D \subseteq V$. Let $D_m \in D$ denote the leader robot of the m^{th} group ($m \in [1, 2, ..., M]$). Let $U = V - D$ denote the subset of robots that are not leader robots. Each robot in U is assigned one leader robot from D. We use the notation u_i to represent an arbitrary robot i. Let $R(u_i) \in D$ denote the leader robot assigned to a robot $u_i \in U$.

Objective: The objective of the rendezvous algorithm is to assemble all robots in U to their assigned leader robots in D, i.e., when $\lim_{t\to\infty} \|\mathbf{q}_{u_i} - \mathbf{q}_{R(u_i)}\| = 0$, $\forall u_i \in U$.

IV. Multi-point Rendezvous Strategy

First, a connectivity graph $A(k)$ is built in a distributed manner with each robot sharing information with its neighbor robots (Sec. IV-A). Then, assuming that the number of groups (M) is fixed, we assign every robot (in U) a leader robot (from D) and decompose $A(k)$ into M shortest-path trees using the method proposed in Sec. IV-B. The rendezvous control algorithm described in Sec. IV-C is applied to each tree separately (and simultaneously). Finally, we also propose a method to optionally choose the best leader robots from a pool of leader robot candidates (Sec. IV-D).

Algorithm 1: Distributed Algorithm for Building a Global Connectivity Graph A

for every robot $u_i \in V$ **do**
  Init $L_{i,-1} \leftarrow \emptyset$ and $L_{i,0} \leftarrow \mathtt{N}_i$ (*local neighbor lists*);
  $c = 0$;
  repeat
    Step 1. Get the unique changes in the local neighbor list L_i as $\bar{L}_{i,c} = L_{i,c} - L_{i,c-1}$;
    Step 2. Send $\bar{L}_{i,c}$ to all $u_j \in \mathtt{N}_i$;
    Step 3. Receive $\bar{L}_{j,c}$ from all $u_j \in \mathtt{N}_i$;
    Step 4. $L_{i,c+1} = L_{i,c} + \bar{L}_{j,c}$ for all $u_j \in \mathtt{N}_i$;
    $c = c + 1$;
  until L_i converges (to global list L);
  Build the global graph A from the converged list L_i;

A. Building the Connectivity Graph in a Distributed Manner

To initiate our rendezvous strategy in a distributed fashion, every robot uses its local sensors to sense neighboring robots and creates a local neighbor set (list) L_i. Each robot then shares its list with neighbors and constructs, using a distributed algorithm (Algorithm 1), global connectivity graph A from the accumulated list L, which contains nodes and edges for all robots in the network. Algorithm 1 is inspired by the distributed consensus algorithm in [15].

To illustrate Algorithm 1, let us consider an undirected cyclic graph with three robots a, b, and c. L of this graph is the accumulation of the three unordered lists by individual robots: $L_a = \{b, c, w_{ab}, w_{ac}\}$, $L_b = \{a, c, w_{ba}, w_{bc}\}$, and $L_c = \{a, b, w_{ca}, w_{cb}\}$. The weights w_{ij} for each edge can be obtained from robot's range sensors, if available. Otherwise, equal weights are presumed.

In Algorithm 1, $\mathtt{N}_i$ denotes the *neighbor set* of a robot i. The convergence to a global list L (hence the graph A) at every robot is obtained in a maximum of $\mathbb{D}(A)$ iterations (i.e., $c \leq \mathbb{D}(A)$), where $\mathbb{D}(A)$ is the diameter of a connected graph A (the number of nodes along the shortest path from one node to its farthest node in the graph A). Thus, $L_{i,\mathbb{D}(A)} = L \; \forall i \in V$, which can be proved by following an approach similar to the Proposition 1 of [15]. Therefore, we omit the proof here.

The worst-case computation cost of the Algorithm 1 depends on the graph diameter $\mathbb{D}(A)$ and the maximum number of neighbors for each robot. From [28], we deduce that the graph diameter is upper bounded by $\frac{(N-1)}{K}$, where K indicates the K-connectedness of the graph. Therefore, the algorithm is scalable because the number of robots in the system is significantly lower than the number of edges and can efficiently be managed in practical applications. Also, compared to a sparse graph (e.g., a path graph), dense graph (e.g., a complete graph) has a lower diameter but a higher number of neighbors. Thus, both the communication and computation workloads are balanced in terms of density of the graph.

Note that the Algorithm 1 is called only in cases of changes in the network topology and the graph $A(k)$ is dynamically updated according to the changes in the local sensor information (for instance, a robot detected a new neighbor).

B. Creating Multiple Groups

Let us presume that the set of leader robots D is selected (or pre-known) depending on their capabilities such as advanced sensing systems, recharging stations, etc. Alternatively, the leader robots can be chosen from a given graph $A(k)$ using the proposed method in Sec. IV-D, which optimizes the maximum traversal distance by any robots in the rendezvous process.

Since our objective is to gather all robots to their leader robots with the least amount of time and energy, we assign each robot $u_i \in U$ a leader robot $R(u_i) \in D$ by optimizing the following objective function:

$$R(u_i) = \arg\min_{D_m \in D} d_{A(k)}(u_i, D_m). \tag{2}$$

Here, the function $d_{A(k)}(u_i, u_j)$ represents the shortest path distance (sum of weights for all edges in the shortest path connecting them) between u_i and u_j in the graph $A(k)$ using Dijkstra's algorithm [29].

Using the initial graph $A(0)$, the robots are grouped together to form M sub-graphs $A_m = (V_m, E_m, W_m)$, where each sub-graph contains nodes, which satisfy the condition:

$$u_i \in V_m \iff R(u_i) = D_m, \forall u_i \in V. \tag{3}$$

Note the sub-graphs can be dynamically reconstructed as the graph evolves with $A(k)$, but we consider a special case in which the graph is split only initially to reduce computational load. On the other hand, each sub-graph is dynamically updated based on local changes in the sub-graph ($A_m = A_m(k)$).

Using the sub-graphs A_m, we construct M shortest-path trees $T_m = (V_m, E_{T_m}, W_{T_m})$ with root nodes as D_m by optimizing the distance cost (using weights W_{T_m}) from each node to the root node. Each node u_i in a tree is assigned a parent $p(u_i)$ and children (if any) $c(u_i)$. Nodes without any children are the leaf nodes (see Sec. III-A). Converting the sub-graphs into tree structures significantly reduces the number of edges from (potentially) $|E_{T_m}| = O(|V_m|^2)$ to $|E_{T_m}| = |V_m| - 1$ resulting in higher computational efficiency.

C. Hierarchical Rendezvous Algorithm

We propose an efficient hierarchical rendezvous algorithm that enables every robot in the tree T_m to rendezvous at its root D_m. Algorithm 2 shows the procedure of the hierarchical rendezvous that runs iteratively until rendezvous is completed.

The proposed algorithm works as follows. The control applied to each robot is to move along the shortest path to the root robot in T_m. All non-root robots start to move towards their immediate parents in the tree hierarchy towards their root node. The robot that has at least one child (non-leaf nodes) can move towards its immediate parent only if all its children are within a safe sensing range SSR_u. Else, it stays and waits for its children until they all reach its SSR. After reaching their immediate parents, the robots will change their parent to the parent of their parents $p(u) \leftarrow p(p(u))$ by updating the edges of the tree (by removing the link $(u, p(u))$ and adding the link $(u, p(p(u)))$ in tree T_m). This also ensures that the list of children in the parent nodes is updated accordingly $(c(p(u)) = c(p(u)) - u, c(p(p(u))) = c(p(p(u))) + u)$.

