Inferring and Learning Multi-Robot Policies by Observing an Expert

Pietro Pierpaoli†, Harish Ravichandar§, Nicholas Waytowich‡, Anqi Li§, Derrik Asher‡, and Magnus Egerstedt†

Abstract—In this paper we present a technique for learning how to solve a multi-robot mission that requires interaction with an external environment by repeatedly observing an expert system executing the same mission. We define the expert system as a team of robots equipped with a library of controllers, each designed to solve a specific task, supervised by an expert policy that appropriately selects controllers based on the states of robots and environment. The objective is for an un-trained team of robots equipped with the same library of controllers, but agnostic to the expert policy, to execute the mission, with performances comparable to those of the expert system. From observations of the expert system, the Interactive Multiple Model technique is used to estimate individual controllers executed by the expert policy. Then, the history of estimated controllers and environmental state is used to learn a policy for the un-trained robots. Considering a perimeter protection scenario on a team of simulated differential-drive robots, we show that the learned policy endows the un-trained team with performances comparable to those of the expert system.

I. INTRODUCTION

One of the primary issues with human control of a multirobot system, or human-swarm interaction, is the cognitive load associated with tracking and delegating tasks to each robot [1]. Along with the poor scalability of a one-to-many approach in controlling teams of robots, further exacerbation of this problem occurs when the robots are equipped with heterogeneous attributes, as the complexity of the robot-to-task allocation process can easily become intractable [2].

To circumvent the scalability issues, abstraction of the state information [3], [4] and control strategies [1], [5] of the system are often used by human operators to effective manipulate the multi-robot systems. In particular, local interaction rules between the robots can be designed such that a desired task-oriented collective behavior emerges at the ensemble level, e.g. [5], [6], [7]. However, the resulting task-oriented controllers cannot be directly implemented towards the solution of complex, real-world missions, which usually require an assortment of these task-oriented controllers (see for instance [8], [9] for implementations of this idea). Even though it is possible to enhance the expressiveness of individual controllers through their sequential combination [10], human-level intelligence and expertise is required in order to interact with unknown and unpredictable environments [11]. Inspired by the paradigm of learning from demonstration [12], [13], we propose a solution to the problem of composing multirobot primitive controllers by leveraging observations of an expert system.

In this paper, we consider an expert system composed by a team of robots solving a mission that requires interaction with an external environment. In order to make the discussion more concrete, let’s consider a perimeter protection scenario, where the the robots are tasked with preventing intruders from accessing a restricted area. In this case, we can consider the intruders as the environment external to the robots. In order to solve this mission, the policy supervising the robots selects actions from a finite list of primitives (e.g., gather at locations, assemble formations, chase intruders) by considering the state of robots and environment. In this paper, we consider the expert policy to be a deterministic automated strategy capable to solve the mission with a certain level of performance. Extensions of the proposed framework to policies obtained with a human in the loop are possible but will not investigated here.

Parallel to the expert, we consider a team of un-trained imitating robots, having access to the same list of primitive controllers used by the expert system, but agnostic to the actual policy controlling them. Our objective is a procedure that allows the imitating robots to construct a policy capable to achieve expert-like performances by collecting noisy observations of the expert system and environmental state during execution of the mission.

We note that, the idea of learning how to create sequences from a pre-defined list of controllers, rather than learning entirely new controllers, is motivated by the rich literature on coordinated control for multi-agent system [14]. In addition, although beyond the scope of the current work, the use of coordinated controllers makes our framework naturally predisposed for fully distributed implementations. Finally, by focusing on high-level strategies that compose well-understood controllers, we can interpret the actions executed by the robots.

The remaining of this paper is organized as follows. In Section I, we discuss the main formulation of the problem, along with expert and imitator dynamical models. In Section III, we describe the estimation scheme used by the imitator to infer primitive controllers from noisy observations.
of the expert. In Section IV we describe the training procedure used by imitator to learn the expert’s policy. Finally, we implement the proposed technique to a perimeter protection scenario on a team of differential-drive robots in Section V.

