Learning to Herd Agents Amongst Obstacles: Training Robust Shepherding Behaviors Using Deep Reinforcement Learning

Jixuan Zhi and Jyh-Ming Lien, *Member, IEEE*

***Abstract*—Robotic shepherding problem considers the control and navigation of a group of coherent agents (e.g., a flock of bird or a fleet of drones) through the motion of an external robot, called shepherd. Machine learning based methods have successfully solved this problem in an environment with no obstacles. Rule-based methods, on the other hand, can handle more complex scenarios in which environments are cluttered with obstacles and allow multiple shepherds to work collaboratively. However, these rule-based methods are fragile due to the difficulty in defining a comprehensive set of behavioral rules. To overcome these limitations, we propose the first known learning-based method that can herd agents amongst obstacles. By using deep reinforcement learning techniques combined with the probabilistic roadmaps, we train a shepherding model using noisy but controlled environmental and behavioral parameters. Our experimental results show that the trained shepherding controller is robust, namely, it is insensitive to the uncertainties originated from either the group behavioral models or from environments with a small of path homotopy classes. Consequently, the proposed method has a higher success rate, shorter completion time and path length than the rule-based behavioral methods have. These advantages are particularly prominent in more challenging scenarios involving more difficult groups and strenuous passages.**

***Index Terms*—Motion and path planning, reinforcement learning, task and motion planning.**

I. Introduction

THE robotic shepherding problem, inspired by sheepdogs and sheep, can be defined as one or more shepherd robots try to guide a team of agents from an initial location to a goal location [1]. Shepherding is applicable to various fields in the real world, such as using robots to herd animals [2], civil crowd control [3], preventing bird strikes near airports [4], facilitating communication between the unmanned ground vehicles and unmanned aerial vehicles [5].

Algorithms to address the shepherding problem can be classified into two main categories: rule-based and learning-based algorithms. Rule-based algorithms define a set of rules that compute the dynamics of the model as a function of the system status. These methods are easy to implement and can work well with a large number of the sheep agents. It has also been shown that multiple shepherd agents can work collaboratively to herd a large group effectively [6]. However, these rule-based methods are fragile due to the difficulty in defining a comprehensive set of rules that can handle all possible cases. Some commonly seen difficulties include turning the coherent group in a different direction or removing those trapped agents from the corners. On the contrary, learning-based algorithms do not need the knowledge about the model and have been shown to be resilient to imperfect group behavior parameters of the sheep. Unfortunately, these learned models have only been trained and tested in simple blank environments. Navigating the sheep among the obstacles has not been considered in these learning-based methods. More specifically, a shepherd trained only in empty environments has no internal representation about the effect of obstacles on the sheep. Therefore, it has never had a chance to learn how to make the group of sheep turn or deform around or near the local geometry of the obstacles.

These limitations are the predominant reasons that shepherding robot cannot robustly herd agents amongst obstacles under uncertain behavioral and environmental models. In this letter, we study these issues in the framework of deep reinforcement learning.

Our main contributions include: (1) a reinforcement learning framework that trains the shepherding model in the environments populated with a small group of coherent agents and obstacles influenced by controlled noise (Sections IV&V), and (2) comprehensive experiments demonstrating the learned model that can control the shepherd's movement with higher probability, shorter completion time and path length in herding the sheep comparing to the rule-based methods (Section VI).

