Proceedings Paper:
Miyauchi, G., Lopes, Y.K. and Groß, R. (2022) Multi-operator control of connectivity-preserving robot swarms using supervisory control theory. In: Proceedings of 2022 International Conference on Robotics and Automation (ICRA). 2022 International Conference on Robotics and Automation (ICRA), 23-27 May 2022, Philadelphia (PA), USA. Institute of Electrical and Electronics Engineers, pp. 6889-6895. ISBN: 9781728196824.

https://doi.org/10.1109/icra46639.2022.9812242

Multi-Operator Control of Connectivity-Preserving Robot Swarms Using Supervisory Control Theory

Genki Miyauchi[1], Yuri K. Lopes[2], Roderich Groß[1]

Abstract— **Involving human operators to support swarms of robots can be beneficial to address increasingly complex scenarios. However, the shared control between multiple operators remains a challenge, especially where communication between the operators is not available. This paper studies the problem of forming a dynamic chain of robots connecting two operators moving within an environment. The robot chain enables operators to share information and robots among themselves. Based on supervisory control theory, we propose a distributed solution which formally guarantees that the deployed robot controllers match the modeled specifications. We validate the controllers through simulations with groups of up to 40 mobile robots in an environment with obstacles, demonstrating the feasibility of the approach.**

I. INTRODUCTION

Swarm robotics studies how large groups of robots can accomplish relatively complex tasks using simple, local interactions. As swarms of robots do not rely on centralized control, they tend to cope well with faulty members, and their performance often scales reasonably well with the number of robots [1]. This makes swarms a promising technology for performing tasks that are scattered over large, unknown environments, such as those in search and rescue missions.

In many real-world scenarios, a swarm of robots would not work fully autonomously. Rather a human operator would exert some control or influence over the swarm [2]. For example, they could provide locations that the swarm must attend to, and in what order. In situations where the operator is not physically present within the environment, they could remotely operate one of the robots [3], [4]. Where the operator is physically present, they could directly interact with the swarm [5], [6], [7]. We are specifically interested in enabling multiple operators to exert shared control over a swarm of robots, which could be beneficial or even necessary in scenarios that exceed the capability of a single operator [8], [9], [10].

The ability to exchange information is often considered a prerequisite for cooperating effectively [11]. Furthermore, the ability to share control of multiple robots can contribute positively towards task performance [12]. However, exchanging information and robots between operators becomes challenging if the mission takes place in GPS-denied environments, or the operators are unable to communicate directly. An example scenario could be robots independently exploring different parts of a communication-constrained environment.

A promising approach for inter-connecting separate locations are robot chains, that is, sequences of robots positioned in such a way that they form a linear communication network between two points of interest. Robot chains can be established dynamically by a swarm of robots [13]. They can relay information, and be used to guide robots to points of interest [14]. They can be used to establish or preserve connectivity among two or more points of interest [15]. In [16], the points of interest represent tasks, and the chains are used to guide the remaining robots between these tasks. Improving the topology of such networks is considered in [15], [17], whereas robot failure is considered in [18]. The aforementioned works are not concerned with human-swarm interaction. To our knowledge, the problem of forming robot chains between two (mobile) operators has not yet been explored in the literature.

This paper proposes a method that enables two operators to exert shared control over a swarm. While performing spatially distributed tasks in the environment, the swarm has to preserve the connectivity among its members and the operators at all times. The latter are either not physically present in the environment, in which case they will remotely operate dedicated lead robots, or they carry a portable device enabling them to interact with nearby robots. The design and analysis of swarm robotics controllers are both challenging [19]. We use a formal framework based on supervisory control theory (SCT) [20], which has been applied for controlling groups of autonomous robots though without the involvement of humans [21], [22]. In SCT, the capabilities of the system (here, the robots) as well as their specifications are expressed using formal languages. The control logic is then obtained from these languages in an automatic fashion. This is in contrast to existing works on robot chain formation, which, although building on formal representations, implemented the control logic in an ad-hoc manner.

The paper is structured as follows. Section II formulates the problem. Section III describes the proposed methodology. Section IV presents the finding obtained from the computer simulations that are used to evaluate the approach. Section V concludes the paper.

II. PROBLEM FORMULATION

The problem scenario involves two human operators with shared control over a group of robots. Their objective is to complete a set of tasks that are scattered in the environment.

[1] G. Miyauchi and R. Groß are with Sheffield Robotics, The University of Sheffield, Sheffield, United Kingdom {g.miyauchi, r.gross}@sheffield.ac.uk

[2] Y. K. Lopes is with Department of Computer Science, Santa Catarina State University, Joinville, Brazil yuri.lopes@udesc.br

The robots, also called *workers*, can perform all of these tasks.