A. Related Work

Robotic systems capable of inferring human behavior have been studied in a variety of contexts, such as human-robot interaction (HRI) [15], [16], autonomous navigation [17], [18], and interactive learning [19], [20]. For instance, within the context of HRI, inferring human behavior is often referred to as intention inference, wherein the robot attempts to identify the underlying intention or goal of the human partner [21], [22]. In context of interactive learning, the robot aims to infer what the human partner is teaching [19]. In these contexts, it is either assumed that labelled training data can be acquired or it is sufficient to classify the unlabelled data into abstract clusters. In our work, we are interested not only in inferring the expert behavior, but also learning from such inferences.

Learning from demonstrations (LfD) is a popular paradigm in robotics that provides a plethora of techniques aimed at learning from and imitating expert behavior [12]. A vast majority of techniques, however, rely on labelled demonstrations and solve a supervised learning problem. A few exceptions to this assumption include learning reward functions from unlabelled demonstrations (e.g., [23], [24]), and learning manipulation skills from unlabelled videos (e.g., [25]).

Both the inference and the learning methods discussed thus far, however, are limited to single-robot tasks and scenarios. In the context of multi-robot systems, prior work has explored identifying joint-intention and shared plans of a group of agents [26], [27]. However, similar to examples in single-robot systems, the inferred information is not utilized to learn control policies. On the other hand, algorithms that learn multi-agent policies (e.g., [28], [29], [30]) rely on labelled expert demonstrations.

As mentioned earlier, a rich body of works exists on task-oriented controllers for multi agent teams. Coordinated controllers based on weighted consensus protocol have been used to achieve, for example, flocking [31], coverage [32], formation control [33], and cyclic pursuit [34]. Examples of alternative methods include the Null Space Methods [35], Navigation Functions [36], and Model Predictive Control [37]. Related to our contribution, solutions to the problem of composing sequences of controllers include formal methods [38], path planning [8], Finite State Machines [39], Petri Nets [40], Behavior Trees [41], and Reinforcement Learning [42], [43]. We take inspiration from a variety of communities in order to develop a unified framework to learn multi-robot controller selection policies from unlabelled and noisy observations of an expert system.

II. Problem Formulation

Let us consider a team of $N$ robots interacting with an external environment having as final objective the completion of a mission. We denote by $x(k) \in \mathcal{D}^N$ and $e(k) \in \mathcal{E}$ the robots’ and environment states at time step $k$ respectively. At each time, the motion of the robots is described by a controller, selected from a library of controllers $\mathcal{L} = \{f_1(x), \ldots, f_M(x)\}$, where each element $f_j(\cdot) : \mathcal{D}^N \to \mathcal{D}^N$ is a continuous function of the state of the robots. At any given time, all robots are driven by the same controller, which excludes the possibility of sub-teams acting in parallel.

A supervising deterministic policy $F : \mathcal{D}^N \times \mathcal{E} \mapsto \mathcal{L}$ selects controllers in $\mathcal{L}$ as function of the states of robots and environment. Then, the discrete-time dynamics of the robots is

$$x(k+1) = F(x(k), e(k)) + v(k), \quad (1)$$

where $v(k)$ is a white Gaussian additive process noise, with known variance. The policy $F$ selects controllers so that the mission is executed with a measurable performance $\Pi^F$. We refer to the system of robots with state $x$ as expert system, to $F$ as expert’s policy, and to $\Pi^F$ as the expert’s performance under the policy $F$. As mentioned above, we assume the expert’s policy $F$ to take actions based on the exact expert and environmental states. To this end, although beyond the scope of this paper, we acknowledge that appropriate estimation schemes can be considered to recover values of $x(k)$ or $e(k)$.

In addition to the expert system just described, we assume the existence of a second team of $N$ robots identical to those in the expert system, whose dynamics is described by the same library of controllers $\mathcal{L}$. We refer to this second system as imitator and we denote its state by $x_I \in \mathcal{D}^N$. The objective of the imitator is to learn an approximation of the expert policy $F$ in order to achieve performance similar to $\Pi^F$. With $\Phi : \mathcal{D}^N \times \mathcal{E} \mapsto \mathcal{L}$ as the imitator’s approximated policy, the dynamics of the imitator is

$$x_I(k+1) = \Phi(x_I(k), e(k)), \quad (2)$$

where, similarly to the expert system, we assume the imitator has access to the exact environment’s state $e(k)$.