Furthermore, as per Algorithm 2, a parent robot u can move only when all its immediate children are within a SSR_u distance from u. Suppose that a child of u, say v, is initially farther than SSR_u from u and arrives at u later than other

Algorithm 2: Hierarchical Rendezvous Control Algorithm for every tree T_m

repeat
- **if** there is a change in connection/mobility status **then**
  - Update A_m and the shortest-path tree T_m;
- **for** $u \in V_m \leftarrow$ every robot in T_m **do**
  - $\dot{\mathbf{q}}_u = 0$ (default state);
  - **if** $u = D_m$ (root node - leader robot) **then**
    - **if** $\|\mathbf{q}_u - \mathbf{q}_j\| \leq SSR_u, \forall j \in c(u)$ **then**
      - $\|\dot{\mathbf{q}}_u\| \geq 0$ ($\dot{\mathbf{q}}_u$ is not restricted. e.g., Herding);
  - **if** $u \neq D_m$ AND a leaf node **then**
    - $$\dot{\mathbf{q}}_u = (\mathbf{q}_{p(u)} - \mathbf{q}_u) \quad (4)$$
    - **if** $\|\mathbf{q}_u - \mathbf{q}_{p(u)}\| < \epsilon$ **then**
      - Update the parent of u: $p(u) \leftarrow p(p(u))$;
  - **if** $u \neq D_m$ AND a parent node **then**
    - **if** $\|\mathbf{q}_u - \mathbf{q}_j\| \leq SSR_u, \forall j \in c(u)$ **then**
      - Apply controller in Eq. (4);
      - **if** $\|\mathbf{q}_u - \mathbf{q}_{p(u)}\| < \epsilon$ **then**
        - Update the parent of u: $p(u) \leftarrow p(p(u))$;
  - Bounding $\dot{\mathbf{q}}_u = \min(S_m, \|\dot{\mathbf{q}}_u\|) \frac{\dot{\mathbf{q}}_u}{\|\dot{\mathbf{q}}_u\|}$;
- **until** $\|\mathbf{q}_u - \mathbf{q}_{D_m}\| \leq \epsilon \; \forall u \in V_m$;

children of u. In this case, only the parent robot u waits for v, while the other children of u keep moving toward u and subsequently will update their parent from u to $p(u)$.

Below, we discuss the theoretical guarantees and the key properties of the proposed algorithm. We first prove that the robots in each tree T_m converge to (rendezvous at) their leader robots D_m (effectively guaranteeing the solution to the problem described in Sec. III-C).

Theorem 1. *For every sub-graph A_m (tree T_m), all the robots in V_m converge to D_m using Algorithm 2, provided that the initial graph $A(0)$ is a connected graph and over a finite time interval between two successive iterations $[k, k+1]$, the graph $A_m(k, k+1) = \cup_{\tau \in \{k,k+1\}}(V_{A_m(\tau)}, E_{A_m(\tau)})$ is jointly connected.*

Proof. We prove convergence and global stability using Lyapunov analysis for discrete control systems. Let us consider the Lyapunov-like candidate function

$$\mathbb{V} = \sum_{i \in V_m} \|\mathbf{q}_{p(i)} - \mathbf{q}_i\|, \quad (5)$$

where $p(i)$ represents the parent node of node i in the tree T_m. For the control in (4), the equilibrium point $\mathbf{q}^{eq} = \mathbf{q}_{D_m}$ is the position of the root node, where $\dot{\mathbf{q}}_u = 0 \; \forall u \in U$. It is clear that $\mathbb{V} = 0$ if and only if every robot in V_m merges into D_m, which is the top-most parent in the tree T_m. Therefore, $\mathbb{V} > 0 \; \forall \mathbf{q}_u \neq \mathbf{q}^{eq}$, and $\mathbb{V} = 0 \iff \mathbf{q}_u = \mathbf{q}^{eq} \forall u \in U$, implying $\mathbb{V}$ is positive definite. The time derivative of $\mathbb{V}$ is,

$$\dot{\mathbb{V}} = \sum_{i \in V_m} \frac{(\mathbf{q}_{p(i)} - \mathbf{q}_i)^T}{\|\mathbf{q}_{p(i)} - \mathbf{q}_i\|} (\dot{\mathbf{q}}_{p(i)} - \dot{\mathbf{q}}_i). \quad (6)$$

As per Algorithm 2, the parent node waits until all the children are within its SSR_u. i.e., $\dot{\mathbf{q}}_{p(i)} = 0$. This is also the case when the nodes are in the close vicinity of the equilibrium, where the root node is the parent for all the nodes, and for the rendezvous case, $\dot{\mathbf{q}}_{D_m} = 0$. Therefore,

$$\dot{\mathbb{V}} = -\sum_{i \in V_m} \frac{(\mathbf{q}_{p(i)} - \mathbf{q}_i)^T}{\|\mathbf{q}_{p(i)} - \mathbf{q}_i\|} \dot{\mathbf{q}}_i, \quad (7)$$

$$\dot{\mathbb{V}} = -\frac{1}{S_m} \dot{\mathbf{q}}_i^2 \leq 0. \quad (8)$$

Hence, we show that the Lyapunov candidate function is negative semidefinite, proving stability of the nonlinear system in the sense of Lyapunov. Note, for the control in (4), $\dot{\mathbb{V}} = 0$ only when all the nodes are within ϵ distance from the root node. When the parent and child are within SSR_u, then both parent and child apply the same control of moving toward their parents (but not in opposite direction). Therefore, $\dot{\mathbb{V}} \leq 0$ still holds. Finally, the system converges to the invariance set where $\dot{\mathbb{V}} = 0$, according to LaSalle invariance principle [30].

Although the graph A_m is time-varying (switching topology), changes between iterations do not introduce discontinuities because of the condition that the graph is *jointly connected* between any two finite time intervals (periodic) [31], [19]. Maintenance of connectivity is addressed in Theorem 2.

Also, in our assumption (Sec. III-B), two robots meet (or merge into) each other when they are very close $\|\mathbf{q}_i - \mathbf{q}_j\| \leq \epsilon$. As soon as a robot meet its parent, it re-assigns its parent to the higher-order parent in the tree. This can induce a jump in $\mathbb{V}$ (a discontinuous decrease in $\mathbb{V}$). However, the number of such jumps is limited by the total number of robots in the system, which is finite. Therefore, these discontinuities do not affect the convergence of the system. This concludes the proof. □

Next, we show that the proposed algorithm guarantees that each robot maintains connectivity with their parents and their children during the rendezvous process.

Theorem 2. *(Connectivity maintenance) Using Algorithm 2, the tree T_m remains connected during the rendezvous process.*

Proof. To prove this, we show that every robot is sensed by at least one robot during a maneuver. Let $P(u, D_m)$ denote the path taken by a robot u to reach its root D_m. Let pu denote any of the parent robots (including D_m) that u pursues in its hierarchy following the path $P(u, D_m)$. As per the tree, pu also shares the subset of the path of u, and pu begins moving towards its parent only if u is within the SSR_u, i.e., pu moves along the path $P(pu, D_m) = P(u, D_m) - P(u, pu)$. Maintaining u within the SSR_u of pu ensures that u is always sensed by pu (because $SSR \leq S_R$), therefore u and pu maintain their connectivity even when both are moving. This also includes the case where $pu = D_m$ is freely maneuvering (to execute herding behavior for example).

In cases where pu cannot move (say, because one of its child is not yet within its SSR), then u passes pu and continues to pursue the higher-level parent ppu (if $pu \neq D_m$). In this case, u will sense ppu because when u meets pu (when $d_{T_m}(u, pu) \leq \epsilon$), u and ppu can both be sensed by each other ($d_{T_m}(u, ppu) \leq S_R$). Since every robot is sensed by at least one robot, mutual protection can be provided for every robot. Thus, u always maintains connectivity with any of the robots in its path, hence A_m (and tree T_m) is always connected. □

Theorem 3. *The anticipated traversal distance of any robot in T_m following Algorithm 2 until it rendezvous is bounded by the shortest path distance $d_{T_m}(u, D_m)$.*

Proof. Trivial. The tree T_m is the shortest path tree generated from the sub-graph A_m. Thus, the path followed by any robot u during the rendezvous process has a length (total distance) of $d_{T_m}(u, D_m)$. □

As per the algorithm, each robot senses and pursues it's parent node. This can be realized by using relative coordinates [11] or bearings [3] of the parent node.

Remark 1. *This implies that the proposed algorithm is coordinate-free and that the robots do not require a global coordinate reference or a global localization system.*

Note, in our proposed method, obstacle detection and avoidance are handled using the local motion controllers of each robot (e.g., [32], [33]) during the phase when the robot moves toward its parent as per Algorithm 2. For sake of brevity, we do not analyze the impact of obstacle avoidance on the behavior of the proposed multi-point rendezvous method. Convergence will not be affected as long as the obstacles do not completely disrupt the sensing capabilities of the robot.

Algorithm 2 is designed to handle the case where a communication or the interaction topology is dynamically updated (see Sec. IV-A). It is also designed to handle cases of mobility or communication faults if the fault is detected and identified by at least one robot in the system[2] (using the method in [34] for example). In such cases, we re-generate the tree T_m using the updated graph $A(k)$, and the algorithm works on the updated tree structure.

Remark 2. *This implies that the proposed algorithm can work even in cases of intermittent connectivity and is tolerant to communication or mobility related faults.*

The root node of every tree is the rendezvous point of the sub-group. The root robots D_m in every tree can freely maneuver as long as all of their immediate children are within the SSR_{D_m} (which is a requirement in Theorem 2 to safely maintain connectivity with their children).