The environment is represented as $\mathcal{W} = \mathbb{R}^2$. It may contain simple obstacles, $\mathcal{O} \subset \mathcal{W}$, such as walls. Let $\mathcal{W}_{\text{free}} = \mathcal{W} \setminus \mathcal{O}$. Each task is defined by tuple $(A, w, n^{\min})$. $A \subset \mathcal{W}_{\text{free}}$ represents the region from which the workers can perform the task. Each region is assumed to be connected, and their sets are disjoint. The quantity of work to be performed is discrete, with $w \in \mathbb{N}_0$ denoting the initial task *demand*. The quantity of work monotonically decreases provided that $n^{\min}$ or more workers perform that task. Formally,

$$w[k+1] = \begin{cases} \max(0, w[k] - n[k]), & \text{if } n[k] \geq n^{\min} \\ w[k], & \text{otherwise} \end{cases} \quad (1)$$

where $n[k]$ is the number of workers performing the task at time k.

Each operator is assisted by a dedicated *lead agent*. The lead agent is unable to perform the tasks. The operator can provide instructions to and obtain information from their agent at all times. The lead agent in turn can provide instructions to and obtain information from any worker or the other lead agent, provided they reside within its local communication range. By doing so, the operators can indirectly exert control over the group of robots. For example, they can request the workers to start or stop performing tasks. For a task to be performed (by any workers), a lead agent has to be present within the region.

We consider two scenarios. In the first one, the operators are not physically present in the environment and the lead agents are mobile robots. The operators remotely control the movements of the lead agents. In the second scenario, the operators are physically present in the environment, and the lead agents are portable devices that the operators have with them, as they move around. In principle, a combination of these scenarios where one operator is physically present, while the other is not, could be considered.

We assume that all robots (i.e. workers and lead agents) are represented by disks. The robots' communication network is an undirected graph. The nodes of this graph are the robots. An edge of this graph represents a pair of robots that can communicate. To do so, the distance between the two (measured from the centers of the corresponding disks) must not exceed the communication range, r_{com}. Moreover, there must be direct line of sight between the two (i.e., no obstacles or other robots in between). All robots that a given robot is able to communicate with are termed that robot's *neighbors*.

We assume that all task specifications $(A, w, n^{\min})$ are known only to the operators. We assume the communication network to be connected at the start of the mission. The objective of the operators is to minimize the time to complete all tasks. The objective of the robots (i.e., workers and lead agents) is to maximize the number of workers that are performing tasks, or are available to do so, while maintaining global connectivity in the communication network at all times.

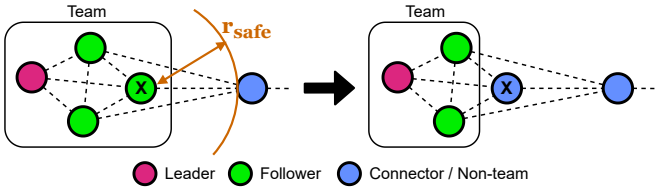


Fig. 1. A situation prior (left) and after (right) a robot joins a chain at its tail, thereby extending it. The box represents a team. Dashed lines represent connections between robots. The robot marked X is sufficiently far away from the connector (condition C1) and observes that it is closer to the connector than its peers (condition C2). It requests leaving its team to become a connector, which is approved by the blue agent.

III. METHODOLOGY

We use a team-centered approach to address the overall problem. At any moment in time each worker is either (i) in team 1, which is led by the first lead agent, (ii) in team 2, which is led by the second lead agent, or (iii) part of a chain that shall connect the two teams. Where a worker is part of a team, we refer to it as a *follower*. Followers accompany their lead agent and execute task-performing behaviors when signaled to do so by the operator. Where a worker is part of a chain, we refer to it as a *connector*.

The connectors do not move. This facilitates connections to neighboring connectors to be preserved. However, as the lead agents move, it will be necessary to make modifications to the chain. Two types of modifications are considered: adding a chain member and removing one.

Fig. 1 illustrates the situation that a new chain member is added. In other words, a follower of a given team must switch to the connector role, thereby extending the chain. The new connector allows the team at the *tail* of its chain to explore the environment further without becoming disconnected.

For follower i to become a connector, two conditions have to be met:

- $C1$: There exists at least one non-team neighbor, and the distance to all non-team neighbors is greater than a safety limit $r_{\text{safe}} + \varepsilon$.
- $C2$: There exists a non-team neighbor, j, that is closer to follower i than to any other team neighbor that is connected to j.[1]

It is possible for more than one follower to satisfy conditions C_1 and C_2. To ensure that only one follower becomes a connector, each follower satisfying both conditions sends a *request* message to the lead agent (or equivalent connector at the tail of the chain, if one has already been established). The agent (or connector) picks the first request received from a follower and sends a message allowing this follower to leave the team. Flooding of messages is used to share request and accept messages throughout the team.

