Training Networked Agents to Perform Herding Tasks using Reinforcement Learning

Jacob Kimball
Department of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30313
jacob.kimball@gatech.edu

Jonathan Zia
Department of Electrical and Computer Engineering
Georgia Insitute of Technology
Atlanta, GA 30313
zia@gatech.edu

August 30, 2020

Abstract

This work considers the problem of learning optimal strategies to control complex multi-species interactions. Using reinforcement learning, a single agent was trained to simulate herding behavior on a randomized flock of multiple agents in an enclosed environment. The resulting model was then tested in the Robotarium, a freely accessible swarm robotics platform located at the Georgia Institute of Technology. Even with the additional restrictions imposed by the true robot dynamics and environment, the single agent was able to complete the full herding task.

1 Introduction

Advances in wireless technology at the beginning of the century allowed for the practical testing and implementation of network control strategies, leading to a surge in interest in the design and utilization of different network topologies. Concurrently, there was also increased interest in modeling biological networks as outlined in [1]. Implementing these biological models in real-world robotic systems can allow for increased understanding behind complex behaviors, and also produce new engineering solutions for analogous problems. The accurate implementation of biological behavior is challenging due to viable constraints such as homeostasis, switch-like responses, and adaptation [2]. As discussed in [3], reinforcement learning can be used to train agents to imitate complex biological constraints within a static environment.

For a single species, a basic model of flocking behavior can be contrived through the use of a δ-disk proximity consensus protocol [4] combined with a leader-follower protocol [5]. The δ-disk proximity consensus protocol differs from the standard rendezvous problem in which a set of connected agents converge on the centroid of their initial positions by enforcing limits on the minimum distance between adjacent agents. δ-disks can be defined by energy functions such that the velocity between adjacent agents is proportional to the distance between agents, and the minimum energy is found when they are separated by a distance δ. Thus, homeostasis occurs when all directly-connected agents are separated by a distance δ according to the network topology. Leader-follower networks implement the simple consensus protocol but also control the movement of a connected "leader" agent to influence continual convergence on a potentially moving target rather than the static centroid.

An example of complex multi-species behavior is exhibited in the predator-prey relationship model. In this model, similar to the leader-follower protocol, the velocity of the predator adjusts based on the current location and velocity of the prey as seen in [6].

2 Main Arguments

2.1 General Framework

Herding tasks may be formulated as a special case of leader-follower control systems. Generally, the goal of herding is to cause a large number of follower agents to converge to a desired state — typically defined by spatial coordinates and heading — using a relatively small number of leader agents. In biological systems, for example, this convergence is achieved by strategic movement of one or more predators, during which prey attempt to distance themselves from the predator while avoiding separation from their flock.

More formally, we consider a system of L leader agents at time t with state $\boldsymbol{\ell}_t \in \mathscr{L}$ and a system of F follower agents with state $\boldsymbol{f}_t \in \mathscr{F}$, where $\mathscr{L}$ and $\mathscr{F}$ are the sets of all possible leader and follower states respectively. We further define the action of each leader agent at time t as $\boldsymbol{a}_t \in \mathscr{A}_t$, where $\mathscr{A}_t$ is the action space of $\mathscr{L}$ (or, the set of possible states of the leader agents at time $t+1$). The goal of herding behavior is therefore for leaders to follow a policy function $\pi : (\mathscr{L}, \mathscr{F}) \mapsto \mathscr{A}_t$ such that $\boldsymbol{f}_{t_0+T} \in \mathscr{F}^\star$, where $\mathscr{F}^\star \subseteq \mathscr{F}$ is the set of desired follower states for a trial beginning at time t_0 and ending after T timesteps.