In order to construct such approximation, the imitator collects a sequence of environmental states $E^k = \{e(k), e(k-1), \ldots, e(0)\}$ and a sequence $Z^k = \{z(k), z(k-1), \ldots, z(0)\}$ of noisy observations of the expert’s state, where

$$z(k) = h(x(k)) + w(k) \quad (3)$$

and $w(k)$ is a white Gaussian additive noise with zero mean and known variance.

In this paper we propose a two-stage procedure in order for the imitator to learn $\Phi$. First, from noisy observation of the expert’s state and exact state of the environment, the imitator creates an online estimate of the controllers being executed by the expert. Then, based on these estimates, we propose a policy approximation procedure that allows the imitator to solve the task with performance comparable to the expert’s performance on the same task.
III. INFERRING CONTROLLERS FROM OBSERVATIONS

In this section, we describe the Interactive Multiple Model (IMM) technique [44], implemented by the imitator to estimate the sequence of controllers \( f_j(\cdot) \in \mathcal{L} \) the expert executes, given a history of observations \( Z^k \). The choice of this technique is motivated by its popularity and computational efficiency and we insist that the same objective could be achieved with alternative estimation schemes.

A. Interactive Multiple Models Estimator

The IMM estimator requires a finite list of controllers (or modes) whose transitions are described by a Markov process. We denote by \( p_{ij} \) the process probability of switching from mode \( f_j \) to \( f_i \), i.e.

\[
p_{ij} = \text{Prob}(F_k = f_i \mid F_{k-1} = f_j).
\]

In addition, we define a bank of filters (e.g. Kalman filters, EKF) each corresponding to a controller in the library \( \mathcal{L} \). By combining expert’s state observations with modes’ transition probabilities, the IMM computes 1) the probability of \( f_j \) being active at time \( k \), which we denote by \( \mu_j(k) \) and 2) the estimate of the expert’s state \( \hat{x}(k|k) \). The estimation technique follows three main steps, which we briefly discuss here for completeness. We refer the reader to [45] for a review of IMM implementations and its possible variations. In the following we denote with \([u]_\pitchfork \) the outer product of vector \( u \), defined as \( uu^T \) and we use the shorthand notation \( x(k^-) \) to denote \( x(k-1) \).

a) Interaction: First, mixing probability are computed by propagating each modes’ probability through the Markov process as

\[
\mu_{ij}(k^-|k^-) = \frac{1}{C_j} p_{ij} \mu_i(k^-) \quad \forall i, j \in M
\]

where \( C_j \) is a normalizing factor. Then, we compute the effect of the modes probabilities on previous state estimates and covariance matrix:

\[
\hat{x}_{0j}(k^-|k^-) = \sum_{i=1}^{M} \hat{x}_i(k^-|k^-) \mu_{ij}(k^-|k^-)
\]

\[
P_{0j}(k^-|k^-) = \sum_{i=1}^{M} \left[ P_i(k^-|k^-) + \left[ \hat{x}_i(k^-|k^-) - \hat{x}_{0j}(k^-|k^-) \right] \mu_{ij}(k^-|k^-) \right]
\]

b) Filtering: The posterior estimates \( \hat{x}_j(k|k) \), covariance matrix \( P_j(k|k) \), and likelihood \( \Lambda_j(k) \) for each mode are computed by applying EKF iterations for each of the known modes. Then, individual mode probabilities are computed as follows:

\[
\mu_j(k) = \frac{1}{C} \Lambda_j(k) \sum_{i=1}^{M} p_{ij} \mu_i(k^-)
\]

where \( C \) is a normalizing factor.

c) Combination: Final state estimates and covariance matrix are the obtained by combining values from each filter weighted by the probability of the corresponding mode

\[
\hat{x}(k|k) = \sum_{j=1}^{M} \hat{x}_j(k|k) \mu_j(k)
\]

\[
P(k|k) = \sum_{j=1}^{M} \left[ P_j(k|k) + [\hat{x}_j(k|k) - \hat{x}(k|k)] \otimes \mu_j(k) \right]
\]

Every time a new observation is collected, the imitator performs an iteration of the procedure just described, from which we obtain the probability distribution for all the controllers in the library. Then, estimated controller being executed by the expert system a time step \( k \) is obtained from a maximum a posteriori (MAP) estimate

\[
f_k^* = \arg \max_{j \in \{1,...,M\}} \mu_j(k).
\]