Fig. 2 illustrates the situation that a chain member is removed. In other words, a connector switches to the follower

[1]Note that although follower i can establish the relative distances between its neighbors, it cannot detect whether there are any obstacles in between them. To establish whether two neighbors are directly connected with each other, it needs to communicate with one of them.

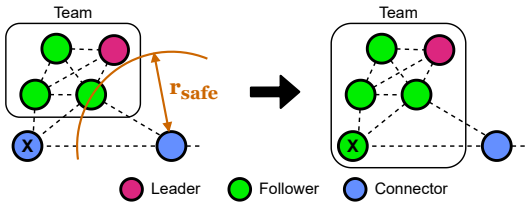


Fig. 2. A situation prior (left) and after (right) a robot leaves a chain, thereby shrinking it. The robot marked X is a connector at the tail of the chain (condition F1) and observes that the team is sufficiently close to the preceding connector (condition F2), rendering itself redundant. It leaves the chain to join the nearby team.

role, thereby shrinking an unnecessarily long chain. The robot joins the closest team.

For connector i to become a follower, two conditions have to be met:

- $F1$: Connector i is at the tail of the chain.
- $F2$: Let $u \in \{1, 2\}$ be the team that is associated with the tail of the chain in the predecessor's direction. There exists a member of team u that is closer to the successor of connector i than safety limit $r_{\text{safe}} - \varepsilon$. An analogous condition is specified for the other team, only one of the conditions needs to be satisfied to fulfill $F2$.

A. SCT Formulation

This section describes how we model the aforementioned chain management behavior using SCT. Our approach is based on [21], which explains SCT fundamentals in detail.

The robot's state evolves via uncontrollable and controllable events. Uncontrollable events represent control inputs, for example, the instructions that operators may issue at any time. Controllable events represent actions that can be triggered by the robot's controller itself, such as performing the locally available task. Each event is either private or public [23]. In this work, we use public events to formally model how messages impact the robots as they propagate through the network. If a robot's controllable event is public, it triggers a corresponding uncontrollable event in neighboring robots.

When designing an SCT controller one needs to define formal languages that express (i) what the robot can do in principle (called *free behavior models*) and (ii) how it should behave (called *control specifications*). The corresponding languages are then automatically combined to produce the controller (called the *supervisor*) that is executed on each robot of a certain type.

All our models are represented by *generators* (which in principle can produce any regular language). They are defined as a 7-tuple [23],

$$G = (Q, \Sigma, \delta, q_0, Q_m, \Sigma^{ext}_{u,pub}, M) \tag{2}$$

where Q is the finite set of states, Σ comprises private controllable and uncontrollable events $\Sigma_{u,priv}$, $\Sigma_{c,priv}$ as well as public controllable and uncontrollable events $\Sigma_{u,pub}$, $\Sigma_{c,pub}$, $\delta : Q \times \Sigma \to Q$ is a partial transition function, $q_0 \in Q$ is the initial state, $Q_m \subseteq Q$ is the set of marked states, which are states that are considered safe for the system, $\Sigma^{ext}_{u,pub}$ is the set of public uncontrollable events in G and other generators, and $M : (\Sigma_{c,pub} \cup \Sigma_{u,pub}) \to \Sigma^{ext}_{u,pub}$ is a mapping from public events to the related public uncontrollable events.

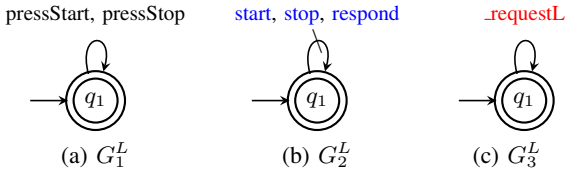


Fig. 3. Free behavior models for the lead agents representing their ability (a) to receive control inputs from the operator, (b) send messages to the workers, (c) and receive messages from them (for details, see text). States are represented by circles. The initial state is indicated by an unlabeled arrow. Marked states are represented by double-line circles. Transitions and associated events are shown as labeled arrows. Arrows with a stroke relate to controllable events, and arrows without a stroke relate to uncontrollable events. Public controllable events and public uncontrollable events are shown in blue and red respectively.