II. Related Work

Researchers proposed various abstract models of shepherding behaviors in swarm robotics. For instance, Vo *et al.* [7] proposed a behavior-based method which selects intermediate goals on the roadmap of the workspace medial axis. Harrison *et al.* [8] represents the controlled group as a deformable shape. Strömbom *et al.* [9] proposed a heuristic method and described two behaviors: driving and collecting. Driving means steering the flock to the target, and collecting means steering the outlier back

Manuscript received October 15, 2020; accepted February 20, 2021. Date of publication March 25, 2021; date of current version April 8, 2021. This letter was recommended for publication by Associate Editor S. J. Guy and Editor N. Amato upon evaluation of the reviewers' comments. *(Corresponding author: Jixuan Zhi.)*

The authors are with the Department of Computer Science, George Mason University, 4400 University Drive MSN 4A5, Fairfax, VA 22030 USA (e-mail: jzhi@gmu.edu; jmlien@cs.gmu.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3068955, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3068955

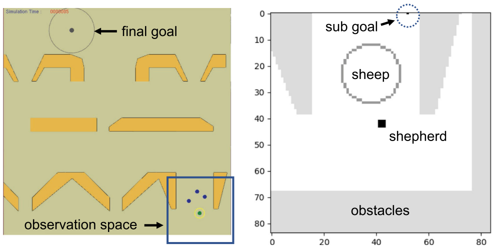

Fig. 1. The global view of the workspace (left) and the shepherd's observation space (right) in a local view.

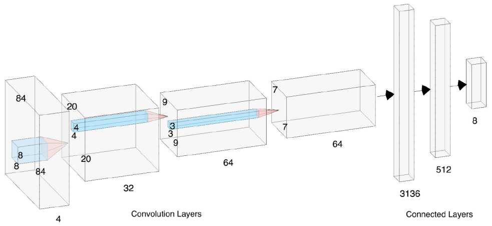

Fig. 2. Deep reinforcement learning architecture used in this work. The first three layers are convolution layers with kernel size shown in blue.

to the flock. Based on [9], Fujioka and Hayashi [10] introduced V-formation control, in which shepherd moves along the V-shape arc enclosing the flock.

Most shepherding models employ one shepherd to control the flock. However, there are some researchers developed multiple shepherds to guide the flock [6], [11]. Recently, Lee and Kim [12] developed a method without centralized control. In their work, the shepherds only attempt to steer the nearest sheep to the target region without considering other members in the flock, with such behavior rules, multiple shepherds can have an arc formation to control the flock. Unfortunately, these rule-based shepherding behaviors are fragile due to the difficulty in defining rules to handle all possible cases.

In recent years, machine learning approaches have been applied to solve shepherding problems. Baumann developed a reinforcement learning method [13] called Growing Neural Gas Q-Learning to herd one sheep using a single shepherd. He used abstract states to generalize the neighboring and similar states which share the same behavior. Apply Q-learning [14] with such small state space, the algorithm works well with linear computational complexity. Nguyen *et al.* [15] applied inverse reinforcement learning and deep reinforcement learning in UAV shepherding. They trained different models for collecting and driving, then aggregated these models to herd a swarm of ground vehicles.

None of these learning methods considered obstacles.

III. Preliminaries

We assume that there is one shepherd and multiple coherent agents representing the sheep; and the shepherd has a local view that means shepherd either can know the positions of sheep and goal inside its view or the directions of sheep and goal outside its view. The mission of the shepherd is to guide the sheep to a goal area represented as a circle (See Fig. 1).

A. Group Behavior Model

Reynolds' Boids [16] is used to model the dynamics of the sheep. The behavior of each sheep is influenced by the location and velocity of the nearby sheep, which in turn define three control forces: separation, alignment, and cohesion. For each sheep, separation force repulses the sheep from the nearby sheep, alignment force steers the sheep in the average direction of the nearby sheep and cohesion force moves the sheep toward the average position of the nearby sheep. In addition, a *fear* force makes the sheep run away from the shepherd when the shepherd appears in the local neighborhood. The local neighborhood is defined by the distance and its field of view. Additional forces are added for obstacle avoidance and damping. In each simulation step, these forces are linearly combined and integrated to update the acceleration, velocity and position of each agents.

Changing the coefficients of the weighted sum of these forces allows us to model different group behaviors. However, tuning these parameter in the Boids model to simulate a particular group of animals (e.g., a flock of birds found in Dulles Airport in Washington DC, USA) is extremely challenging and often impossible. Therefore, the controller must operate under the assumption that the behavior model of the group is inaccurate.