B. Multi-Robot Coordinated Controllers Inference

We demonstrate the implementation of the IMM estimator to the identification of the controllers assuming an expert system composed of 6 robots. Taking inspiration from the literature on coordinated behaviors for multi-agent systems [5], we assume each of the controllers in the library \( \mathcal{L} \) to be composed by two terms. The first term corresponds to a coordinated effect between the robots, which is represented by a weighted consensus term [14]. The second term is the agent’s individual objective, use to represent, for example, one or more leaders in the team. Each controller can then be described as:

\[
f_j(x) := -L_{w,j}(x) + v_j(x)
\]

where \( L_{w,j} : \mathcal{D}^N \mapsto \mathcal{D}^N \) is the weighted Laplacian of the controller \( j \), while \( v_j : \mathcal{D} \mapsto \mathcal{D} \), represents a leader-like controller. In this case, we consider \( \mathcal{L}_b \) to be composed by the following 4 behaviors.

1) Circular formation:

\[
u_i = \sum_{j \in \mathcal{N}_i} (\|x_i - x_j\|^2 - \delta_{ij,c}^2)(x_j - x_i)
\]

where \( \delta_{ij,c} \) is the desired separation between robots \( i \) and \( j \) corresponding to a circle-shaped formation.

2) Cone formation:

\[
u_i = \sum_{j \in \mathcal{N}_i} (\|x_i - x_j\|^2 - \delta_{ij,k}^2)(x_j - x_i)
\]

where \( \delta_{ij,k} \) is the defined as above, but corresponds to a cone-shaped formation.

3) Cyclic pursuit:

\[
u_i = \sum_{j \in \mathcal{N}_i} R(\theta)(x_j - x_i) + (x_c - x_i)
\]

where \( \theta = 2r \sin \frac{\pi}{N} \), \( r \) is the radius of the cycle formed by the robots, and \( R(\theta) \in SO(2) \). The point \( x_c \in \mathcal{D} \) is the center of the cycle.
4) Leader-follower:

\[ u_i = \sum_{j \in N_i} (||(x_i - x_j)^2 - \delta_j^2||)(x_j - x_i) \quad i = 2, \ldots, 5 \]

\[ u_1 = \sum_{j \in N_i} (||(x_i - x_j)^2 - \delta_j^2||)(x_j - x_1) + (x_g - x_1) \]

where \( \delta_j \) is the desired separation between the robots, \( x_g \in D \) represents the leader’s goal, and subscript 1 denotes the leader.

In this case, since we are not interested in approximating the expert’s policy but rather to assess the performance of the IMM, we assume the expert to randomly select any of the controllers described above at Poisson instants of time. The results obtain from simulations are reported in Fig. 1. As we can observe from results in Fig. 1, the IMM correctly estimates all modes, provided a minimum sojourn time for the controllers is respected.

To illustrate our approach, we utilize a neural network (NN) to parameterize the imitator’s policy. The parameters of the network are trained using the standard back propagation algorithm. Note that the choice of neural networks over other models is motivated by its universal approximation capability [46] and is not central to our framework. Any model that is capable of sufficiently capturing the expert policy would be applicable.

V. EXPERIMENTAL VALIDATION

In this section, we implement the policy approximation technique described in the previous sections, on the observations of an expert system performing a perimeter protection mission. During expert’s execution of the mission, the imitator collects observations of the expert’s state and environment’s state, in order to estimate what controllers are used by the experts for different configurations of the environment. Then, using data from the estimation process, we train the imitator’s policy, which is finally executed by the imitator on the same scenario.

A. Perimeter Defense Scenario

Perimeter protection can be considered both as a stand-alone scenario or as a sub-task common to many games, which makes it an ideal testing scenarios for multi-agent control protocols [47]. In this scenario, the expert team is tasked with defending a region while an adversarial team, representing the environment, tries to intrude. With reference to the domain represented in Fig. 2, the intruders’ objective is to reach the inner most circular region (dark blue), which we denote by \( B \). The objective of the defenders is to prevent intruders from reaching \( B \). We now described both defenders and intruders strategies, summarized in Fig. 3.