Remark 3. *This implies that the proposed algorithm enables herding behavior guided by the moving leader (root) robots.*

Herding here means that the leader robots guide (herd) their group member robots to the final destination positions [6] either during or after the rendezvous process. Note herding is different from *flocking* behavior [35] in which all the robots tend to travel in the same direction keeping their formation.

D. Choosing Leader Robots (Rendezvous Points)

In situations where the leader robots are not known or cannot be chosen based on their capabilities[3], we present a simple and efficient strategy to optimally choose M leader robots (rendezvous points) among all the robots in the network.

In multi-robot applications, it is highly desirable to achieve rendezvous in the shortest possible time. This goal can be met if we minimize the (anticipated) maximum distance traveled by any robot during the rendezvous process. This limits the time and energy consumed by any robot in the system. Note, Theorem 3 provides the bound for this maximum distance which impacts the time to rendezvous. Therefore, we search the optimal leader robots set D from the set V in the initial graph $A(0)$ by minimizing the following cost function

$$C(D) = \max_{\forall D_m \in D} \max_{\forall u \in V_m} d_{A(k)}(u, D_m), \tag{9}$$

subjected to the following constraints

$$L_m \leq |V_m| \leq U_m \tag{10}$$

where $|V_m|$ denotes the number of robots in the group in which D_m is the leader after applying the grouping algorithm in Sec. IV-B. L_m and U_m are the bounds for $|V_m|$. Also, considering a distance bound T_D, we have additional constraint as follows.

$$d_{A(0)}(u, R(u)) \leq T_D, \ \forall u \in (V - D). \tag{11}$$

The first constraint balances the sizes of all M groups whereas the second constraint restricts the maximum distance traveled by any robot in the rendezvous task.

Since the cost function to be minimized (Eq. 9) is non-smooth and non-differentiable, standard analytical methods such as Integer Linear Programming (ILP) cannot be applied directly. Also, the number of robots in the pool of potential candidates of leader robots V_{pl} among all the robots is very limited in practice ($|V_{pl}| << N$). Thus, the scalability of the search solution is not an issue.

Therefore, we propose Brute-Force search solution which is simple and efficient to solve the following optimization problem:

$$D^* = \arg\min_{D \in D_{pl}} C(D) \tag{12}$$

where $D_{pl} = \binom{V_{pl}}{M}$ is the set of all candidate solutions for D, and D^* is the optimal set of leader robots which will eventually be used in Algorithm 2.

Using Algorithm 1, each robot builds the global graph A in a distributed manner. This results in every robot sharing an identical graph A. Then, D^* is calculated using the initial graph $A(0)$ as per Algorithm 3. Note that while this optimization can be performed distributively by applying Algorithm 3 locally on every robot, doing so is only needed in the initial stage; therefore, we assume that a central agent performs this particular optimization and shares the resulting D^*.

In Algorithm 3, the initial connectivity graph $A(0)$ is used to generate a table T_b, which is an N by N matrix that presents the all-path shortest-path distances between every robot pair, with each row representing a source robot. For example, the element of T_b at the ith row and jth column presents the shortest-path distance between two robots u_i and u_j. By definition, $T_b(i, i)$ is zero.

[2]Note our assumption that the faults are identifiable is different from the assumption used in the Tverberg partition based method [2] which can work even under the presence of unidentified faults, however at the cost of a limitation that a densely connected initial graph $A(0)$ is required.

[3]An example of such situation is a homogeneous multi-robot system where all the robots have equal capabilities.

Algorithm 3: Calculate D^*

Data: Graph A from Algorithm 1
Generate table T_b using Dijkstra's algorithm on A;
$cost = \inf$;
for $D \in$ List of M leaders chosen from ${}^{M}C_{N}$ selection cases **do**
  if D satisfies the constraints (10) and (11) **then**
    $cost_o \leftarrow C(D)$ from Eq. (9);
    if $cost_o < cost$ **then**
      $cost \leftarrow cost_o$;
      $D^* \leftarrow D$;

We vary the source robot among all robots and derive the shortest path from the selected source to every other robot using Dijkstra's algorithm [36]. The time complexity of Dijkstra's algorithm is $O(N^2)$, so varying over N source robots is $O(N^3)$. Suppose that T_b is built. Then, we arbitrarily choose M source robots out of N robots. There are ${}^{M}C_{N}$ selection cases in total. For each case, the chosen M source robots are the rendezvous robots. In other words, each case represents a situation where M rendezvous robots are selected. We derive the cost function in (9) associated with each case, and search for the case that returns the minimum cost while obeying the constraints in (10) and (11).

The complexity of Algorithm 3 will scale up with the number of edges in the graph $A(k)$. Thus, the proposed solution is effective in cases where $A(k)$ is a sparse graph. Addressing this problem for dense graphs is an avenue for future work.

V. Experimental Evaluation

In this section, we first present the experiment setup, evaluation metrics, and list the experiment scenarios considered in the experiments. Then, we describe each of the scenarios along with a discussion of the achieved results. Note the experiment results of all the scenarios are available in the video[4].

A. Experiment Setup

To validate the proposed method, we conducted extensive simulation experiments in the Robotarium MATLAB platform [10] which is a free multi-robot and swarm robotics experiment testbed recently made available by Georgia Tech [10]. We chose the Robotarium because it provides testbeds for both simulations and hardware experiments with the GRITSbot robots [37] using the same software implementation.

Robotarium has an 2D arena of size $3m \times 3m$, in which up to 15 robots (currently supported) can be used in hardware experiments; however, in the simulations, the number of robots are not restricted. The GRITSbot robot (size $4cm \times 4cm$) exhibits unicycle dynamics, but it can be also controlled via single integrator dynamics. The coordinates of all the robots can be accessed in every iteration during the experiments. The common parameter settings are $S_R = 0.8m$, $S_m = 0.1\,\mathrm{m/s}$, $\epsilon = 0.1m$, $SSR = 0.5S_R$, max number of iterations $k_{max} = 1000$. The Robotarium Robotarium updates the pose at a maximum of 30 Hz.

[4] Youtube link: https://youtu.be/uaiCnw79Sb8

B. Evaluation Metrics

We used the following three metrics to measure the performance of the rendezvous task in our experiments.

1) Lyapunov candidate function (Convergence):

$$L = \sum_{u_i \in U} \|\mathbf{q}_{u_i} - \mathbf{q}_{R(u_i)}\|. \tag{13}$$

2) Total distance traveled by all robots (Distance):

$$\mathtt{D}_{sum} = \sum_{u_i \in U} \sum_{k=1}^{k_{max}} \|\mathbf{q}_{u_i}(k+1) - \mathbf{q}_{u_i}(k)\|. \tag{14}$$

3) Number of iterations ($k_{stop} \leq k_{max}$) to reach the following stop condition:

$$\text{Stop at } k_{stop} \text{ if } \|\mathbf{q}_{u_i} - \mathbf{q}_{R(u_i)}\| \leq \epsilon,\ \forall u_i \in U. \tag{15}$$

These metrics show how fast the algorithm converges to achieve rendezvous, and how long the robots travel before achieving rendezvous.

C. Experiment scenarios

We consider the following five scenarios to verify and demonstrate the effectiveness and the performance of the proposed rendezvous strategy:

- *Scenario 1* - Single-point rendezvous
- *Scenario 2* - Multi-point rendezvous
- *Scenario 3* - Multi-point rendezvous with identifiable faults and/or detectable communication failures
- *Scenario 4* - Herding with multi-point rendezvous
- *Scenario 5* - Optimization of rendezvous points

Using Scenario 1 (Sec. V-D), we verify the effectiveness of the Algorithm 2 in terms of convergence and scalability and analyze how it compares to the state-of-the-art consensus algorithms. We verify in Scenario 2 (Sec. V-E) the multi-point capability of our proposed hierarchical rendezvous algorithm. Then, in Scenario 3 (Sec. V-F) we examine how the proposed rendezvous algorithm performs under mobility faults and intermittent connectivity. This scenario is also used to validate whether the algorithm guarantees convergence of non-faulty robots. Scenario 4 (Sec. V-G) exemplifies an effective herding behavior facilitated by the multi-point rendezvous algorithm. Finally, in Scenario 5 (Sec. V-H) the advantages of choosing optimal leader robots are observed.