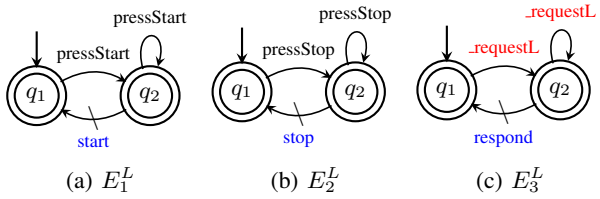
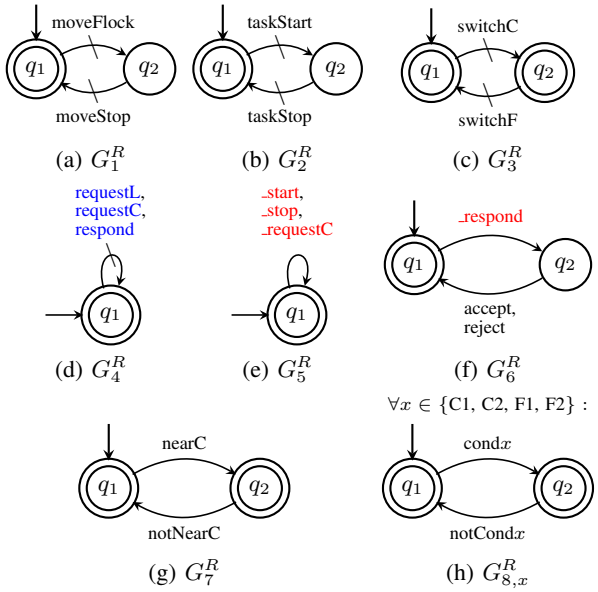


Fig. 4. Control specification for the lead agents allowing them (a–b) to transmit a signal upon receiving the corresponding operator input, and (c) to send a response when a request from a follower to become a connector is received (for details, see text).

In the following we detail the SCT models used by the lead agents and workers, respectively.

1) Lead agent models: We assume that the lead agent is a robot that is remotely controlled by a human operator. As the role of the lead agent is to instruct the followers' supervisors through public events, only communications (but not its movement) are modeled. Fig. 3 shows the free behavior models of the lead agent. G_1^L represents the operator input device used to control the lead agent. The events correspond to the operator pressing a start (*pressStart*) or stop (*pressStop*) button to indicate whether followers should execute a task. G_2^L represents the lead agent's ability to send messages to neighboring robots. The lead agent can send task-related messages (events *start* and *stop*) and respond to requests received from followers to become a connector (event *respond*). G_3^L represents the lead agent's ability to receive such requests (event *_requestL*) from workers to become a connector.

Fig. 4 shows the control specifications of the lead agents. E_1^L and E_2^L define the public controllable events that are enabled once the corresponding operator input is received. E_3^L defines the message a lead agent sends upon receiving a request from a follower that wishes to become a connector. Multiple such requests can be received, and the decision to accept or reject these is handled by the operational procedures of the lead agent (i.e. within call-back function of the corresponding event). Only the first request is ac-

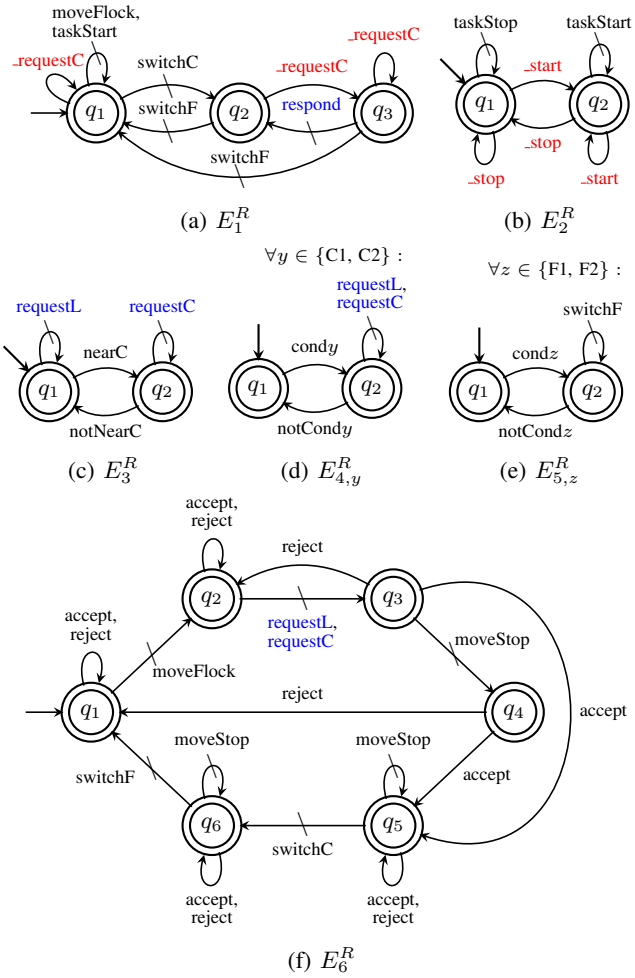


(a) G_1^R (b) G_2^R (c) G_3^R (d) G_4^R (e) G_5^R (f) G_6^R (g) G_7^R (h) $G_{8,x}^R$

Fig. 5. Free behavior models for the worker robots representing their ability (a) to either flock or remain stationary, (b) to work on a task at their current location, (c) to switch between the follower and connector roles, (d) to transmit and (e) receive messages, (f) to process the responses received, (g) to detect nearby connectors, and (h) to determine whether the conditions $C1, C2, F1, F2$ are satisfied (for details, see text).

cepted (for any period without connectors). The operational procedures also ensure that request and respond messages include identifying information, helping to respond positively to specific robots.

