Learning to Herd Amongst Obstacles from an Optimized Surrogate

Jixuan Zhi[1] and Jyh-Ming Lien[1]

***Abstract*— This paper investigates how a shepherd robot can efficiently steer a coherent group by intelligently moving behind the group in obstacle-filled environments. It was been shown that a model trained by deep reinforcement learning can guide a small number (2-4) of agents among obstacles. However, herding a larger group becomes significantly more challenging because it exhibits the characteristics similar to manipulating a deformable object, i.e., the system is dynamic and the problem is highly underactuated. To overcome these challenges, we show that a model can be trained more effectively via an optimized surrogate, such as a potential field that optimizes the control quality of the group without explicitly considering the placement of the shepherd. Our experiments demonstrate that the trained model is robust to noise for group behaviors and environments. Compared to the rule-based method, the proposed approach maintains a higher probability of guiding the sheep and better control quality.**

I. Introduction

Robotic shepherding aims to navigate a group of agents from one location to a goal location by controlling one or more external agents. The robotic shepherding problem has applications in animal herding [1], airport safety control [2], coordination between the unmanned aerial vehicles and unmanned ground vehicles [3] and civil crowd control [4].

Existing solutions can be categorized into two types of methods: rule-based methods and learning-based methods [5]. In rule-based methods, researchers defined a set of rules that determine the dynamics of the shepherding control system. These models perform well, even for a large number of sheep in obstacle-filled environments. However, these methods assume perfect knowledge of the controlled group and perform poorly in some cases such as turning the agents around a corner to a different direction. Conversely, learning-based approaches assume no prior knowledge about the group behavior model. However, most of these methods only handle blank environments without obstacles. Our previous work [5] showed for the first time that a learning-based method can be beneficial for herding agents among obstacles. However, this approach can only guide a limited number (2 to 4) of agents.

Given a space filled with obstacles, finding a solution to a given shepherding problem needs to consider both high-level path planning and low-level agent dynamics [6]. This problem becomes even more challenging if the quality of the solution is considered. Examples include avoiding splitting the group and minimizing the shepherd's travel distance. Moreover, herding a larger group further increases the difficulty because the system starts to exhibit the characteristics similar to a deformable object. In addition, the robot often does not have a perfect behavior model of the group, thus the trajectories of group agents are unpredictable for the robot. The aforementioned challenges prohibit us to train a deep reinforcement learning model directly.

In this work, we show that a model can be trained effectively via an optimized surrogate. We use a potential field that optimizes the control quality of the group as such a surrogate that enables us to steer the group without explicitly considering the placement of the shepherd first. Effective group representation is also critical for group control to handle noise or uncertainty. Some methods use a bounding circle to model the group in sparse environments [7], but it can not present the deformable quality of the group since the shape of the group is changed due to split or merger and some researchers represent the group as individual agents which is restricted to small flocks [8]. Harrison et al. [9] designed a deformable shape to represent the sheep to increase scalability. In this work, we represent the sheep as a pixel blob, which is a set of pixels within a user-defined distance. Therefore, the group can be represented as a union of all pixel blobs and every pixel value represents the density within. This representation is more accurate than bounding circle representations and is more effective than individual agent representation.

By constructing the optimized surrogate and a more accurate group representation, our method overcomes many current limitations. The main contributions of the work are: (1) Constructing the optimized potential field as the surrogate to train the shepherding model for a large number of sheep in an obstacle-filled environment. (2) a new data representation called pixel blobs to represent the group, and (3) comprehensive experiments illustrating the trained model that can guide the sheep with a higher success rate, a better control quality in comparison to the rule-based algorithms and the previous learning method. Fig. 1 illustrates the proposed method that uses an optimal potential field as a surrogate to train the model. The details are given in Sec. IV.

II. Related Work

Most works in robot herding focus on modeling and several works consider the problem from a control systems perspective. There are two main types of methods: rule-based methods and learning-based methods. We also briefly review some literature of herding from control theory community.