D. Scenario 1 - Single-point rendezvous

In this scenario, we verify the convergence property of the Algorithm 2 when $M = 1$ under the cases $N = [10, 20, 30]$. Additionally, this scenario is also used to compare the performance of our hierarchical tracking algorithm against two algorithms available in the literature: the standard coordinates based consensus algorithm in [11] and the circumcenter based consensus algorithm in [12]. The robot initial positions and the respective initial graphs A are shown in Fig. 2.

Since the consensus algorithms are designed to converge at a consensus point determined by the geometric distribution of the robots, we adapted them to rendezvous at a specific

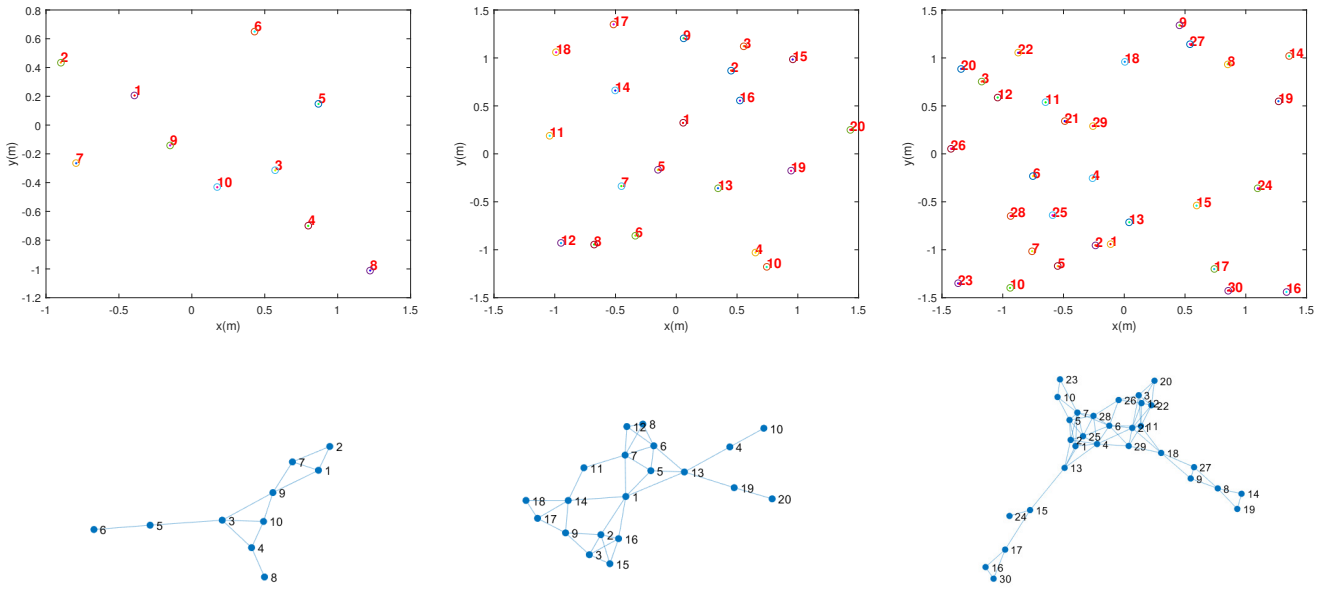


Fig. 2: The top figures show the initial positions of the robots used in the experiments with $N = 10$ (left), $N = 20$ (center), and $N = 30$ (right). The respective initial network/interaction topologies (undirected graphs) are shown in the bottom figures.

root robot node (instead of a consensus point) by assigning a zero velocity to the root robot. However, the consequence of such adaptation is that these algorithms perform slower and the robots trajectories may be significantly changed during the course of the process. To the best of our knowledge, there is no rendezvous control algorithm in the literature that converges to a specific root node, hence this adaptation to the state-of-the-art consensus algorithms is deemed necessary for comparison.

1) Results of Scenario 1: Figure 3 presents the resulting trajectories and the convergence plots (Eq. 13) of the three algorithms compared in Scenario 1 for the different number of robots. The results for the metrics of the sum of distance traveled (Eq. 2) and the number of iterations to reach rendezvous stop condition (Eq. 15) are provided in Table I.

Algorithm	Metric	$N = 10$	$N = 20$	$N = 30$
Ours	$\mathtt{D}_{sum}(m)$	9.31	23.11	39.45
	k_{stop} (#)	549	292	304
CircumCenter [12]	$\mathtt{D}_{sum}(m)$	12.76	30.69	58.7
	k_{stop} (#)	592	484	346
Coordinates [11]	$\mathtt{D}_{sum}(m)$	13.99	32.5	46.9
	k_{stop} (#)	615	654	-

TABLE I: Results of the metrics E and k_{stop} in Scenario 1.

From Table I, we can notice that the proposed strategy outperformed the other algorithms in terms of shorter combined travel distance by all robots to achieve rendezvous ($\mathtt{D}_{sum}$) and lower number of iterations required to converge (k_{stop}).

It can be seen from Fig. 3 that the trajectory of the proposed hierarchical rendezvous algorithm is smoother than the other two algorithms compared here. This can be attributed to the fact that the consensus-based algorithms almost achieved rendezvous at a common point (based on their distribution) as they are originally supposed to, and then continued to rendezvous at the desired leader robot due to the adaptation mentioned before.

In terms of convergence rate, the hierarchical algorithm had the fastest convergence in general and continued to increase when the number of robots (N) is increased. Note, when the number of robots is increased, the number of neighboring nodes for each robot is larger. This resulted in more efficient movements observed by the Robotarium pose updates (faster convergence) when N is larger.

Although the k_{stop} metric showed a significant decrease when N is increased from 10 to 20, it did not show such a decrease when N is increased to 30 from 20. Perhaps, this is because of the current Robotarium design which is expected to scale up to 100 robots in future.

Similar to circumcenter based algorithm [12], our algorithm guarantees connectivity maintenance resulting in the connected graph at the end. On contrary, the coordinates based consensus algorithm [11] does not guarantee connectivity maintenance, and thus resulted in a disconnected graph for the case $N = 30$. In fact, we implemented an improved version of this coordinate-based algorithm from [4] where potential fields (weights) are used to ensure connectivity maintenance. Although the improved algorithm guaranteed connectivity, the convergence rate of [4] was significantly lower than [11] mainly because the control inputs in [4] are weighted based on the distance between the robots which resulted in slower movement when the robots are close to achieving rendezvous. Thus, we chose not to present the results of the consensus algorithm in [4].

E. Scenario 2 - Multi-point rendezvous

The efficiency of the multi-point rendezvous algorithm for $N = 30$ with a different number of rendezvous points ($M =$