2) Worker models: Fig. 5 shows the free behavior models of the workers. G_1^R represents robot motion. It allows the robot to either remain stationary (state q_1) or to flock with its team (state q_2). Event *moveFlock* enables the flocking behavior, whereas event *moveStop* enables the robot to stop. The actual flocking behavior is realized in the operational procedures (see section III-B). This abstraction allows reusing the same SCT models on multiple platforms (e.g. legged and wheeled robots), provided that a connectivity-preserving flocking behavior can be implemented on such platform. G_2^R represents the worker's ability to execute a task-performing behavior. Event *taskStart* and *taskStop* start and suspend task execution. G_3^R represents the ability to change the role to follower (event *switchF*) and connector (event *switchC*). A worker is initially a follower (state q_1). G_4^R and G_5^R represent the worker's ability to send and receive messages from other neighboring robots. The worker's request to become a connector is represented as *requestL* and *requestC*, which are sent to a lead agent or a connector respectively. G_6^R represents the worker's ability to process the response to a request it has previously sent. When the event *_respond* is received, the operational procedures internally triggers either event *accept* or *reject*, depending on whether the robot matches the identifying information that was appended with event *_respond*. G_7^R represents whether a worker detects a connector (event nearC) or not (event notNearC). Finally, G_8^R represents whether the conditions we have defined for switching between roles are satisfied. $G_{8,C1}^R$ and $G_{8,C2}^R$ monitor the conditions to switch to a connector, whereas $G_{8,F1}^R$ and $G_{8,F2}^R$ monitor the conditions to switch to a follower.



(a) E_1^R (b) E_2^R (c) E_3^R (d) $E_{4,y}^R$ (e) $E_{5,z}^R$ (f) E_6^R

Fig. 6. Control specifications for worker robots allowing them (a) to perform certain actions depending on their role, as a follower to start tasks and flock, as a connector, to send a response when a request from a follower to become a connector is received, (b) to start or stop performing tasks when the corresponding signals are received by the lead agent, (c) to determine whom request messages should be sent to, (d) to send request messages for becoming connectors when the conditions are satisfied, (e) to become a follower when the conditions are satisfied, and (f) to become a connector when the received response was accepting them (for details, see text).

Fig. 6 shows the control specifications of the worker robots. E_1^R specifies the actions that are enabled in the follower and connector state, respectively. As a follower (state q_1), the robot is allowed to start tasks and flock. As a connector (state q_2), it is allowed to respond to requests from followers. While being at the tail of the chain, it accepts the first such request. E_2^R ensures a worker starts or stops working on tasks when a corresponding signal was received from the lead agent. E_3^R specifies that if a worker detects a connector (event *nearC*), the request must be sent to the connector. Otherwise, the request is sent to the lead agent. $E_{4,C1}^R$ and $E_{4,C2}^R$ ensure together that a request to become a connector can be made only when conditions C1 and C2 are both satisfied. Similarly, $E_{5,F1}^R$ and $E_{5,F2}^R$ together allow a

robot to switch to a follower only when conditions F1 and F2 are both satisfied. E_6^R describes the process for a follower to become a connector. The first action a follower must perform is to flock with the team (state q_2). If the follower decides to send a request to become a connector (state q_3), it must stop moving and wait for a reply (state q_4). The follower will only switch to a connector if the request was accepted.

We synchronize the free behavior models and specifications using local modular synthesis [24], [23]. After synchronization, the local modular supervisors of lead agents have a total of 6 states and 25 transitions (sum of 3 supervisors), whereas the follower robots have 46 states and 196 transitions (sum of 8 supervisors).

B. Flocking Behavior

Flocking is a research topic of its own [25], [26]. In this paper, it enables robots in the follower state to accompany the lead agent, as the latter moves within the environment. It is not used by robots in any other state.

The robot platform used in this study is the e-puck [27], which is a mobile differential-wheeled robot. The e-puck has a circular body of radius 3.5 cm. Its body is equipped with eight proximity sensors, which are distributed along its perimeter. We assume a range-and-bearing system to provide the relative positions of nearby agents. We use $\mathbf{p}_{ij}$ to denote robot j's position in the local coordinate system of robot i.

Similar to [15], [25], the robot uses virtual forces to determine its direction of movement. The virtual force of robot i is given by

$$\mathbf{u}_i = \alpha \mathbf{u}_i^{\text{attract}} + \beta \mathbf{u}_i^{\text{repulse1}} + \gamma \mathbf{u}_i^{\text{repulse2}} \tag{3}$$

where α, β and γ are positive scalars to weigh the influence of the force components.