B. Deep Q-Learning

A regular reinforcement learning problem considers four components: (1) a current state s in Observation space, (2) an action a chosen by the agent in Action space, (3) a transition model $P(s'|s,a)$ for next state and (4) a reward r given by the action and state. The optimized policy in each state is found by maximizing the cumulative reward over time. At time t, we define the cumulative reward as this quantity:

$$R_t = r_{t+1} + \gamma r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

where γ is the discount factor.

Deep Reinforcement Learning [17] started a new era in reinforcement learning development. A deep neural network is used to represent both state and action onto a value, namely $Q(s,a)$ and has the potential to handle large configuration space in shepherding problem. Deep Q-Learning uses *replay buffer* [17] of the past experiences and samples training data from it. Each data in the replay buffer includes the current state, the action, the next state, the reward, and whether the goal is reached or not. Schaul *et al.* [18] proposed priority replay buffer to improve the sampling efficiency. Van Hasselt *et al.* [19] developed the Double Q-learning technique which can reduce the overestimation of Q-value. We apply priority replay buffer and double Q-learning techniques in this work.

IV. Method

A. Model Architecture

Our basic framework (Fig. 2) follows the model proposed in [17]. It is a deep neural network architecture that has three convolution layers and two fully connected layers and there is a rectifier nonlinearity (RELU) unit for each hidden layer.

B. Observation Space

We consider a partially-observable continuous world, in which the shepherd robot can only observe the state of the world in a field of view centered around itself. In practice, 84×84 pixels local frame is the input. To push our environment back to the Markov Decision Process domain, we use the *frame stack technique* [20] to maintain several frames from the past and use them as the observation at each state.

Representing the Sheep: A critical question in training the shepherding behavior is how the group of sheep should be represented. We choose to represent the group as a bounding circle as shown in Fig. 1 and define the position of the sheep as the center of the group. Our experiments showed that the circle representation allows faster convergence in training comparing to representing individual member as pixels [21].

Goal and Waypoints: In each 84×84 frame, we use 3×3 pixels in the center of frame to represent the shepherd, one pixel to represent the goal, 3×3 pixels to represent a single sheep. If there are multiple sheep, we can use a circle to represent them, the center of a circle is called “sheep center” and the radius is the maximum distance between sheep and the sheep center. With center and radius, we draw a hollow circle. For boundaries and obstacles in the environment, we draw solid polygons. In the local frame, we use different pixel values for different objects. See Fig. 1 for an example.

The goal (or waypoint) and the sheep are sometime outside of shepherd local field of view, but these information is essential for the shepherd. For the goal, we connect the shepherd position and the goal position as a line segment, and get the intersection of the segment and the boundary of the local view frame, then we mark the intersection as the goal inside the local frame. For the sheep, if the circle is partially inside the local frame, we draw that part of the circle in the local frame. If the circle is completely outside the local frame, we connect the shepherd and sheep center as a line segment, mark the intersection as the sheep in the boundary.

To handle environments populated with obstacle, a path planner will inquire the probabilistic roadmaps (PRM) [22] and find a steering path connecting the sheep (center) to the goal. It is important to note that such a roadmap is constructed to provide the shepherd a high-level idea of where to steer the sheep to. The roadmap, however, does not instruct how the shepherd should place itself in order to move the sheep around the corner of an obstacle or squeeze the sheep into a narrow corridor. These critical behaviors are needed for successful navigation and are learned in our work.

C. Action Space

The shepherd robot takes semi-discrete actions as illustrated in Fig. 3. In every step, one of eight directions is selected by the neural network and a force along that direction with fixed magnitude is applied to the shepherd. In addition, to make the action appear to be continuous, we add a clipped Gaussian noise to the direction of the force. The deviation of each direction angle is between $-22.5°$ and $+22.5°$, then we can cover all the entire circle.