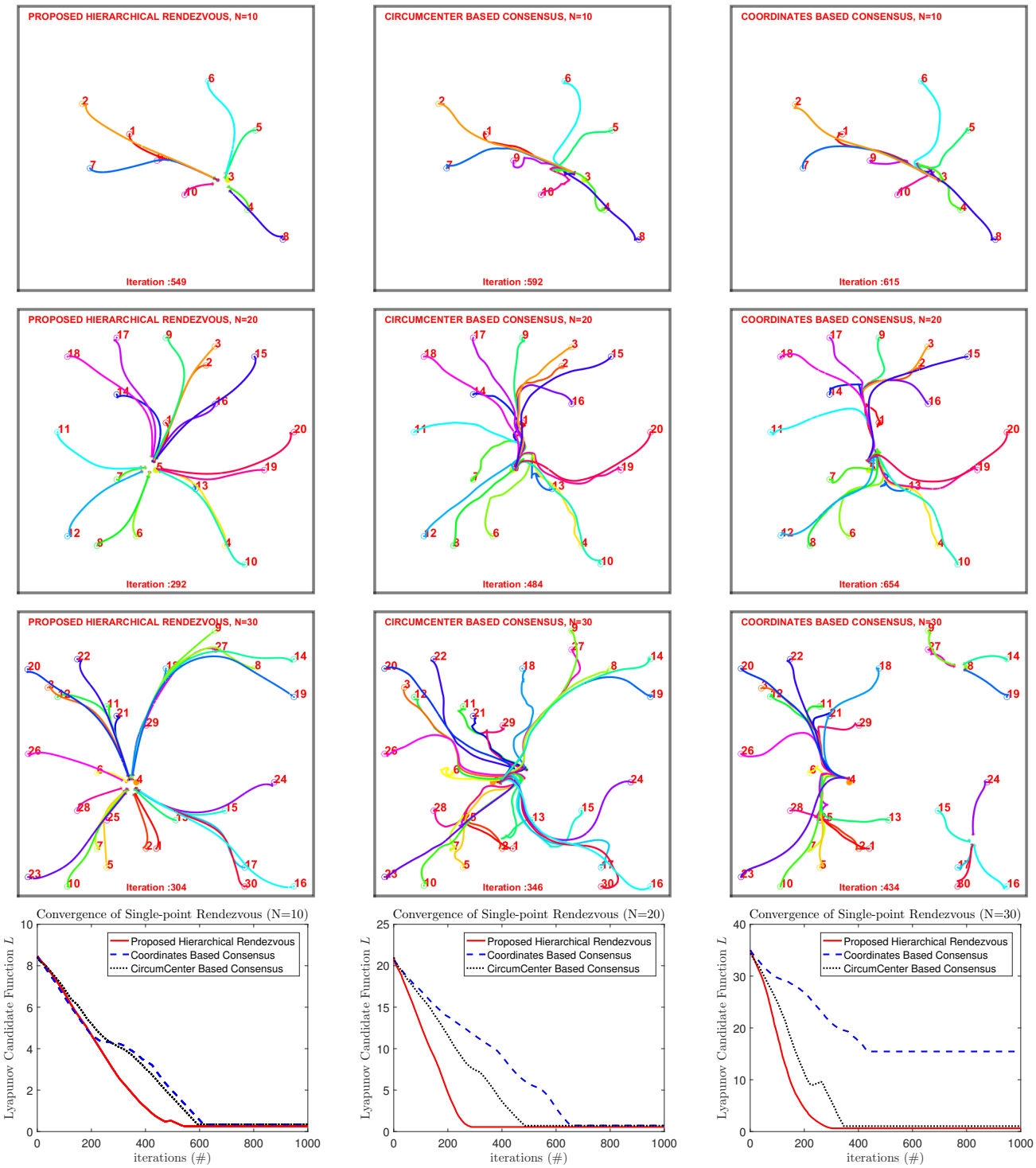


Fig. 3: Results of Scenario 1 - The proposed rendezvous Algorithm 2 (left) is compared with the circumcenter [12] (center) and coordinates [11] (right) based consensus algorithms from the literature under different number of robots. The rendezvous point for the cases $N = 10$, $N = 20$, $N = 30$ are the robots 3, 5 and 4 respectively. The first row shows the final trajectories of the robots in case $N = 10$, the second row shows the final trajectories of the robots when $N = 20$, the third row shows the final trajectories when $N = 30$, and the bottom row compares the rate of convergence (Eq. 13) in all the three cases.

$[2, 4, 6]$) is analyzed here. This scenario also verifies how the algorithm scales in terms of the number of rendezvous points. Note the leader robots (the rendezvous points) are pre-selected by running the optimal rendezvous point selection strategy in Sec. IV-D.

1) Results of Scenario 2: In Fig. 4, we show the results of the converging trajectories of the robots obtained in scenario 2. The trees generated for each leader group is presented at

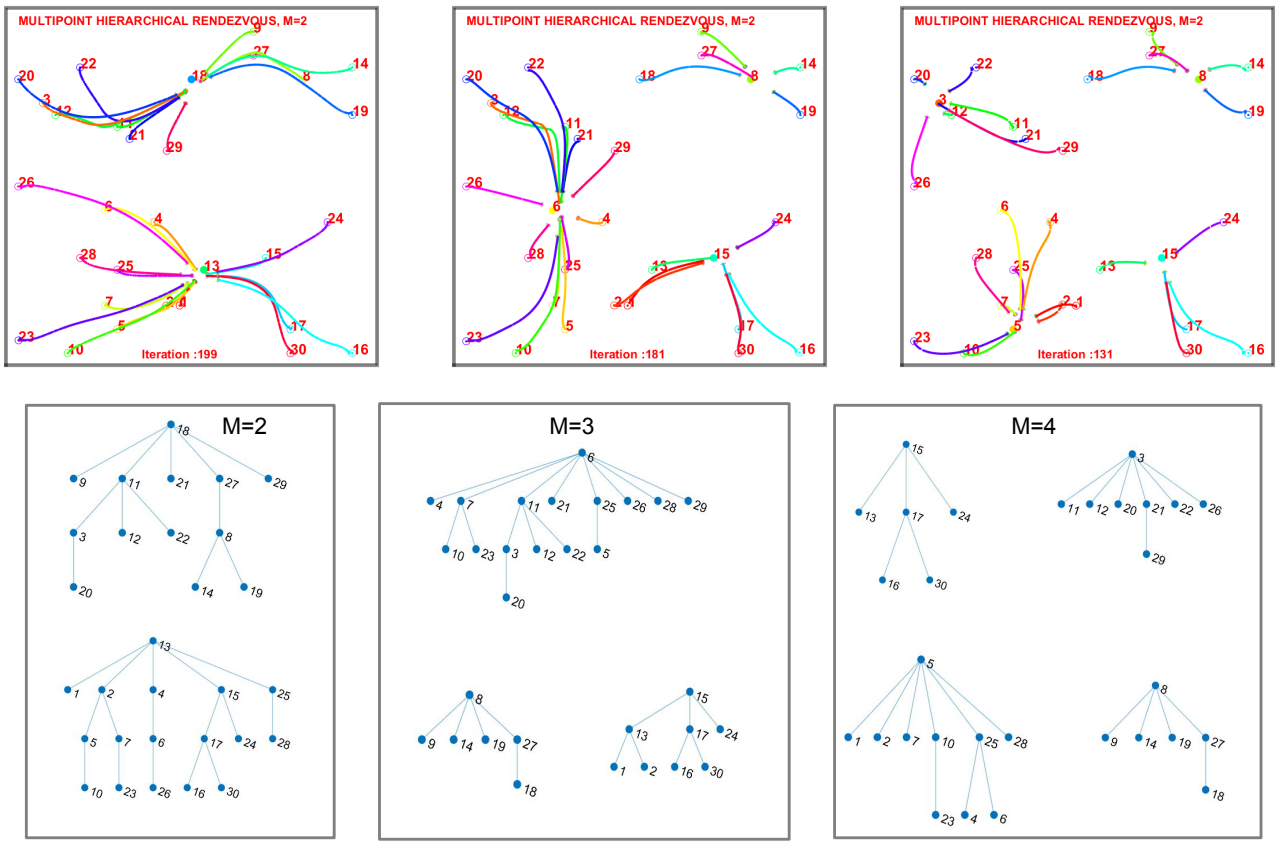


Fig. 4: Results of Scenario 2 - Multi-point rendezvous ($N = 30$) with $M = 2$ (left), $M = 3$ (center), and $M = 4$ (right). The initial trees (created using the approach in Sec. IV-B) rooted at their leader robots of each group are depicted in the bottom row.

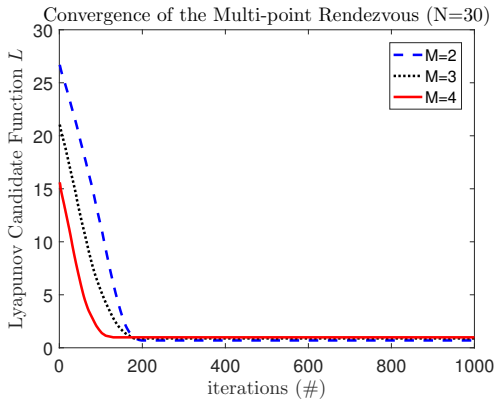


Fig. 5: Multi-point rendezvous convergence in Scenario 2.

the bottom row of the Fig. 4. In the case of $M = 2$, the robots were split into two groups of size 17 and 13 robots. When $M = 3$, the number of robots in each group were 16, 8, and 6. For $M = 4$, the groups had 10, 8, 6, and 6 robots. Although it is possible to experiment with higher M values, it will not present meaningful results in terms of convergence or efficiency. It can be easily seen that the trajectories of the robots were not a straight path to their leader robots because the algorithm routes the non-root robots to track their parents in the hierarchical tree.

Table II shows the metrics $\mathtt{D}_{sum}$ and k_{stop} in scenario 2. The robots traveled shorter due to the fact that the number of robots in each group is reduced when M is increased.

The resulting tree graphs in Fig. 4 show that the grouping algorithm in Sec. IV-B led to evenly distributed subgroups from the initial graph $A(0)$ (see the bottom right graph of Fig. 2). Also, see that within every subgraph, the robots converged with smooth trajectories toward their leader robots (rendezvous nodes).

The convergence of the algorithm under different M values are shown in Fig. 5. It can be seen that the higher the number of leaders (M), the faster the convergence due to the fact that each group runs the algorithm in parallel and the average size of the group is reduced when M is increased.