Force component $\mathbf{u}_i^{\text{attract}}$ causes robot i to follow the lead agent. It is defined as

$$\mathbf{u}_i^{\text{attract}} = \frac{1}{|N_i^{\text{attract}}|} \sum_{j \in N_i^{\text{attract}}} \frac{\mathbf{p}_{ij}}{||\mathbf{p}_{ij}||} \tag{4}$$

where N_i^{attract} denotes the set of neighbors of robot i that are either the lead agent or members of the same team that are closer to the lead agent than robot i (based on hop count).

Force component $\mathbf{u}_i^{\text{repulse1}}$ causes robot i to get repelled from all neighboring robots. It is defined as

$$\mathbf{u}_i^{\text{repulse1}} = -\frac{1}{|N_i|} \sum_{j \in N_i} \frac{1}{d_{ij}^3} \frac{\mathbf{p}_{ij}}{||\mathbf{p}_{ij}||} \tag{5}$$

where N_i is the set of neighbors of robot i.

In addition to $\mathbf{u}_i^{\text{repulse1}}$, the force component $\mathbf{u}_i^{\text{repulse2}}$ uses proximity sensors to prevent collisions if the distance to an obstacle gets too close. It is defined as

$$\mathbf{u}_i^{\text{repulse2}} = -\frac{1}{8 d_{\text{max}}} \sum_{j=1}^{8} (d_{\text{max}} - d_j) \hat{\mathbf{v}}_j \tag{6}$$

where $d_{\text{max}} = 10\,\text{cm}$ is the assumed range of the proximity sensors, d_j is the distance extracted from the j^{th} sensor, and $\hat{\mathbf{v}}_j$ is the unit vector pointing from the robot's center to the j^{th} sensor. Where sensor j detects no object, we set $d_j = d_{\text{max}}$.

C. Message Relaying

In addition to the SCT models discussed, we realize an extension that enables messages to be relayed between the operators. It is straight forward to realize this via public events.[2] When an operator chooses to send a message to the other operator, its lead agent broadcasts the message to its neighboring robots. Using a hop count comparison (which is realized via the operational procedures), the workers establish whether a connector is preceding or succeeding with respect to the particular team target. This allows messages to be efficiently relayed until they reach the other team agent.

D. Robot Exchange

We realize a further extension to enable operators to dynamically exchange workers via the chain[2]. The request for additional workers is sent to the other lead agent using the message relaying extension described in the previous section. To define the behavior of a worker to travel along the chain, we introduce a new role called *traveler*. Upon receiving a signal from the lead agent, a follower switches to a traveler and starts moving along the chain to join the other team. We implement a chain following behavior using a PD controller that makes a traveler move along the left side of the chain. Once the traveler detects a robot from the team to join, it becomes a follower of that team.

IV. RESULTS

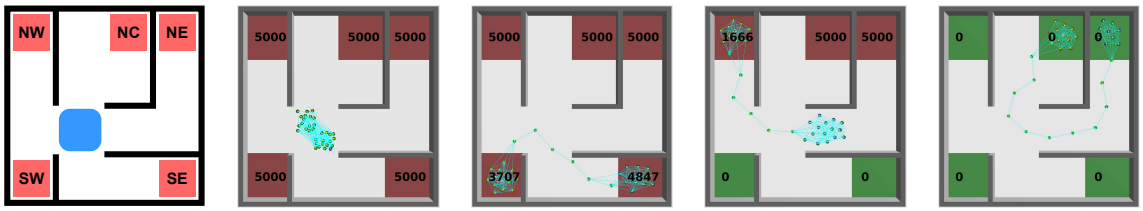
We evaluate the performance of our method in simulation using the ARGoS simulator [29]. Trials were conducted in a 4 m×4 m arena containing walls [see in Fig. 7(a)]. Five tasks were distributed in the arena. The blue region represents the area where the lead agents and workers were initially deployed, separated as two team-clusters. The initial position of the robots within each cluster was randomized in each trial. We use a communication range of $r_{\text{com}} = 0.8\,\text{cm}$ for all robots and flocking weights $\alpha = 1$, $\beta = 6000$, and $\gamma = 15$ in all trials. Video recordings of the simulation can be found in the supplementary material.

A. Performance of Chain Management

We first analyze the workers' ability to maintain a chain between the lead agents. Each task is given an initial demand of $w = 5000$ and require $n^{\text{min}} = 1$ robot to perform. Simulated operators are used to drive the lead agents to task locations and signal their workers to start or stop performing tasks. Both operators send one heart-beat messages per second to each other, which then should be relayed via the chain. We record whether these messages are successfully received. Fifty trials are performed with 20, 30, and 40 workers, respectively, that are split equally among the teams at start. Trials are terminated once all tasks are completed or once 700 seconds have elapsed, whichever happens first.