It should first be noted that in the domain of networked control, the dynamics of each agent are determined by the state of the agents in its visible range, also called *neighbors* of the agent. To model this behavior, we consider each agent as a vertex in the graph $G_t = (\boldsymbol{V}, \boldsymbol{E}_t)$ with vertices $\boldsymbol{V}$ and edges $\boldsymbol{E}_t \subseteq \mathscr{E}$, where $\mathscr{E}$ is the set of all possible edges between vertices in $\boldsymbol{V}$. In this case, edges are formed between neighboring agents; notably, edges may have directionality when observation is not mutual between neighbors. For convenience, we express $\boldsymbol{V} = \boldsymbol{V}_L \cup \boldsymbol{V}_F$ as the union of nodes representing leader and follower agents respectively. Since movement of agents in spatial dimensions may cause changes in graph connectivity [4], the edges in the graph — and therefore the graph itself — are expressed as a function of time. We further suppose that the state transition $\boldsymbol{f}_t \longrightarrow \boldsymbol{f}_{t+1}$ is governed by a mapping $\Psi : (\mathscr{L}, \mathscr{F} \mid \mathscr{G}) \mapsto \mathscr{F}$ which encodes the dynamics of the follower agents in response to their neighbors, as determined by a connectivity graph in the set of valid graphs $\mathscr{G}$.

As it stands, herding tasks are difficult to perform with traditional closed-loop control systems. This is because achieving convergence of the follower agents is a complex process that requires the leader agents to exhibit short-term behavior to achieve a long-term goal. To do so, leader agents must react to the movement of the followers and plan trajectories accordingly, though such trajectories may not yield short-term convergence of $\boldsymbol{f}_t$. To solve such problems, reinforcement learning has emerged as an effective and robust method [3].

In reinforcement learning, agents are typically trained to follow a policy which maximizes a value referred to as *reward*. We begin by defining a global reward function of the system state at time t as

$$R_\Psi(\boldsymbol{\ell}_t, \boldsymbol{f}_t, G_t) = \frac{1}{F} \sum_{j=1}^{F} R_\Psi^{(j)}\left(\boldsymbol{\ell}_t, \boldsymbol{f}_t, G_t\right). \tag{1}$$

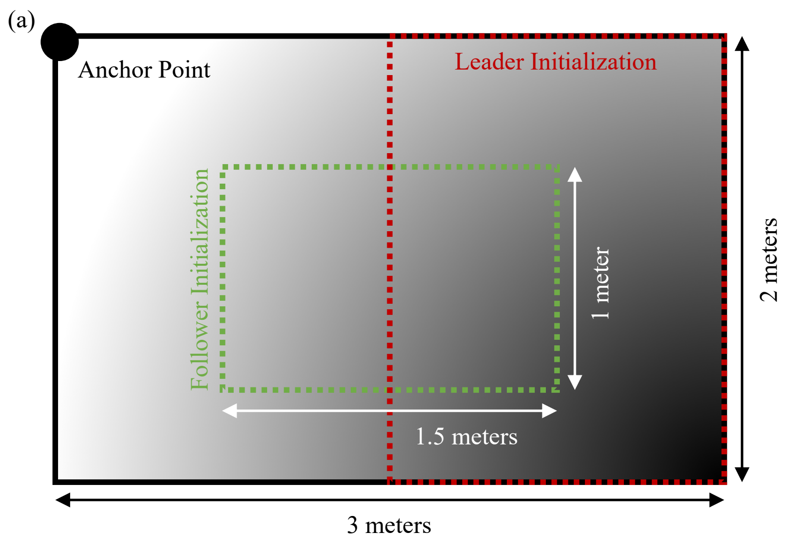
where $R_\Psi^{(j)}$ is the a local reward function for the j^{th} follower agent, which depends on the connectivity graph of the system and its underlying dynamics, as will be shown. From here, it is easy to see that developing a policy which selects actions based on maximizing reward at each step would be akin to a traditional gradient ascent method – however, as aforementioned, the emergence of complex behavior required for tasks such as herding may not be feasible with such methods. To develop a more holistic view of the state of the system at time t, we therefore define the *quality* of the system as