F. Scenario 3 - Multi-point rendezvous with faults

This scenario is designed to verify the robustness of the proposed rendezvous strategy under identified communication and mobility faults. The number of robots and rendezvous points in this scenario is $N = 30$ and $M = 2$. The

Metric	$M = 2$	$M = 3$	$M = 4$
$\mathtt{D}_{sum}(m)$	29.51	21.81	15.62
k_{stop} (#)	199	182	131

TABLE II: Results for the Algorithm 2 in Scenario 2.

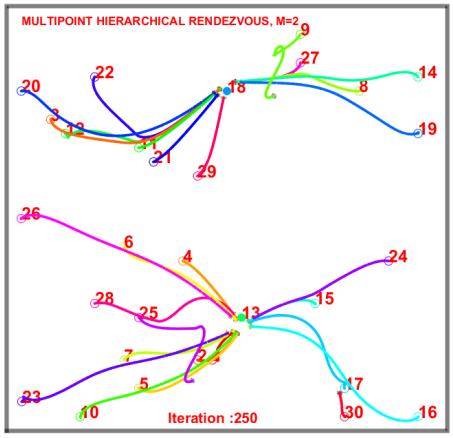


Fig. 6: Robot trajectories in Scenario 3 - Multi-point rendezvous with faulty robots. The robots 9 and 25 are simulated with mobility faults (at $t = 25$ and $t = 50$ respectively) whereas the robots 30 and 11 are simulated with communication faults (at $t = 25$ and $t = 50$ respectively). The robot 11 regained its connection at $t = 100$ (hence, it could achieve rendezvous with its root robot 18).

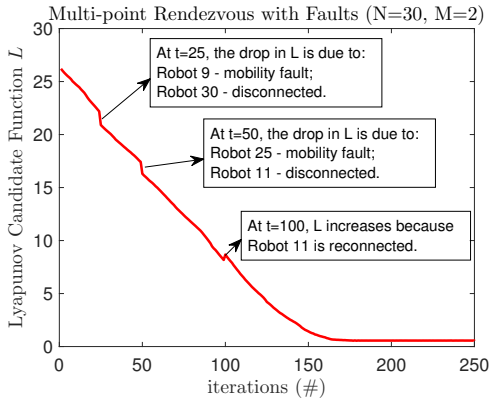


Fig. 7: Convergence plot of Scenario 3. The Lyapunov function L in Eq. 13 is calculated only for non-faulty robots. Note the sudden drop and raise in the L value due to the injected (simulated) faults. The non-faulty robots achieved rendezvous.

corresponding initial graph is shown in Fig. 2 (right) and the corresponding trees are shown in Fig. 4 (left, case $M = 2$).

At arbitrary instances, randomly selected robots (except the root nodes) are injected with mobility faults (by setting their velocities to follow a random sinusoidal trajectory following the strategy in [2]) or with communication faults (by removing the robot in the graph to simulate disconnection for instance).

Specifically, the following faults situations are simulated:

- *At iteration* $t = 25_+$, robots 9 and 30 are respectively injected with a mobility fault and a communication fault (sudden disconnection).
- *At* $t = 50_+$, the robot 25 faces a mobility fault and the robot 11 faces a communication disruption (to simulate an intermittent connectivity).
- *At* $t = 100_+$, the robot 11 regains connectivity with its neighbors (e.g., using a reactive motion planner [38]).

1) Results of Scenario 3: Figures 6 and 7 show the results of the trajectories and convergence plot in Scenario 3. It can be seen from Fig. 6 that the robot 30 was participating in the rendezvous process initially, but as soon as it got disconnected, it did not move. On the other hand, the robot 11 (intermittent connectivity) lost its connection and did not move initially, nevertheless, it completed rendezvous process as soon as it got reconnected. Similarly, the robots 9 and 25 (mobility faults) got removed from the tree as soon as they became faulty and identified as faulty robots. The other non-faulty robots adapted their trajectories accordingly to achieve rendezvous.

These behaviors can also be observed from the convergence of the Lyapunov candidate function (with only non-faulty robots) shown in Fig. 7 where the faults are tolerated by the proposed rendezvous algorithm.

G. Scenario 4 - Herding with multi-point rendezvous

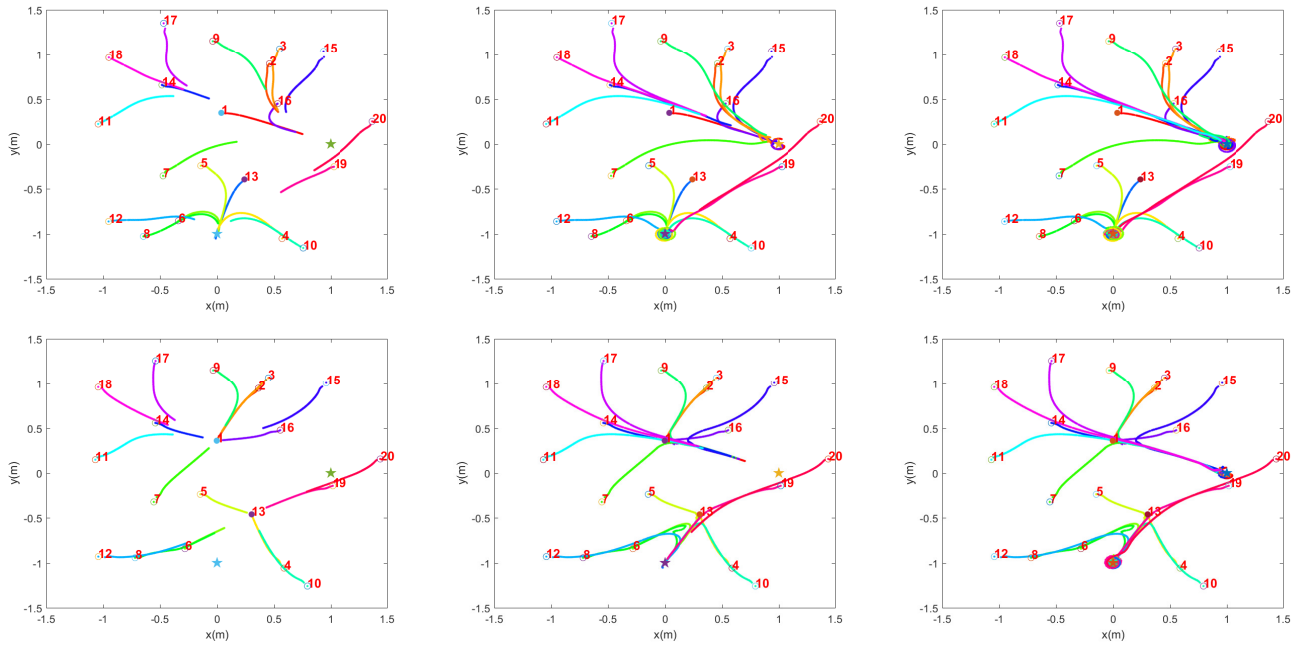
In this scenario, we show how the proposed algorithm enables herding behavior by aiding the movement of rendezvous (leader) robots to their specific goal positions in the arena during or after achieving rendezvous. We simulate this scenario with $N = 20$ and $M = 2$ (see the center graph in Fig. 2). The leader robots are chosen as 1 and 13 which are capable to reach their final destinations $(1, 0)$ and $(0, -1)$ (indicated as 'stars') respectively in the environment.

We show two cases where the herding starts immediately after rendezvous process is initialized (at $t = 1$, first case) and the herding starts in between the rendezvous process (at $t = 100$, second case). We do not show the case where the herding starts after the rendezvous is completed (when the stop condition is reached), because in this case both the rendezvous and the herding tasks are disconnected.

1) Results of Scenario 4: Figure 8 presents the evolution of trajectories in Scenario 4. In Fig. 8, the initial robot positions are denoted by hollow circles for member robots and solid circles for the leader robots. The final destinations are indicated as stars. The herding behavior is achieved by setting a position controller of the leader robots to their final destination positions. Nonetheless, they can move toward their destinations (only) as long as they can see their immediate children within the SSR. Thus, the rendezvous process is not disrupted during the herding movement.

It can be seen that in both cases the robots achieved rendezvous while herding to the final destination by the leader robots. At $t = 100$, the trajectories of the robots indicate that the robots perform both rendezvous and herding in the first case while the robots perform rendezvous only in the second case. At $t = 200$, the herding is almost completed in the first case whereas it took up to $t = 300$ iterations to complete herding in the second case. This is because the herding started later in the second case compared to the first case.