Fig. 7(b–e) shows a sequence of snapshots from a typical trial using 30 workers. The two operators first ask their teams to complete tasks SW and SE, respectively, then move to

[2]For the extended models, see the online supplementary material [28].



(a) (b) 0s (c) 50s (d) 175s (e) 347s

Fig. 7. (a) The bounded arena with walls. Red areas represent task locations. The two lead agents and all workers are deployed in the blue area. (b–e) show sequence of snapshots taken from one of the simulations where 30 workers maintain a chain between two lead agents while performing the tasks. The numbers represent the task's remaining task demand. (b) The robots in their initial positions. (c–e) The simulation after 50, 175, and 347 seconds.

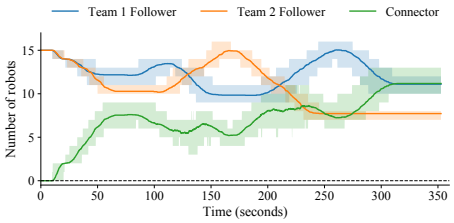


Fig. 8. Average number of 30 workers in each role. Followers are shown separately according to their team. The solid lines represent the mean and the transparent regions represent the minimum and maximum values.

TABLE I

SIMULATION RESULTS FOR CHAIN MANAGEMENT

No. of robots	Completion time (s)	Final no. of connectors	Successful message (%)
20	517.55	10.68	97.7
30	340.17	11.18	98.4
40	298.98	11.91	97.3

focus on tasks NW, NE and NC. Fig. 8 shows the average number of workers in each role when using 30 workers. After initial deployment, the number of followers in each team reaches back to 15 workers for team 1 at around 260 seconds and for team 2 at around 165 seconds. This demonstrates the worker's ability to both build a chain to explore new areas but then subsequently shrink it, recovering the original state.

The full results are reported in Table I. While task completion time decreased as more workers were used, the number of connectors at the end of the mission remained similar across all trials. This highlights how the number of connectors used to maintain a connection between the two teams is similar regardless of the number of workers in each team. Over 97% of messages were successfully exchanged between the lead agents in all three team sizes. This shows that the robot chain remained connected and the workers were able to relay the message as intended by the models.

B. Performance of Robot Exchange

To validate the exchange of robots over the chain, the scenario is altered by changing the minimum number of robots needed for task NC to be $n^{\min} = 15$ and the task demand of task NW to be $w = 7500$. The leader would request as many robots that were needed at the time of discovering the task, plus two. As a control study, we compare against the case where the whole team, receiving the request for additional robots, migrates upon completing their task, rather than sending the requested robots. The trials are repeated 50 times for each method using 30 workers.

TABLE II

SIMULATION RESULTS FOR ROBOT EXCHANGE

	Completion time (s)	Wait time (s)	Distance (m)
Team Migration	482.02	164.30	100.46
Robot Exchange	461.69	87.43	21.03

The results are reported in Table II. The strategy of exchanging robots along the chain completes the whole mission significantly faster than the strategy of migrating the whole team along the chain (two-sided, Mann-Whitney test, $p<0.001$). We also analyzed the duration an operator has to wait from the moment of placing the request until the workers perform the task. For the robot exchange strategy, the wait time for task NC was almost half of that for team migration. This is because, for robot exchange, an operator can immediately send robots upon receiving a request even if it is currently working on a task. The total distance traveled by the lead agent and its workers since receiving the request is also largely reduced as only the requested number of robots needed to travel to the other team. These results illustrate how robot exchange can be used by operators to efficiently share available workers together.

V. CONCLUSION

We presented a method for two human operators to exert shared control over a swarm of robots to cooperatively perform spatially distributed tasks. We used SCT to formally design the control logic, which required only a formal description of the system and specifications, plus an implementation of all event-specific handlers. The robots autonomously established a connectivity-preserving communication network that enabled the two lead agents (or human operators) to interact while independently moving through the environment. The communication network supported the exchange of robots between operators. We validated the approach using embodied simulations. Future work will (i) enable the connectivity-preserving robots to continue moving once part of the network; (ii) consider more than two operators; and (iii) involve human participants, thereby investigating the merits of the connectivity-preserving approach in practical scenarios.

REFERENCES

[1] H. Hamann, *Swarm Robotics: A Formal Approach*. Springer, 2018.

[2] A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis, "Human interaction with robot swarms: A survey," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 9–26, 2016.

[3] M. A. Goodrich, S. Kerman, and S.-Y. Jun, "On leadership and influence in human-swarm interaction," in *2012 AAAI Fall Symposium Series*, 2012.