$$Q_\Psi(\boldsymbol{\ell}_t, \boldsymbol{f}_t, G_t \mid \pi) = \mathbb{E}\left[\sum_{t'=t}^{t+\lambda} R_\Psi(\boldsymbol{\ell}_t, \boldsymbol{f}_t, G_t) \mid \pi\right], \tag{2}$$

or, the expected cumulative reward over $\lambda \leq T$ timesteps where $\boldsymbol{\ell}_t$ at each step is determined by some policy π and $\boldsymbol{f}_t$ is determined by the mapping Ψ. By tailoring R such that the expected reward is higher when the system exhibits the desired behavior (in this case, when $\boldsymbol{f}_t \longrightarrow \boldsymbol{f}^\star \in \mathscr{F}^\star$) the quality function reflects the likelihood that the defined objective may be achieved from the current state. Furthermore, by accumulating reward over λ steps rather than a single step, the quality function reflects the longer-term potential of the current system state rather than simply the local gradient of the reward function. While the reward function is well-defined, however, the quality function must be estimated empirically by learning an approximation $\hat{Q}_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$ such that

$$\boldsymbol{\theta} = \underset{\hat{\boldsymbol{\theta}}}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} \left(\hat{Q}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\ell}_n, \boldsymbol{f}_n, G_n) - \tilde{Q}_\Psi(\boldsymbol{\ell}_n, \boldsymbol{f}_n, G_n \mid \pi)\right)^2 \tag{3}$$

where $\hat{Q}_{\hat{\boldsymbol{\theta}}}$ is the estimated quality and $\tilde{Q}_\Psi$ is the observed quality for each of N samples observed during simulation. Note that the parameterization $\boldsymbol{\theta}$ encodes the policy π and dynamics Ψ of the simulation, though these values are not explicit inputs of the function $\hat{Q}_{\boldsymbol{\theta}}$. Though out of scope of the current work, it has been shown that iterative performance of this process guarantees convergence of the estimated quality function to the optimal value function for the selected policy [7]. This method of defining and learning the quality of system states is known as *temporal difference* learning; in the case where one looks λ steps ahead, this is abbreviated to *TD-λ*.



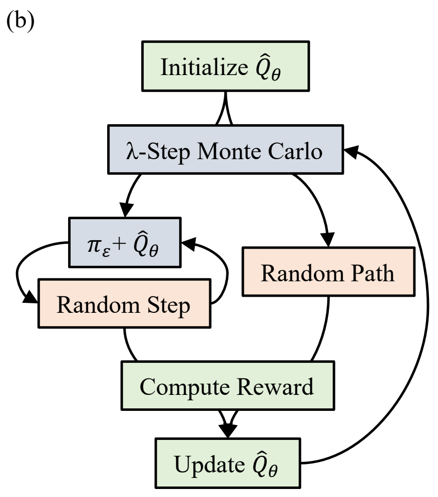


Figure 1: (a) Setup of simulation environment. Dashed rectangles indicate regions of random initialization of the agents. Shading illustrates the gradient of the cost function, with darker shading indicating a higher cost. (b) Box diagram of experimental procedure. Quality function updates are indicated in green; Monte Carlo simulation steps are shown in blue (exploitation) and orange (exploration). Detailed explanations in text.

Generally, the concept of reward in reinforcement learning is tied to the concept of cost; namely, positive reward is achieved when a sequence of actions results in lowering the cost of some objective function; conversely, a sequence of actions that increases cost over λ steps yields a negative reward. In this work, we define the cost function $C : (\mathscr{F}, \mathscr{F}^\star) \mapsto \mathbb{R}$ as a measure of the distance between the state $\boldsymbol{f}_t$ of the follower nodes and desired state $\boldsymbol{f}^\star \in \mathscr{F}^\star$. In this manner, the local reward — for a single agent and timestep — may be defined as