A. Rule-based methods

Existing rule-based methods can be classified into methods for obstacle-free environments and methods using path

[1]Jixuan Zhi and Jyh-Ming Lien are with the Department of Computer Science, George Mason University, 4400, University Drive MSN 4A5, Fairfax, VA 22030 USA, {jzhi,jmlien}@gmu.edu

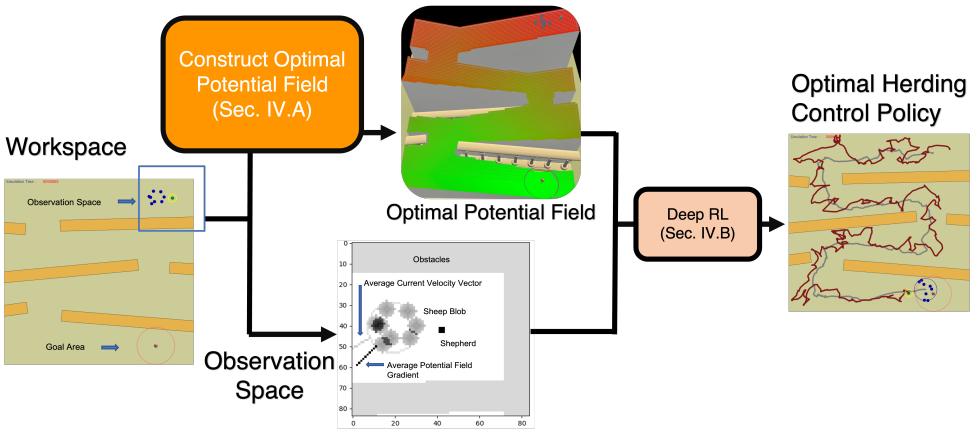

Fig. 1. An overview of the proposed method. Given an environment filled with obstacles and a group of agents, our approach generates an optimal potential field, which is used as a surrogate to train an optimal herding policy.

planning for obstacle-filled environments.

Without obstacles. Strömbom et al. [10] applied Reynolds' boids [11] and designed a heuristic approach with two rules: driving and collecting. Driving defines the rule that guides the group to the goal and collecting defines the rule that guides the outlier agent back to the large group. Bennett and Trafankowski [12] proposed an algorithm inspired by the human commands for sheepdogs. The sheepdog controls the group by moving around the group in a different direction. If the center of the group is on the left of the goal, the sheepdog circles the group in the clockwise direction, otherwise in the counter-clockwise direction.

With obstacles. Some researchers combine motion planning methods and rule-based techniques to shepherd in obstacle-filled environments. Bayazit et al. [13] developed a rule-based roadmap structure for shepherding behaviors. Vo et al. [14] extended the roadmap-based method with a tree-based planner to handle large groups in more complicated environments. Recently, Elsayed et al. [15] proposed an evolutionary path planner in a cluttered environment, This method performs better in completion time than the baseline but has a larger standard deviation in some scenarios. In general, these roadmap methods need to manually construct a series of complicated rules to address various scenarios.

B. Learning-based methods

Recently, group control with learning methods draws researchers' attention. Nguyen et al. [7] applied deep Q-learning for UAV herding. By training two models for two abilities of driving and collecting, the framework can herd a group of ground vehicle. Our previous work [5] considers obstacles in the environment. We trained the model in obstacle-filled environments under a reinforcement learning framework. The results show the approaches are more effective than the rule-based approaches but limited to a small number of sheep.

C. Control theory methods

The control community considers herding as part of swarm robotics. In [16], the authors constructed a nonlinear function to model the interaction between herder and targets, and developed a herding method in a switched-systems framework. In [17], multiple noncooperative herds and several herders are considered. The authors proposed control strategies by reducing the systems to nonholonomic vehicle models. Elamvazhuthi et al. [18] proposed a control strategy for a "leader" agent that guides a group of agents to a target probability distribution using the mean-field approach.