[4] P. Walker, S. A. Amraii, M. Lewis, N. Chakraborty, and K. Sycara, "Human control of leader-based swarms," in *2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2013, pp. 2712–2717.

[5] J. Penders, L. Alboul, U. Witkowski, A. Naghsh, J. Saez-Pons, S. Herbrechtsmeier, and M. El-Habbal, "A robot swarm assisting a human fire-fighter," *Advanced Robotics*, vol. 25, no. 1-2, pp. 93–117, 2011.

[6] B. Gromov, L. M. Gambardella, and G. A. Di Caro, "Wearable multi-modal interface for human multi-robot interaction," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2016, pp. 240–245.

[7] 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.

[8] J. M. Whetten, M. A. Goodrich, and Y. Guo, "Beyond robot fan-out: Towards multi-operator supervisory control," in *2010 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2010.

[9] S. Nagavalli, M. Chandarana, K. Sycara, and M. Lewis, "Multi-operator gesture control of robotic swarms using wearable devices," in *Proceedings of the Tenth International Conference on Advances in Computer-Human Interactions*. IARIA, 2017.

[10] J. R. Grosh and M. A. Goodrich, "Multi-human management of robotic swarms," in *International Conference on Human-Computer Interaction*. Springer, 2020, pp. 603–619.

[11] J. MacMillan, E. E. Entin, and D. Serfaty, "Communication overhead: The hidden cost of team cognition." in *Team Cognition: Understanding the Factors That Drive Process and Performance.* American Psychological Association, 2004, pp. 61–82.

[12] J. Patel and C. Pinciroli, "Improving human performance using mixed granularity of control in multi-human multi-robot interaction," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020, pp. 1135–1142.

[13] S. Nouyan, R. Groß, M. Bonani, F. Mondada, and M. Dorigo, "Teamwork in self-organized robot colonies," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 695–711, 2009.

[14] M. Dorigo *et al.*, "Swarmanoid: a novel concept for the study of heterogeneous robotic swarms," *IEEE Robotics & Automation Magazine*, vol. 20, no. 4, pp. 60–71, 2013.

[15] N. Majcherczyk, A. Jayabalan, G. Beltrame, and C. Pinciroli, "Decentralized connectivity-preserving deployment of large-scale robot swarms," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4295–4302.

[16] L. Garattoni and M. Birattari, "Autonomous task sequencing in a robot swarm," *Science Robotics*, vol. 3, no. 20, 2018.

[17] S. Yi, W. Luo, and K. Sycara, "Distributed topology correction for flexible connectivity maintenance in multi-robot systems," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[18] V. S. Varadharajan, D. St-Onge, B. Adams, and G. Beltrame, "Swarm relays: distributed self-healing ground-and-air connectivity chains," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 4, pp. 5347–5354, 2020.

[19] M. Birattari, A. Ligot, D. Bozhinoski, M. Brambilla, G. Francesca, L. Garattoni, D. Garzón Ramos, K. Hasselmann, M. Kegeleirs, J. Kuckling, F. Pagnozzi, A. Roli, M. Salman, and T. Stützle, "Automatic Off-Line Design of Robot Swarms: A Manifesto," *Frontiers in Robotics and AI*, vol. 6, p. 59, 2019.

[20] P. Ramadge and W. Wonham, "Supervisory Control of a Class of Discrete Event Processes," *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[21] Y. K. Lopes, S. M. Trenkwalder, A. B. Leal, T. J. Dodd, and R. Groß, "Supervisory control theory applied to swarm robotics," *Swarm Intelligence*, vol. 10, no. 1, pp. 65–97, 2016.

[22] J. A. Dulce-Galindo, M. A. Santos, G. V. Raffo, and P. N. Pena, "Autonomous navigation of multiple robots using supervisory control theory," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3198–3203.

[23] Y. K. Lopes, S. M. Trenkwalder, A. B. Leal, T. J. Dodd, and R. Groß, "Supervisory control of robot swarms using public events," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7193–7199.

[24] M. H. de Queiroz and J. E. R. Cury, "Synthesis and implementation of local modular supervisory control for a manufacturing cell," in *Sixth International Workshop on Discrete Event Systems, 2002. Proceedings.*, 2002, pp. 377–382.

[25] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Şahin, "Self-organized flocking in mobile robot swarms," *Swarm Intelligence*, vol. 2, no. 2, pp. 97–120, 2008.

[26] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek, "Optimized flocking of autonomous drones in confined environments," *Science Robotics*, vol. 3, no. 20, 2018.

[27] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, vol. 1, no. 1, 2009, pp. 59–65.

[28] G. Miyauchi, Y. K. Lopes, and R. Groß, "Online supplementary material," 2021. [Online]. Available: https://doi.org/10.15131/shef.data.16608421

[29] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.