$$\begin{aligned} R_\Psi^{(j)}\left(\boldsymbol{\ell}_t, \boldsymbol{f}_t, G_t\right) &= C\left(\boldsymbol{f}_t^{(j)}, \boldsymbol{f}^\star\right) - C\left(\boldsymbol{f}_{t+1}^{(j)}, \boldsymbol{f}^\star\right) \\ &= C\left(\boldsymbol{f}_t^{(j)}, \boldsymbol{f}^\star\right) - C\left(\Psi^{(j)}\left(\boldsymbol{\ell}_t, \boldsymbol{f}_t \mid G_t\right), \boldsymbol{f}^\star\right) \end{aligned} \tag{4}$$

where $\boldsymbol{f}_t^{(j)}$ is the state of the j^{th} follower agent at time t and the function $\Psi^{(j)}$ is the output of the mapping Ψ for agent j. Thus, the local reward for each agent is dependent on both the connectivity graph of the agents and the dynamics governing the system of connected agents.

As shown in Equation 2, the definition of quality depends on selection of a policy π governing the actions of the leader agents. While π and Q may be learned in parallel, there exist several simpler ad-hoc methods for policy selection. Among these, the most common is the greedy policy. Greedy policies select actions via

$$\boldsymbol{a}_t = \underset{\boldsymbol{a}}{\text{argmax}}\ \hat{Q}_{\boldsymbol{\theta}}(\boldsymbol{a}, \boldsymbol{f}_t, G_t \mid \pi_G),\ \boldsymbol{a} \in \mathscr{A}_t \tag{5}$$

such that the selected action maximizes expected reward from following the greedy policy π_G at each step [8]. In this work, a greedy policy was used during testing, while a variant known as the ϵ-greedy policy (π_ϵ) was used during the training of $\hat{Q}_{\boldsymbol{\theta}}$. In this method, the probability of following the greedy policy at each timestep is (1 - ϵ) while the probability of selecting a random action from the set of valid actions $\mathscr{A}_t$ is ϵ. In this manner, exploitation of learned patterns is balanced with exploration of new patterns during training to combat aberrant convergence to local minima. In the sections that follow, these concepts will be synthesized to enable herding of follower agents by governing the actions of leader agents with a learned quality estimation $\hat{Q}_{\boldsymbol{\theta}}$ coupled with a greedy policy.

2.2 Herding Formulation and Objective

The experimental methods in this work are illustrated in Figure 1. As shown in Figure 1(a), the simulation environment for the herding task was a two-dimensional rectangular surface. This environment was chosen to match the geometry of the testing environment of the Robotarium at the Georgia Institute of Technology where the proposed methods were ultimately tested. The goal of the herding task was to cause $F = 5$ follower agents to converge to the anchor point using a single leader agent ($L = 1$).

To do so, $\boldsymbol{\ell}_t^{(i)}$ and $\boldsymbol{f}_t^{(j)}$ were defined as the (x, y)-coordinates of the leader and follower agents as measured from the anchor point. Since the goal of this work was to cause the follower agents to converge to the anchor point, the cost function C for the j^{th} follower at time t was defined as

$$C\left(\boldsymbol{f}_t^{(j)}, \boldsymbol{f}^\star\right) = \|\boldsymbol{f}_t^{(j)} - \boldsymbol{f}^\star\|_2^2, \tag{6}$$

or the squared ℓ_2-norm distance between the follower agent and the anchor point. As such, cost was decreased — and reward was increased — with respect to each follower when the system state at time t caused convergence of the follower to the anchor point at time $t+1$. Notably, no constraints were imposed on the leader agent in this paradigm.