III. Preliminaries

We now formally define the shepherding problem. We assume that there exists one shepherd and a coherent group. Given the goal region defined as a circle (Fig. 1), the shepherd's task is to steer the group to the goal region.

A. Group Behavior Model

In this work, the group behavior model is based on Reynolds' Boids [11]. Each boid is influenced by three different forces from the nearby boids, namely the separation force that ensures the boids do not collide, the alignment force that the boids' heading directions are align, and that cohesion force that ensures the boids are near the center of the neighboring boids. In addition to the forces for obstacle avoidance and damping, a *fear* force is added to move the group away from the shepherd. In each simulation step, the weighted sum of these forces update the state (acceleration, velocity and location) of each sheep. Altering the coefficients of different forces can model different group behaviors.

Note that neither these behavior parameters nor the models are shared with the shepherd. Because our shepherd robot has no prior knowledge of the sheep's behavior model, the proposed work can be generalized to other behavior models.

B. Deep Q-Learning

We consider four components in a reinforcement learning problem: (1) current state, (2) action, (3) transition model from the current state to the next state, and (4) reward. We want to find the optimal policy in every state by maximizing the cumulative reward over time. In motion planning,

reinforcement learning can be viewed as a type of feedback planning with explicit model [19].

Reinforcement learning is beneficial to addressing the infinite number of states in the shepherding problem. For example, a deep neural network is used in reinforcement learning [20] for approximating the state and action pair value. Researchers proposed different methods to improve the performance of original deep Q-learning. In this work, priority replay buffer [21], double Q-learning [22] and noisy network [23] are used.

C. Potential Function Methods

A potential field derived from a given workspace can be used to navigate a single or a group of agents. We can view the potential function as energy hence the gradient of the potential function is force [24]. The force can be applied to the particle robot for moving. Usually, attractive forces are directing the robot to the goal, and repulsive forces keep the robot away from the obstacles. The combinations of those forces guide the robot from the start position to the goal position among obstacles. The potential field can be constructed so it is not sensitive to the uncertainty in the group of agents. This is beneficial for the shepherding problem because the behavior model of the flock can be unknown or very noisy.

IV. Method

There are two parts in our proposed method. see Fig. 1. Firstly, we will construct an optimized global potential field, in which the group members will gradually flow into a goal state efficiently by following the gradient without considering the controllers. Secondly, we use deep reinforcement learning to train the model guided by the optimized potential field. Namely, during the actual group control, the planner positions the shepherd in the vicinity of the group to produce a fear force approximating the force resulting from the gradient of the optimized global potential.

A. Construct Optimized Potential Field

We construct a $N \times N$ grid map for the environment to build the potential field. The grid size is equal to 2.5 pixels for the learning part.

1) Potential Field Functions: Here we introduce the building blocks of the potential field used in the proposed work.

Attractive potential. The attractive potential $U_{att}(p)$ is monotonically increased with distance between current position p and the goal position p_{goal}. For each position p in the environment, the attractive potential is computed as follows:

$$U_{att}(p) = \begin{cases} \frac{1}{2}\alpha||p, p_{goal}||^2, & \text{if (a)} \\ \alpha(d^*_{goal}||p, p_{goal}|| - \frac{1}{2}(d^*_{goal})^2), & \text{if (b).} \end{cases} \quad (1)$$

and

$$\begin{cases} ||p, p_{goal}|| \leq d^*_{goal}, & \text{(a)} \\ ||p, p_{goal}|| > d^*_{goal}, & \text{(b)} \end{cases} \quad (2)$$

where $\alpha > 0$ is a scaling factor, $||p, p_{goal}||$ is the distance between position p and goal position p_{goal}. d^*_{goal} is a threshold where planner switches between two potential functions.