IV. LEARNING TO COMPOSE BEHAVIORS

In this section, we introduce our approach to learn the expert policy \( F \). We propose to learn the imitator’s policy, denoted by \( \Phi \), that approximates the expert policy in mapping the current environmental and estimated robot states onto the appropriate behavior from the controller library \( \mathcal{L} \). As noted in [2], the policy is attempting to capture the general strategy of choosing the behaviors from the expert as opposed to encoding a deterministic sequence of behaviors. In order to train such a policy, we collect the training data from \( N \) episodes. Thus, the training data is given by \( D = \{D^1, D^2, \ldots, D^N\} \), where \( D^i = \{f^i, \hat{x}^i(k), e^i(k)\} \), \( \forall k \) in each episode, denotes the data associated with the \( i^{th} \) episode.

As can be seen from the notation, the training process does not assume access to either the true behavior sequence of the expert systems or the true state of the expert system. Put another way, the system is trained using inferred quantities \( f^* \) and \( \hat{x} \) provided by the IMM filter. Once the imitator policy \( \Phi \) is learned, it can be utilized to compose the individual behaviors of the imitator system in order to solve the task of interest.
in **ATTACK** mode proceed straight towards the center of region $B$. During attacks, if any of the intruders encounter a defending robot within distance less than $\varepsilon_d$, the intruder switches its state to **RETREAT** and moves back to uniformly selected points in region $S$.

2) **Defenders’ policy**: The policy employed by the defending robots acts by selecting controllers in the library $L$. Here, we assume $L$ to be composed of the same controllers described in Section III-B, namely **LEADER-FOLLOWER**, **CYCLIC-PURSUIT**, and two different formation assembling protocols **CONE FORMATION** and **CIRCULAR FORMATION**. When none of the intruders are within distance $r_R$, defending robots execute the **CIRCULAR FORMATION** inside region $b$. If either one or two intruders are between distances $r_B$ and $r_r$, defenders switch to a **CYCLIC PURSUIT** controller. Finally, if a single intruder is within distance less than $r_r$ from the center of $B$, the defenders execute the **LEADER-FOLLOWER** controller towards the intruder. If both intruders are within distance $r_r$, the policy selects the **CONE FORMATION**.

Finally, we define the mission performance $\Pi$ as the ratio between the number of intruders’ successful attacks and the total number of initiated attacks.

![Fig. 3: Attack and defense policies. Intruders actions are described by one of three possible modes: LOITER, RETREAT, and ATTACK. Defenders actions are described by four different coordinated controls, RENDEZVOUS, LEADER-FOLLOWER, CYCLIC PURSUIT, and FORMATION CONTROL.](image)

### B. Results

The perimeter protection scenario was implemented on the Robotarium simulator [48], which includes a number of features of the real robots, e.g., differential drive kinematics and collision avoidance through barrier control functions. In the following, we define an episode as the time between initiation of two consecutive intruders’ attacks. Data collected in simulation were processed by the IMM to produce training data $D$ consisting of 200 episodes. The performance of the IMM in estimating controllers executed by the expert during the perimeter defense scenario is reported in Fig. 5. We note that, in this scenario, correctly estimating **CYCLIC-PURSUIT** and **LEADER-FOLLOWER** is particularly challenging since, by definition of the expert policy, these controllers are executed for a limited period of time, and as a result, the estimator is unlikely to acquire sufficient evidence to accurately infer the underlying behavior.

![Fig. 5: IMM estimation performance in the perimeter protection scenario. Each bar represents the fraction of occurrences that a controller was correctly estimated by the IMM.](image)

Since the expert policy described in Section V-A does not depend on the state of the robots, we trained two different models one assuming knowledge of this prior information ($\Phi^{IMM^*}$) and a second without this prior knowledge ($\Phi^{IMM}$). In addition, in order to evaluate the effects of the IMM performance on the results, additional policies were trained assuming exact knowledge of the expert system’s states and controllers. Similar to the IMM-based policies, we trained one ground-truth-based policy assuming prior information ($\Phi^{GT^*}$) and one without ($\Phi^{GT}$). In the remainder of this section, we present results comparing the above mentioned four policies in conjunction the expert policy $F$.