To simplify the complexity of this task, the dynamics of the follower agents were dictated only by the leader agent. At each timepoint t, the graph G_t was defined as a δ-disk graph such that each agent had a visibility radius of $\delta = 1.5$ meters. The dynamics of the j^{th} follower agent was given by

$$\dot{\boldsymbol{f}}_t^{(j)} = \sum_{i=1}^{L} \mathbb{1}\left\{\|\boldsymbol{f}_t^{(j)} - \boldsymbol{\ell}_t^{(i)}\|_2 \leq \delta\right\} \frac{\left(\boldsymbol{f}_t^{(j)} - \boldsymbol{\ell}_t^{(i)}\right)}{\|\boldsymbol{f}_t^{(j)} - \boldsymbol{\ell}_t^{(i)}\|_2^2} \tag{7}$$

where $\mathbb{1}\{\cdot\}$ is an indicator function. This equation describes a δ-disk weighted consensus protocol where follower agents are repelled by the leader agent at a distance of less than 1.5 meters. The propensity of leader agents to affect follower agents is normalized by the squared distance between the agents such that the strength of this effect is diminished with increased distance. More formally, the dynamic mapping Ψ was therefore defined as

$$\Psi^{(j)}(\boldsymbol{\ell}_t, \boldsymbol{f}_t \mid G_t) = \boldsymbol{f}_t^{(j)} + \dot{\boldsymbol{f}}_t^{(j)}. \tag{8}$$

With these functions and variables defined, the following section describes training and testing of the quality function.

2.3 Training and Testing Protocols

In this work, the estimated quality function $\hat{Q}_{\boldsymbol{\theta}}$ was modeled as a neural network with $2F + 2L = 12$, a single hidden layer of 10 units, and an output layer with a single unit. The hidden layers had hyperbolic tangent activation functions while the output unit had a strictly linear activation function. The model inputs represented the (x, y)-coordinates of the follower and leader agents while the output was trained to estimate the expected reward based on these positions.

While the dynamics of the follower agents were dictated by the mapping Ψ, the state transition $\boldsymbol{\ell}_t \longrightarrow \boldsymbol{\ell}_{t+1}$ was determined by Equation 5. To simplify analysis, the velocity of the leader was fixed to $u_\ell = 0.1$ meters per second while the heading $\phi \in \Phi$ was chosen by the policy. Namely, the possible headings were defined as angular offsets from the x-axis; valid offsets were given by $\Phi = [0, \pi/4, \pi/2, 3\pi/4, \pi, 5\pi/4, 3\pi/2, 7\pi/4]$ radians. Therefore, at each time t, the action space $\mathscr{A}_t$ contained the actions

$$\boldsymbol{a}_\phi = \boldsymbol{\ell}_t + \begin{bmatrix} u_\ell \cos\phi \\ u_\ell \sin\phi \end{bmatrix} \forall \phi \in \Phi. \tag{9}$$

The action of the leader agent at each timestep was therefore chosen from the set of available actions which maximized expected reward, as shown in Equation 5.

For eventual testing in the Robotarium at the Georgia Institute of Technology, a simulation environment was created for model training and testing as shown in Figure 1(a). The simulation surface was a 3 meter by 2 meter rectangle; agents in this space were represented by the (x, y)-coordinates and were treated as point masses without restricted movement to simplify analysis. To reflect the simulation environment of the Robotarium, however, the leader agent velocity u_ℓ was selected as half the maximum velocity of Robotarium agents; the velocity of the follower agents was further thresholded by $0.5 \times u_\ell$ both for Robotarium implementation and simulation to allow the leader agent enough velocity to encircle the follower agents. Finally, boundaries were enforced for leader and follower agents by setting the x- and y-component of the velocity vector to 0 when following that trajectory would cause the agent to cross the horizontal and vertical boundaries respectively.

The training procedure for the quality function is shown in Figure 1(b). After initializing the neural network $\hat{Q}_{\boldsymbol{\theta}}$ with random weights, training samples were generated by performing Monte Carlo (MC) simulations of herding tasks, with each MC simulation representing one training sample. To begin each simulation, the follower agents were initialized within the green region in Figure 1(a) with uniform probability; similarly, the leader agent was initialized within the red region with uniform probability.