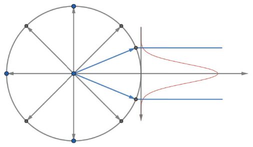

Fig. 3. Discrete actions.

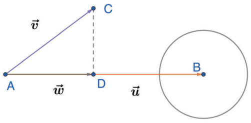

Fig. 4. Projection reward $\vec{w}$.

D. Reward Structure

The reward function consists of three parts: moving reward, violation reward and goal reaching reward.

1) Moving Reward: Moving reward includes two parts: projection reward and path reward.

Projection Reward: To ensure that the sheep is moving in the desired direction and reach the goal as fast as possible, we introduce the projection reward. Let the current sheep position be A, new position be C and the goal position be B in the goal region represented by the circle; See Fig. 4. We further let $\vec{u} = B - A$ be the vector from current sheep position to goal position, and let $\vec{v} = C - A$ be the vector form current sheep position to new sheep position. We compute the projected vector $\vec{w}$ of vector $\vec{v}$ onto vector $\vec{u}$. We determine the reward based on the direction and length of the vector $\vec{w}$, i.e., the reward has the same sign as the dot product of $\vec{v}$ and $\vec{u}$. The reward is further scaled by a factor. The scale factor for negative reward must be larger than that of a positive reward (which are 4 and 2, resp.) to avoid local optimal.

To encourage exploration in searching for optimal controller, we penalize the action that causes the sheep for staying still or moving too little (< 1 pixel) or circling around the goal.

Path Reward: To make the moving reward more accurate, we determine the path reward based on the geodesic distance. The reward function calculates the difference of the geodesic distances of two states. If the available path of the new state is less than the available path of the old state, the reward is positive, otherwise, the reward is negative. We also add a scale factor of the reward and a small threshold (1 pixel) to give a small negative reward as same as the projection reward which also can avoid circular motion of the sheep.

2) Violation Reward and Reaching Goal Reward: We consider two factors of violation reward. Firstly, for multiple sheep, we need to consider the cohesion of the group. We define the circle with its center and the radius in Section IV-B and penalize the group with a large negative reward if the radius is large. This threshold is equal to the view range of the sheep (25 pixels). We also define a termination threshold if the radius is larger than 30 pixels. Therefore, the shepherd can still be trained to control the bounding circle as small as possible. However, if the sheep are too scattered, then we terminate the training as the shepherd is unlikely to merge the sheep back into a single group using local behaviors. Secondly, we want the shepherd to have effective explorations around the group, so we penalize the shepherd with a large negative reward if the shepherd is far away from the sheep center. The threshold is two times of the view range of the sheep. The termination threshold for this type is 60 pixels.

If circle size or distance in these violation cases are larger than failure threshold or the simulation exceeds the maximum time step of the training episode, the simulation will terminate this episode and report its failure.

On the contrary, a large positive reward is given when all the sheep reach the goal region and the simulation ends the episode successfully.

V. Details in Training Shepherding Controller

In this section, we provide details on the training process. In particular, the controller is trained with random noise injected to the environmental and behavioral models. In real-life, there is always some noise, such as noise in sensing the locations of the sheep or the geometry, position or orientation of the obstacles, therefore the simulated noise is added and could make our model more robust.

For the training parameters, we use the ADAM optimizer [23] with mini batches of size 32, and use ϵ-greedy policy to explore the environments with ϵ changed linearly from 1.0 to 0.5 for one slope and from 0.5 to 0.02 for the other smaller slope over the first one million frames and fixed at 0.02 thereafter. The priority replay buffer size is 20 000, the discount factor is 0.99, and the learning rate is 0.0001. The double Q-learning technique is also applied. We use the frame-skipping technique to reduce training time. More precisely, the shepherd robot chooses actions on every $k = 5$th frame , and the chosen action is repeated in the intermediate (i.e., skipped) frames. The time steps of training and evaluation were determined by the complexity of the given scenario, we used the simple rule method to estimate the time step and determined the allowance.