It's worth noting that the same algorithm proposed in Sec.IV-C executed both rendezvous and herding simultaneously. This is done by setting the velocities of the leader robots to zero (for rendezvous) and to follow a trajectory to a destination position (for herding).



(a) $t = 100$. (b) $t = 200$. (c) $t = 300$.

Fig. 8: Scenario 4 - Evolution of trajectories at iterations $t = 100$, $t = 200$, and $t = 300$. The robot herding stared at $t = 1$ in the first case (top row) and at $t = 100$ in the second case (bottom row). The leader robots in both cases are 1 and 13 and the destination position for the leader robots while herding their group members are $(1, 0)$ and $(0, -1)$ respectively. The earlier the herding started, the faster the final destination was reached while simultaneously performing the rendezvous.

H. Scenario 5 - Multi-point rendezvous with optimization of rendezvous points

Through this scenario, we explicate the advantages of choosing optimal rendezvous points (selecting leader robots) using the strategy in Sec. IV-D. To better show this scenario, we need the higher number of robots, and hence we chose $N = 60$. Given the limited number of robots in Robotarium, we created our own simulation setup in MATLAB replicating point dynamics in a cluttered (obstacle-rich) environment of size $50m \times 50m$ with $N = 60$ and $M = 2$.

In the simulation of Scenario 5, we applied a constraint on sensing ability such that a robot can sense another robot only if there is a line of sight path between them (obstacle-free movements). The simulation settings for this scenario are $\epsilon = 0.05m$, $S_R = 10m$, $SSR = \epsilon$, $S_m = 0.3\,\mathrm{m/s}$. Other parameter settings are the same as that of the previous scenarios. The sampling rate at which the algorithm runs is 10 Hz. A sample trial is shown in Fig. 9 in which the distribution of robots along with their connectivity links are captured.

As with the previous scenario, we do not implement an obstacle avoidance algorithm in robot movements. However, to deal with the obstacles, we applied a constraint on the sensing ability. A robot can sense another robot only if there is a line of sight path between these two robots, i.e., if there are no obstacles on the way. The parameters settings ensure that all the robots follow obstacle-free paths while tracking their hierarchical parents to rendezvous at its root node.

We compare the performance of optimal rendezvous points against randomly selected rendezvous points in a Monte-Carlo simulation with 50 trials (each with random initial positions forming a connected graph). The settings for the optimization constraints are chosen as $L_m = 6$ and $U_m = 45$ while T_D is unconstrained to enable longer feasible paths to the leaders.

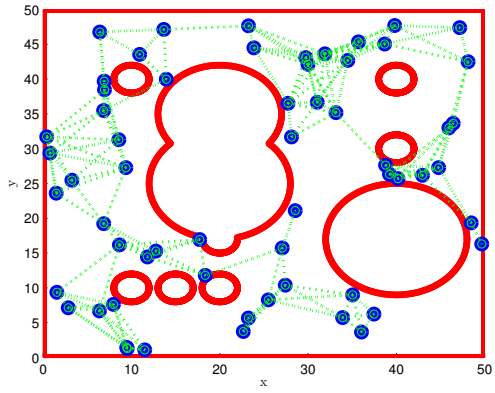


Fig. 9: Scenario 5 - An example trial showing the random deployment of initial positions of all robots (indicated as blue circles) in an obstacle-rich environment. The green dotted lines represent the connectivity links between the robots. The red curves indicate the boundaries of the obstacles.

1) Results of Scenario 5: Figure 10 shows the convergence plot in Scenario 5 with the L values averaged over the iterations. The Table III shows the results of the evaluation metrics in Scenario 5 - Energy E (Eq. 2) and the number of iterations to reach the stop condition k_{stop} (Eq. 15) - their mean and standard deviation (STD) over all the trials.

Selection of Leader Robots	D_{sum} (m)		k_{stop} (#)	
	Mean	STD	Mean	STD
Random (Monte-Carlo)	1747	584	2316	751
Optimized (Sec. IV-D)	1160	134	1234	129

TABLE III: Results of the metrics E and k_{stop} in Scenario 5.

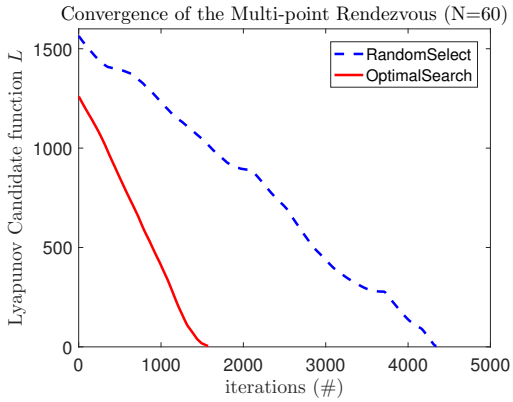


Fig. 10: Convergence plot in Scenario 5 - Comparison of L (averaged over 50 iterations) in case of optimally selected leader robots using the proposed algorithm in Sec. IV-D against randomly selected leader robots.

It can be observed from Fig. 10 that the simulations with optimized leader robots resulted in a significantly faster convergence compared to that of the simulations with randomly selected leader robots. It is important to note that the selection of leader robots may be due to the robot capabilities in some applications where such optimization is not possible. Nevertheless, in applications where it is feasible to optimize the leader robots, the proposed strategy can be greatly beneficial.

From the Table III, we can see that the mean and standard deviation of evaluation metrics (E and k_{max}) are considerably better when the optimal search method in Sec. IV-D is used compared to the case where the optimal search method is not used. This means that the rendezvous algorithm quickly achieved rendezvous (the robots traveled shorter distances in total) because of having optimal rendezvous points as a result of the optimization. This is an indirect reduction of total energy consumption of the robots because the robot energy costs are dominated by the movements (locomotion energy) [21].

Although the optimization algorithm (Sec. IV-D) is scalable for sparse graphs, we are exploiting other methods such as Linear Programming to choose optimal leader robots from an initial dense graph as part of our future work.

VI. Conclusions

In this paper, we propose a novel coordinate-free multi-point rendezvous strategy and a new, efficient hierarchical rendezvous algorithm to gather robots in different groups. This is a step towards achieving autonomous multi-point recharging and solving coordinated multi-point flocking control problems.

Through extensive simulation experiments (with the Robotarium multi-robot testbed), we demonstrated the effectiveness of the proposed algorithm along with its properties such as guaranteed convergence, connectivity maintenance, and tolerance to intermittent connectivity and identifiable mobility faults. We have also shown that the proposed algorithm outperformed the state-of-the-art consensus controllers in terms of convergence rate and capability.

Moreover, we introduced and validated an application of *multi-group herding of multi-agent systems*, aided by the proposed multi-point rendezvous algorithm. Finally, we experimentally verified a simple and effective optimization strategy for selecting rendezvous points (leader robots) that allows the robots to achieve faster convergence and travel shorter distances. The experiments and results are summarized in the video available at https://youtu.be/uaiCnw79Sb8.

Future works include devising a failure detection system aided by the hierarchical tree structure to detect a robot's communication or mobility failure by its immediate parent robot using a heartbeat communication mechanism and analyzing the history of changes in the graph for example.

Acknowledgment

This work was partially supported through the "NSF Center for Robots and Sensors for the Human Well-Being, NSF Grant No. 1439717".

References

[1] J. Lin, A. Morse, and B. Anderson, "The multi-agent rendezvous problem. an extended summary," in *Cooperative Control*. Springer, 2005, pp. 257–289.

[2] H. Park and S. A. Hutchinson, "Fault-tolerant rendezvous of multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 565–582, June 2017.

[3] R. Zheng and D. Sun, "Multirobot rendezvous with bearing-only or range-only measurements," *Robotics and Biomimetics*, vol. 1, no. 1, p. 4, Oct 2014.

[4] M. Zavlanos, M. Egerstedt, and G. Pappas, "Graph theoretic connectivity control of mobile robot networks," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1525–1540, 2011.

[5] C. Vrohidis, P. Vlantis, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Reconfigurable multi-robot coordination with guaranteed convergence in obstacle cluttered environments under local communication," *Autonomous Robots*, pp. 1–21, 2017.

[6] A. Pierson and M. Schwager, "Bio-inspired non-cooperative multi-robot herding," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 1843–1849.

[7] N. Mathew, S. L. Smith, and S. L. Waslander, "A graph-based approach to multi-robot rendezvous for recharging in persistent tasks," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3497–3502.