At the beginning of the simulation, it was decided whether actions would be chosen using the ϵ-greedy policy π_ϵ or whether $\boldsymbol{a}_t$ would be chosen randomly from $\mathscr{A}_t$ at each step. To balance exploration with exploitation, the probability

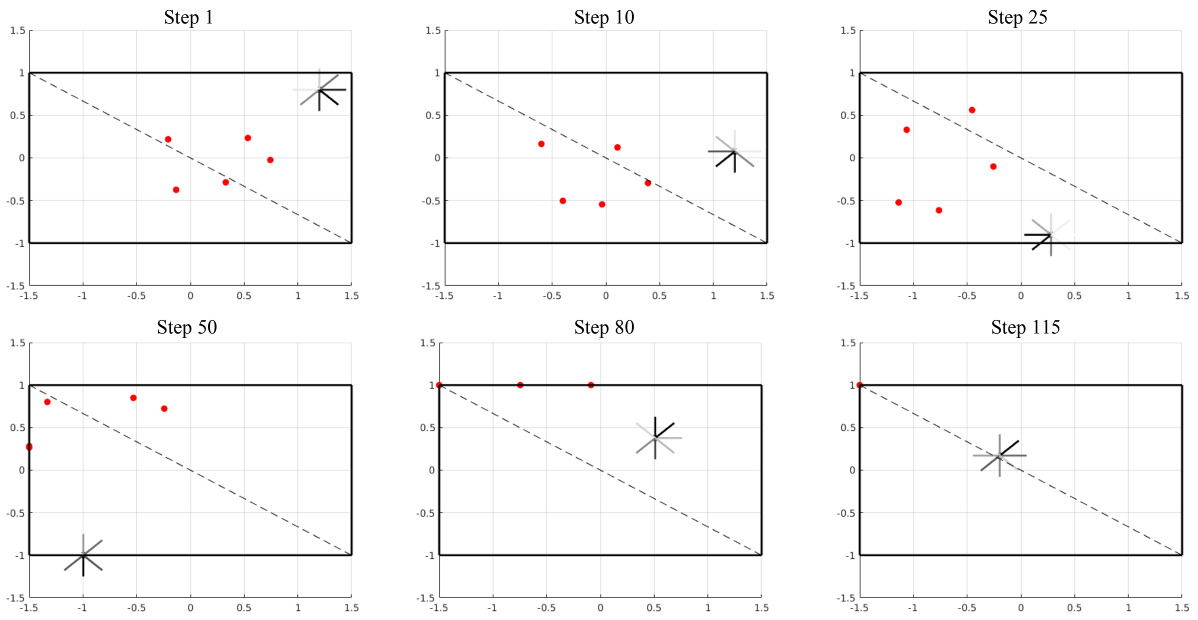


Figure 2: Testing simulation using trained quality function $\hat{Q}_{\boldsymbol{\theta}}$ with greedy policy π_G. Follower agents are shown in red and leader agent is shown in black. Lines originating from leader agent indicate each of the possible headings; the line color indicates the estimated reward for each heading, with darker shading indicating higher estimated reward.

of choosing the random path was set to 25%. Furthermore, if π_ϵ was selected at the beginning of the simulation, there was a 25% chance of selecting $\boldsymbol{a}_t$ randomly at each step and a 75% chance of following π_ϵ. In this manner, exploration and exploitation were further balanced.

Each MC simulation was run for $T = 100$ steps with $\lambda = 10$. For each timestep $t \in [1, T]$, the position of each agent was collected to form the 12-unit input vector to $\hat{Q}_{\boldsymbol{\theta}}$. After λ steps of the simulation, the accumulated reward in the interval $[t, t + \lambda]$ was recorded as the training label to the input vector, computed at each step using Equation 1. In cases near the end of each simulation where $t + \lambda > T$, the accumulated reward was calculated over the interval $[t, T]$. The training samples generated during each simulation were appended in the matrix X and the corresponding labels for each sample were appended to the vector $\boldsymbol{L}$. After each MC simulation, the weights of $\hat{Q}_{\boldsymbol{\theta}}$ were updated using backpropagation to fit the samples X to the labels L. This process was repeated for 4000 training iterations.