To help shepherd navigate around the obstacles, we construct a roadmap using two variants of probabilistic motion planners that sample one-third of configurations uniformly [22] and the other two-thirds on the medial axis of the environments [7].

To study the robustness of the trained controller, we consider two types of obstacles in the environment: fixed obstacles and perturbed obstacles. The physical size of all simulated environments is $50 \times 50\ \mathrm{m}^2$ with 1 m $=$ 5 pixels.

1) Fixed Obstacles: We first consider the environments with predefined obstacles. In practice, the environment (such as in a farm or ranch) is usually fixed, and the sheep routinely moves toward one location for grazing and the other location for drinking. Therefore, we create a fixed environment as shown in Fig. 5. We set two positions and alternate between these two positions as the start and goal. Fig. 5 shows two trajectories in the opposite directions that successfully herd a group of three sheep in the filter environments. Since the challenges faced in herding the group in these two directions are quite different, we split the filter environment into filter-0 (Fig. 5 left) and filter-1 (Fig. 5 right).

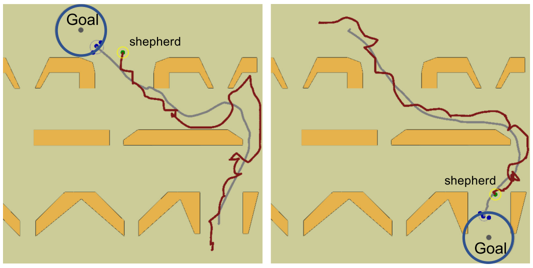

Fig. 5. The fixed obstacles env. (filter env.) with shepherd (red) & sheep paths (grey) generated by the proposed method.

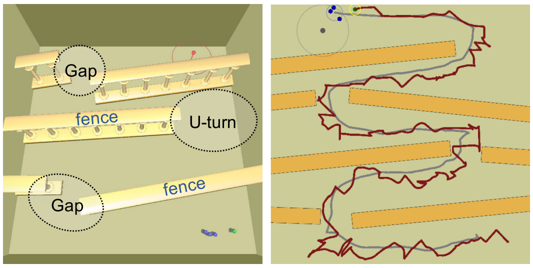

Fig. 6. We perturb the positions and orientations of the fences that form three (left) or four (right) U-turns and gaps during training.

With this fixed environment, we randomly change the group behavior parameters, namely separation, alignment, and cohesion from a clipped Gaussian distribution. To be more specific, the range of separation coefficient is between 0.5 and 1.5, the range of cohesion coefficient is between 3 and 7, the range of alignment coefficient is between 2 and 4, and fear coefficient is fixed on 15. All the coefficients scale linearly. The terminal time steps of each training episode is 6000. The average training time is about 15 hours.

1) Perturbed Obstacles: Next, we train the controller in environments with randomly perturbed obstacle. Fig. 6 shows two examples composed of fences which partition the spaces into long corridors with narrow gaps and sharp turns. To train the controller, we create three-layer obstacle environments (Fig. 6, left) and perturb the position and angle of each fence using a clipped Gaussian noise. The range of position noise is between -2.5 and $+2.5$ pixel, and the range of the initial angle is between -8° and 8°. For the gaps formed by two fences, the shorter fence may appear or disappear randomly thus changes the difficulty of the herding problem. We also randomly select the start from the top and the goal from bottom in the environment and vice versa.

Similar to the scenarios with fixed obstacles, group behavior coefficients are randomized, but the fear coefficient is now randomly selected between 13 and 17.