![Fig. 6: Training and validation accuracy of the imitator neural network with ground truth data ($\Phi^{GT}$), with IMM data ($\Phi^{IMM}$), with ground truth data and prior ($\Phi^{GT^*}$), and with IMM data and prior ($\Phi^{IMM^*}$).](image)

All four policies are represented using a multi-layer neural network with two 32 neuron, fully connected layers with hyperbolic tangent activation and an output classification layer with softmax activation to classify between the four
control modes. For each policy, the training data were split into 80% training and 20% validation data. Training results are reported in Fig. 6. As one would expect, training with ground truth data results in very high training and validation performance irrespective of whether prior information about the policy structure is assumed. In support of our framework, the IMM-based policy trained without prior information ($\Phi_{IMM}$) is shown to perform similarly to the ground-truth-based policies. Perhaps more interestingly, we note that $\Phi_{IMM}$ results in better imitation performance than the other IMM-based policy trained with prior information ($\Phi_{IMM}^*$). We believe this is because the robot state estimates $\hat{x}$, while seemingly superfluous, might have helped $\Phi_{IMM}$ better deal with mislabelled training data.

Finally, mission performance of the learned policies was evaluated in the perimeter defense scenario. In Tab. I actual mission performance obtained with the different policies are shown, while the error rates of the different policies with respect to the expert policy are shown in Fig. 7. In order to better understand the performance of the different policies, we separate attacks executed by a single intruder (SOLO), by both the intruders (DUO), and overall attacks (COMBO). Despite the limited performance in estimating two of the controllers as shown in Fig. 5, we observe that the imitator performance does not deviate significantly from the target one. This is explained by the limited impact of CYCLIC PURSUIT and LEADER-FOLLOWER on the actual mission performance.

|    | $F^*$ | $\Phi_{GT}$ | $\Phi_{IMM}$ | $\Phi_{GT}^*$ | $\Phi_{IMM}^*$ |
|----|-------|-------------|--------------|---------------|----------------|
| SOLO | 6.7   | 1.6         | 0.7          | 2.3           | 9.2            |
| DUO  | 13.5  | 8.0         | 3.5          | 7.6           | 16.1           |
| COMBO| 10.5  | 4.9         | 2.1          | 4.9           | 12.3           |

TABLE I: Mission performance results. Percentage of intruders’ successful attacks over total attacks.

Fig. 4: Screen shots from Robotarium simulation of the perimeter protection scenario. All robots are colored in orange and intruders are distinguishable from the yellow disc representing their capture area. The three blue circles, from dark to light, represent $B$, $r_r$, and $r_R$ respectively. Text in figure represents the behavior executed by the defenders.

VI. CONCLUSIONS

In this paper we addressed the problem of transferring a mission-specific control policy to an un-trained group of robots by exploiting repetitive observations of an expert system. Histories of observations of the expert’s state were first converted into sequences of known controllers. Then, histories of controllers and environmental states were used to train approximated policies. The work presented in this paper suggests at least two future directions of work. On one side, the fact that controllers’ estimation performance is directly correlated to the similarities between controllers’ outputs, motivate efforts towards the definition of a measure for the separability between controllers. Similarly, controllers that are executed by the expert system for short periods of time exacerbate the weakness of the estimation scheme. Nevertheless, as observed in this work, different controllers might have different importance on the final mission performance. For this reason, being able to evaluate from observations of a system the relative importance of its actions towards its performance is another interesting direction of work.
[42] T. R. Mehta and M. Egerstedt, “An optimal control approach to mode generation in hybrid systems,” *Nonlinear Analysis: Theory, Methods & Applications*, vol. 65, no. 5, pp. 963–983, 2006.

[43] P. Pierpaoli, T. T. Doan, J. Romberg, and M. Egerstedt, “A reinforcement learning framework for sequencing multi-robot behaviors,” *arXiv preprint arXiv:1909.05731*, 2019.

[44] H. A. Blom and Y. Bar-Shalom, “The interacting multiple model algorithm for systems with markovian switching coefficients,” *IEEE transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, 1988.

[45] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, “Interacting multiple model methods in target tracking: a survey,” *IEEE Transactions on aerospace and electronic systems*, vol. 34, no. 1, pp. 103–123, 1998.

[46] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[47] D. Shishika and V. Kumar, “Local-game decomposition for multiplayer perimeter-defense problem,” in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 2093–2100.

[48] 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 *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1699–1706.