After training, model performance was visualized by performing MC simulations in the same manner as above, instead using the greedy policy π_G with the trained quality function. The output for an iteration of testing is shown in Figure 2, with the full video available for viewing here.

2.4 Robotarium Implementation

Before testing the trained model in the Robotarium, a simulation was conducted using the Robotarium MATLAB library. In the simulation protocol, the agents were first instructed to proceed to their starting positions at waypoints defined in the highlighted regions of Figure 1(a). From these starting positions, the trained model was then used with the greedy policy π_G to determine the action of the leader at each timestep. As before, the leader velocity was fixed to $u_\ell = 0.1$ meters per second; follower dynamics were provided via Equation 7 and the velocity of the followers was again thresholded by $0.5 \times u_\ell$. At each timestep, the graph G_t was re-computed as a δ-disk where $\delta = 1.5$ meters. Finally, the boundaries of the simulation environment and the safety radius of each agent was enforced using boundary certificates. The results of this simulation are shown in Figure 3, with the full video available for viewing here.

2.5 Results and Discussion

As shown in Figures 2 and 3, a leader agent was trained using reinforcement learning to herd follower agents to an anchor point. In doing so, complex behaviors emerged within 4000 training iterations, namely the strategies of (1) remaining outside the complex hull defined by the positions of the follower agents; (2) partitioning the herd into separate groups; (3) utilizing the boundaries of the environment for assistance; and (4) awareness of the desired anchor point.

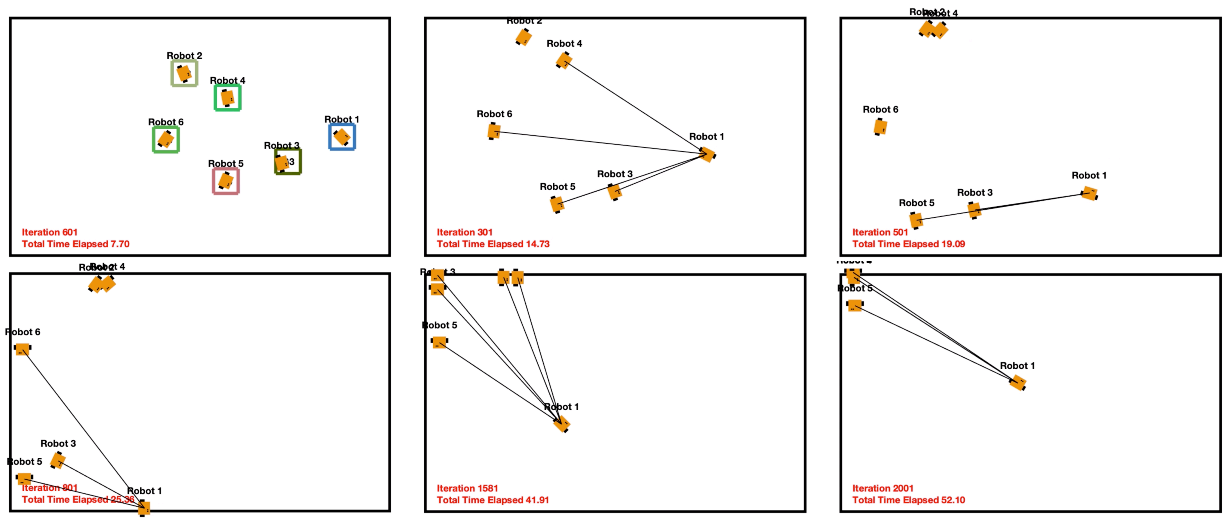


Figure 3: Testing trained model in Robotarium MATLAB simulation. Initial waypoints (colored squares) and δ-disk graph connections (black lines) are overlaid.