VI. Evaluating the Trained Controller

In this section, we highlight the effectiveness and robustness of the trained controller over the rule-based approaches. The

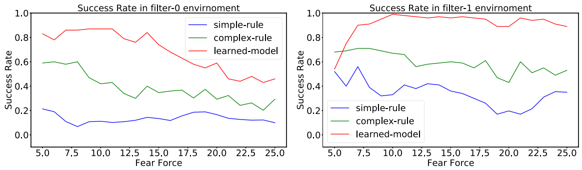

Fig. 7. Success rate in different levels of fear in the filter environment. High fear force models easily scared agents (e.g., geese) that are more difficult to control. The learned controller is trained with fear coefficient 15.

trained controller is tested with random noise injected to our environmental and behavioral models. All the experiments were run on a single desktop, equipped with a NVIDIA TITAN X (Pascal) and an Intel Xeon CPU 2.3Ghz and 32 Gb RAM.

We compare the trained controller with two rule-based methods, namely the simple-rule and complex-rule methods [1], in which manually created rules instruct how the shepherd should place itself to steer the sheep around the corner or squeeze them into a narrow corridor. Comparing to the simple-rule method, the complex-rule method uses a stop-turn steering policy [1], which allows shepherd to stop the sheep and steering them toward a new direction. Both methods are capable of merging the separated sheep back to a single group. These rule-based methods remain fixed in each experiment.

A. *Robustness of Herding Controller in Fixed Environment*

In this experiment, we study the robustness of the trained controller trained in a fixed environment subject to the noise in the behavioral parameters. More specifically, we vary the *fear force* by changing the fear force coefficient between 5 and 25 while the separation and cohesion coefficients are fixed at 1 and 5, respectively. Large fear force models easily scared agents (e.g., geese) that are more difficult to control. Although we focus on *fear* in this work, we observe similar results by varying the separation and cohesion coefficients [21].

The robustness is measured using the success rate, completion time and path length. In these experiments, we collect data from 500 runs, including failed cases, from each method and the terminal time steps of each testing episode is 5000.

Success Rate: In Fig. 7, the proposed model maintains a higher success rate compared to the other methods even when the controlled group becomes more difficult to herd. Even when we train the controller using the fear force at value 15, the model shows a high success rate for a large ranges of fear coefficient from 5 to 25.

Completion Time: In addition to the success rate, the quality of the path is equally important. We measure the quality using the time needed to complete a herding task. Fig. 8 shows the completion time of the successful cases for different levels of fear force in the filter environments. In these figures, the learned model spends less time herding than the other two rule-based methods. Simultaneously, the standard deviation of completion time is also the smallest among all methods. The figures show that the learned model completed the herding task as soon as possible, and it is robust to maintain this quality.

Path Length: Path length is another criterion for measuring the controller quality. Fig. 9 shows path length obtained using different levels of fear force coefficient. From these figures, we find that the learned model moves significantly less compared to that controlled by the rule-based methods. This property is useful when considering the energy cost. We also note that the standard deviation of mean path length with the learned model is smaller than other rule-based algorithms. This indicates that the learned model is robust to maintain its efficiency even when the group becomes more difficult to control. Similar results are observed when the group coherence varies [21].

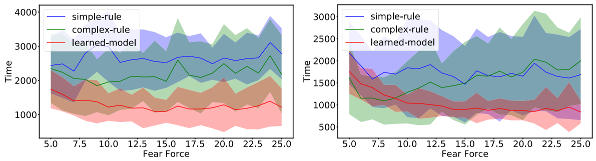

Fig. 8. Completion time (simulation steps) of the successful runs for different levels of fear coefficient in the filter-0 (left) and filter-1 environments.

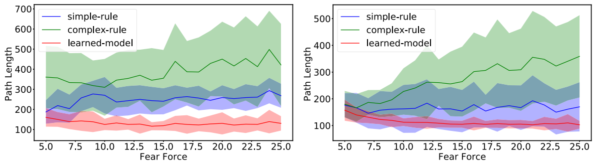

Fig. 9. Path length of the successful runs for different levels of fear coefficients in the filter-0 (left) and filter-1 environments. The unit of path length is meter in simulation. (1 m = 5 pixels).