[8] B. Li, B. Moridian, and N. Mahmoudian, "Underwater multi-robot persistent area coverage mission planning," in *OCEANS 2016 MTS/IEEE Monterey*, Sept 2016, pp. 1–6.

[9] M. Meghjani, S. Manjanna, and G. Dudek, "Multi-target rendezvous search," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 2596–2603.

[10] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The robotarium: A remotely accessible swarm robotics research testbed," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1699–1706.

[11] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on automatic control*, vol. 48, no. 6, pp. 988–1001, 2003.

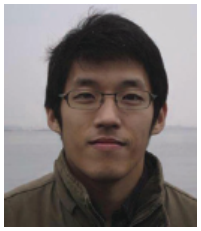
[12] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita, "Distributed memoryless point convergence algorithm for mobile robots with limited visibility," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 818–828, 1999.

[13] D. V. Dimarogonas and K. J. Kyriakopoulos, "On the rendezvous problem for multiple nonholonomic agents," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 916–922, May 2007.

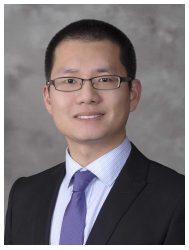
[14] L. Sabattini, C. Secchi, N. Chopra, and A. Gasparri, "Distributed control of multirobot systems with global connectivity maintenance," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1326 – 1332, 2013.

[15] J. Alonso-Mora, E. Montijano, M. Schwager, and D. Rus, "Distributed multi-robot formation control among obstacles: A geometric and optimization approach with consensus," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5356–5363.
[16] J. Yu, S. M. LaValle, and D. Liberzon, "Rendezvous without coordinates," *IEEE Transactions on Automatic Control*, vol. 57, no. 2, pp. 421–434, Feb 2012.
[17] H. Park and S. Hutchinson, "Robust rendezvous for multi-robot system with random node failures: an optimization approach," *Autonomous Robots*, Feb 2018.
[18] M. Ji, A. Muhammad, and M. Egerstedt, "Leader-based multi-agent coordination: controllability and optimal control," in *2006 American Control Conference*, June 2006.
[19] Z. Chen and H. T. Zhang, "Analysis of joint connectivity condition for multiagents with boundary constraints," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 437–444, April 2013.
[20] H.-T. Zhang, Z. Chen, and M.-C. Fan, "Collaborative control of multivehicle systems in diverse motion patterns," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 4, pp. 1488–1494, 2016.
[21] R. Parasuraman, K. Kershaw, P. Pagala, and M. Ferre, "Model based on-line energy prediction system for semi-autonomous mobile robots," in *Intelligent Systems, Modelling and Simulation (ISMS), 2014 5th International Conference on*. IEEE, 2014, pp. 411–416.
[22] T. Setter and M. Egerstedt, "Energy-constrained coordination of multi-robot teams," *IEEE Transactions on Control Systems Technology*, 2016.
[23] P. Lin, W. Ren, H. Wang, and U. M. Al-Saggaf, "Multiagent rendezvous with shortest distance to convex regions with empty intersection: Algorithms and experiments," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, pp. 1–9, 2018.
[24] P. Zebrowski, Y. Litus, and R. T. Vaughan, "Energy efficient robot rendezvous," in *Fourth Canadian Conference on Computer and Robot Vision*, Canada, 2007.
[25] K. Krishnanand and D. Ghose, "Theoretical foundations for rendezvous of glowworm-inspired agent swarms at multiple locations," *Robotics and Autonomous Systems*, vol. 56(7), pp. 549 – 569, 2008.
[26] B. W. Douglas, *Introduction to Graph Theory*, 2nd ed. Illinois, USA: Prentice Hall, 2001.
[27] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton University Press, 2009, electronically available at http://coordinationbook.info.
[28] "Graphs of maximum diameter," *Discrete Mathematics*, vol. 102, no. 2, pp. 121 – 141, 1992.
[29] S. M. Lavalle, *Planning Algorithms*. Cambridge University Press, 2006.
[30] J. P. La Salle, *The stability of dynamical systems*. SIAM, 1976.
[31] E. Montijano and C. Sagüés, "Robotic networks and the consensus problem," in *Distributed Consensus with Visual Perception in Multi-Robot Systems*. Springer, 2015, pp. 9–19.
[32] J. Kim, "Cooperative exploration and networking while preserving collision avoidance," *IEEE transactions on cybernetics*, vol. 47, no. 12, pp. 4038–4048, 2017.
[33] B.-C. Min, R. Parasuraman, S. Lee, J.-W. Jung, and E. T. Matson, "A directional antenna based leader–follower relay system for end-to-end robot communications," *Robotics and Autonomous Systems*, vol. 101, pp. 57–73, 2018.
[34] F. Arrichiello, A. Marino, and F. Pierri, "Distributed fault detection and recovery for networked robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 3734–3739.
[35] R. Vaughan, N. Sumpter, J. Henderson, A. Frost, and S. Cameron, "Experiments in automatic flock control," *Robotics and autonomous systems*, vol. 31, no. 1, pp. 109–117, 2000.
[36] S. Pettie, "A new approach to all-pairs shortest paths on real-weighted graphs," *Theoretical Computer Science*, vol. 312, no. 1, pp. 47 – 74, 2004, automata, Languages and Programming.
[37] D. Pickem, M. Lee, and M. Egerstedt, "The gritsbot in its natural habitat-a multi-robot testbed," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4062–4067.
[38] S. Caccamo, R. Parasuraman, L. Freda, M. Gianni, and P. gren, "Rcamp: A resilient communication-aware motion planner for mobile robots with autonomous repair of wireless connectivity," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 2010–2017.

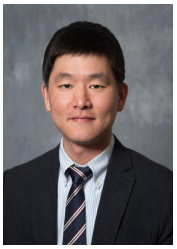
Ramviyas Parasuraman is currently a postdoctoral researcher at Purdue University. His research mainly focuses on networked robot systems, rescue robots and teleoperation. From 2014 to 2016, Dr. Parasuraman held a postdoctoral research position at KTH Royal Institute of Technology, where he worked in two European projects TRADR and RECONFIG. Previously, he worked at the European Organization for Nuclear Research (CERN) as a Marie-Curie Fellow from 2011-2014 and received his Ph.D. in Robotics and Automation from Universidad Politcnica de Madrid, Spain, in 2014. He received M.Tech from Indian Institute of Technology Delhi (2010) and B.Engg (2008) from Anna University, India. Dr. Parasuraman serves as a reviewer and a program committee member in several journals and conferences in robotics.



Jonghoek Kim is an Assistant Professor in the Department of Electrical and Computer Engineering at Hongik University, Republic of Korea. His research is on target tracking, control theory, robotics, multi-agent systems, and optimal estimation. He worked as a senior researcher at Agency for Defense Development in Republic of Korea from 2011 to 2018. In 2011, he earned a Ph.D. degree co-advised by Dr. Fumin Zhang and Dr. Magnus Egerstedt at Georgia Institute of Technology, USA. Jonghoek Kim received his M.S. in Electrical and Computer Engineering from Georgia Institute of Technology in 2008 and his B.S. in Electrical and Computer Engineering from Yonsei University, Republic of Korea in 2006.



Shaocheng Luo is a Ph.D. student in Robotics in the Department of Computer and Information Technology at Purdue University (West Lafayette). Prior to beginning his Ph.D. program in 2014, Mr. Luo obtained his B.S. degree in Mechanical Engineering from Harbin Institute of Technology, China in 2009, and M.S. degree in Mechatronic Engineering from Zhejiang University, China in 2012. His research interests include multi-robot systems, robotic coordination control, and wireless communication. In addition to pursuing his Ph.D. degree, Mr. Luo has developed interests in robotic system applications in environmental monitoring and operations and Cyber-Physical Systems.



Byung-Cheol Min is an Assistant Professor in the Department of Computer and Information Technology at Purdue University. He directs the SMART Lab at Purdue University, which performs research in multi-robot systems, robotic sensor networks, and human-robot interaction with emphasis in field robotics and assistive technology and robotics. From 2014 to 2015, prior to his faculty position at Purdue University, Dr. Min held a postdoctoral fellow position at the Robotics Institute of Carnegie Mellon University. He received his Ph.D. in Technology with a specialization in Robotics from Purdue University, West Lafayette, IN USA, in 2014. He received his M.S. degree in Electronics and Radio engineering in 2010 and B.S. degree in Electronics Engineering in 2008 from Kyung Hee University, Republic of Korea.