While successful in this basic task, this work elucidates several opportunities for future work. Selecting arbitrary anchor points that do not lie on the edges of the environment may enable the emergence of more complex, realistic herding behavior and provide a system that is more generalizable. This may be accomplished by increasing λ during temporal difference learning and imposing a penalty for agents that are near the edges of the environment. Furthermore, while this work only utilized a single leader agent, future work should explore the use of multiple follower agents, which may operate using a de-centralized control system. Finally, from the machine learning standpoint, there is much room for optimization on the current method. Future work may seek to perform more robust hyperparameter optimization (rather than the heuristic method performed in this work) and implement more robust training protocols that weigh recent iterations more heavily than early training iterations in which the value function was less mature.

A key assumption in this work is that the leader agent has complete visibility of the simulation environment. In practical scenarios, this may be a weak assumption, as follower agents which lie outside the leader's δ-disk proximity may not be incorporated into the leader's decision-making. Though agents in this work did not affect neighbors outside their δ-disk proximity, this assumption remains an open limitation of this method.

The simulation shown in Figure 3 was implemented in the Robotarium at the Georgia Institute of Technology; the full video of this simulation is available for viewing here. As shown in the video, the simulated behavior was partially-emulated in the physical system of robots, however barrier certificates which were used to prevent collision of the robots with each other or with the boundaries of the environment prevented exploitation of the environment boundaries for the herding task. However, the leader agent nevertheless exhibited complex herding behavior as shown in the simulation; to fully-enable such implementation in the future, barrier certificates should be incorporated into the simulation environment to allow quantification of such factors by the learned value function.

2.6 MATLAB Implementation

The code for this data project as well as instructions for use are available here.

3 Conclusion

This work has shown that simple network control principles coupled with reinforcement learning can be used to emulate complex behaviors exhibited during herding tasks. Specifically, a single leader agent was shown to drive five randomly-placed follower agents into a designated corner of the environment using greedy adherence to an experimentally-trained value function. Ultimately, this work demonstrates the potential of computational methods that lie at the nexus of network control and machine learning.

References

[1] R. Albert, J. Baillieul, and A. E. Motter, "Introduction to the special issue on approaches to control biological and biologically inspired networks," *IEEE Trans. Control Netw. Syst.*, vol. 5(2), pp. 690–693, 2018. arXiv:1807.00038v1.

[2] H. Steel and A. Papachristodoulou, "Design constraints for biological systems that achieve adaptation and disturbance rejection," *IEEE Trans. Control Netw. Syst.*, vol. 5(2), 2018. 10.1109/TCNS.2018.2790039.

[3] E. Neftci and B. Averbeck, "Reinforcement learning in artificial and biological systems," *Nature Machine Intelligence*, March 2019. 10.1038/s42256-019-0025-4.

[4] C. He, Z. Feng, and Z. Ren, "A flocking algorithm for multi-agent systems with connectivity preservation under hybrid metric-topological interactions," *PLoS ONE*, vol. 13(2):e0192987, 2018. [Online]. Available: https://doi.org/10.1371/journal.pone.0192987

[5] D. Gu and Z. Want, "Leader–follower flocking: Algorithms and experiments," *IEEE Transactions on Control Systems Technology*, vol. 17(5), pp. 1211–1219, 2009. 10.1109/TCST.2008.2009461.

[6] F. Tang, B. Si, and D. Ji, "A prey-predator model for efficient robot tracking," *IEEE International Conference on Robotics and Automation*, 2017. 10.1109/ICRA.2017.7989409.

[7] P. Dayan, "The convergence of td(λ) for general λ," *Machine Learning*, vol. 8, pp. 341–362, 1992.

[8] Y. Efroni, G. Dalal, B. Scherrer, and S. Mannor, "Multiple-step greedy policies in online and approximate reinforcement learning," *32nd Conference on Neural Information Processing Systems*, 2018c. arXiv:1805.07956.