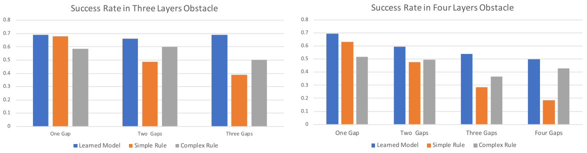

Fig. 10. Success rates in perturbed three-layer and four-layer obstacles.

B. *Success Rate With Perturbed Obstacles*

Here, we study the success rate of the learned model using the perturbed obstacles, which allow us to study the performance of the learned model in more realistic scenarios in which sensors are noisy and the environmental model may contain uncertainty. To evaluate the learned model, group-behavior parameters are randomized in the range identical to those used in the training. The success rate of the learned model (trained only with three layers) is obtained from the environments populated with three or four layers of perturbed obstacles. We conduct 100 experiments for each method and environment (2100 experiments in total), and the terminal time steps of each testing episode is 10 000.

Form Fig. 10(left), the success rate of the learned model is around 70% in all cases because the model is trained in the three-layer environment. From Fig. 10(right), the test environments has higher difficulty. Consequently, the success rates decreased but those of the learned model remain at least 15% higher than the rule-based methods.

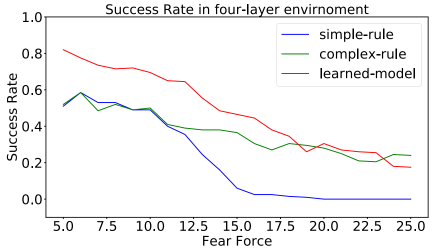

Fig. 11. Success rates with varying fear in a fixed four-layer environment.

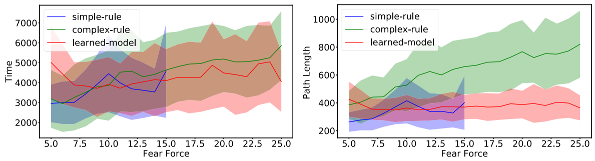

Fig. 12. Complete time (left) and path length (right) of success runs for different levels of fear force in a four-layer environment.

C. Robustness of Controller Learned From Perturbed Obstacles

In this section, we evaluate the model trained with perturbed obstacles by studying the robustness to different behavioral parameters. We conduct experiments in the most difficult four-layer obstacles environment with four gaps created by 8 fences. The obstacles are however not perturbed. The terminal time steps of each testing episode is 10 000.

Success Rate: In Fig. 11, when the fear force coefficient is small, the simple-rule and complex-rule methods share similar success rates, but our controller learns a side-to-side steering policy and performs better. When the fear force coefficient is larger than 10, the complex-rule method outperforms simple-rule method because of its more advanced stop-turning policy. Our controller learns to turn and has the similar result as the complex rule method does.

Controller Quality: We measure the controller quality as before. Fig. 12(left) shows the completion time of successful cases under different levels of fearfulness. The results show again that the learned model spends less time on average than the shepherd with the complex-rule method. The completion times between the learned model and the simple-rule method have no significant difference. Fig. 12(right) shows that the shepherd controlled by the learned model moves less than that controlled by the rule-based methods.

VII. Conclusion and Future Work

In this letter, we reported that controllers trained under the deep reinforcement learning framework can effectively herd a group of agents in environments populated with obstacles. We demonstrated that the train controller is robust to the uncertainties in the group behavioral model and in environmental model, e.g., S-shape, that has few path homotopy classes. We showed that the proposed learning-based method has higher success rate, shorter completion time and path length than the traditional rule-based method even in the presence of uncertainties. A major limitation of our work is that the proposed method can only handle the group of 2 to 4 agents. The rule-based method can handle dozens of agents. Although circle representation can potentially allow us to steer a larger group, navigation remains difficult if the circle gets too big, in comparison to the size of the free space. We plan to investigate different data representations, such as a tight bounding polygon. The second limitation is that different controllers must be trained for different types of environments. Training a single controller with all scenarios failed to converge after 48 hours with sufficient success rate. Finally, we also observe that training in environments with many path homotopy classes is difficult to converge [21].

References

[1] J.-M. Lien, O. B. Bayazit, R. T. Sowell, S. Rodriguez, and N. M. Amato, “Shepherding behaviors,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2004, pp. 4159–4164.

[2] B. Bat-Erdene and O.-E. Mandakh, “Shepherding algorithm of multi-mobile robot system,” in *Proc. 1st IEEE Int. Conf. Robotic Comput.*, 2017, pp. 358–361.

[3] J. Schubert and R. Suzic, “Decision support for crowd control: Using genetic algorithms with simulation to learn control strategies,” in *Proc. IEEE Mil. Commun. Conf.*, 2007, pp. 1–7.

[4] A. A. Paranjape, S.-J. Chung, K. Kim, and D. H. Shim, “Robotic herding of a flock of birds using an unmanned aerial vehicle,” *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 901–915, Aug. 2018.

[5] L. Chaimowicz and V. Kumar, “Aerial Shepherds: Coordination Among Uavs and Swarms of Robots,” *in Distrib. Auton. Robotic Syst. 6.*, 2007, pp. 243–252.

[6] J.-M. Lien, S. Rodriguez, J.-P. Malric, and N. M. Amato, “Shepherding behaviors with multiple shepherds,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2005, pp. 3402–3407.

[7] C. Vo, J. F. Harrison, and J.-M. Lien, “Behavior-Based motion planning for group control,” in *Proc. IEEE/RSJ Int. Conf. Intelli. Robots Syst.*, 2009, pp. 3768–3773.

[8] J. F. Harrison, C. Vo, and J.-M. Lien, “Scalable and robust shepherding via deformable shapes,” in *Proc. Int. Conf. Motion Games*, 2010, pp. 218–229.

[9] D. Strombom *et al.*, “Solving the shepherding problem: Heuristics for herding autonomous, interacting agents,” *J. Roy. Soc. Interface*, vol. 11, no. 100, 2014, Art. no. 20140719.

[10] K. Fujioka and S. Hayashi, “Effective shepherding behaviours using multi-agent systems,” in *Proc. IEEE Region 10 Conf.*, 2016, pp. 3179–3182.

[11] A. Pierson and M. Schwager, “Bio-inspired non-cooperative multi-robot herding,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 1843–1849.

[12] W. Lee and D. Kim, “Autonomous shepherding behaviors of multiple target steering robots,” *Sensors*, vol. 17, no. 12, 2017, Art. no. 2729.

[13] M. Baumann, “Learning shepherding behavior,” Ph.D. dissertation, University of Paderborn, 2015.

[14] C. J. Watkins and P. Dayan, “Q-learning,” *Mach. Learn.*, vol. 8, no. 3-4, pp. 279–292, 1992.

[15] H. T. Nguyen *et al.*, “A deep hierarchical reinforcement learner for aerial shepherding of ground swarms,” in *Proc. Int. Conf. Neural Inf. Process.*, 2019, pp. 658–669.

[16] C. W. Reynolds, *Flocks, herds and schools: A distributed behavioral model*, ACM, vol. 21, 1987, no. 4.

[17] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, 2015, Art. no. 529.

[18] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” 2015, *arXiv:1511.05952*.

[19] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning,” in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[20] M. G. Bellemare, J. Veness, and M. Bowling, “Investigating contingency awareness using atari 2600 games,” in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 864–871.

[21] J. Zhi and J.-M. Lien, “Learning to herd agents amongst obstacles: Training robust shepherding behaviors using deep reinforcement learning,” 2020, *arXiv:2005.09476*.

[22] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[23] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, *arXiv:1412.6980*.