We need to calculate the distance $||p, p_{goal}||$ on the grid map, and we use wave-front algorithms to compute the distance to the goal for each grid with four-connectivity neighbors. The maximum distance d^{max}_{goal} to the goal can be found, then we parameterize the threshold d^*_{goal} based on the maximum distance d^{max}_{goal}:

$$d^*_{goal} = \beta d^{max}_{goal}, \quad (3)$$

where β is a scaling factor between 0 and 1.

Combining Eq. 1, Eq. 2 and Eq. 3, the attractive potential can be computed when given the scaling factor α and β.

Repulsive potential. For repulsive potential value, the repulsive force should be stronger when the robot is closer to an obstacle. For each position p in the environment, the repulsive potential can be computed as follows:

$$U_{rep}(p) = \begin{cases} \frac{1}{2}\gamma(\frac{1}{D(p)} - \frac{1}{d^*_{obst}})^2, & \text{if} D(p) \leq d^*_{obst} \\ 0, & \text{if} D(p) > d^*_{obst}. \end{cases} \quad (4)$$

where $\gamma > 0$ is a scaling factor, $D(p)$ is the distance from position p to the closest obstacle in the space. d^*_{obst} is the threshold to admit the robot to disregard obstacles.

Similarly, we use the Breadth-first search method to compute the distance $D(p)$ to the closest obstacle and the maximum closest distance d^{max}_{obst}, then we parameterize the threshold d^*_{obst} based on the maximum closest distance d^{max}_{obst}:

$$d^*_{obst} = \delta d^{max}_{obst}, \quad (5)$$

where δ is a scaling factor between 0 and 1.

Applying Eq. 5 to Eq. 4, we can calculate the repulsive potential values when given the scaling factor γ and δ.

Compute the gradient. We cannot compute the gradient by differentiating the potential function with respect to distance in a grid map. Therefore, we compute the gradient for each cell as follows: we define a circle area and the cell position P_{center} is the center of the circle. In this circle area, we find the cell P_{min} with the smallest potential value compared to the potential value of the center cell. The difference between these potential values can be the magnitude of the gradient, and a vector from center P_{center} to the cell P_{min} can be the direction of the gradient. The force produced by the gradient is proportional to the gradient of the potential function.

2) Find the optimal parameters of potential field for navigation: For different environments, we need to discover the optimal parameters of the potential field for sheep to flow into a goal area efficiently and maintain group behaviors without considering the shepherd. We use the completion time and maximum group radius as objectives to optimize the potential field parameters, namely, we want to find an optimized potential field that can minimize the completion time and group radius when the sheep "flow" into the global minimum. A small radius means the sheep remain a coherent

group moving through the field and a shorter completion time means the sheep reach the goal faster.

We find the optimal parameters in two steps. Firstly, we find the optimal parameters for attractive potential function because it is more important than repulsive potential function. We set the scaling factor γ and δ for the repulsive potential function to be 0. We fix the scale factor $\alpha = 1$ as well because it is a linear factor. We only need to find the optimal value for scale factor β. We change the value β from 0 to 1 with an interval of 0.05. Then we did 100 experiments for each value and fine the best value β^* that optimize the time and radius constraints.

Secondly, we find the optimal values of scaling factor γ and δ for repulsive potential function. We set the α to be 1 and β to be β^* for attractive potential function. Then we chose a pair (γ, δ) for repulsive potential function. The value of γ is changed from 1 to 100 with an interval of 5. δ is changed from 0 to 1 with an interval of 0.2. (the interval should be larger than or equal to $1/d_{obst}^{max}$ to avoid repeated calculation). Similarly, we did 100 experiments for each pair value and fine the best pair values (γ^*, δ^*) that optimize the time and radius constraints.

Finally, we find a set of values $(1, \beta^*, \gamma^*, \delta^*)$ as the local optimal parameters for this environment to complete the task. See Fig. 1 for an optimal global potential field for a three-layer environment.

B. Deep Q-Learning using Optimized Potential Field

1) Model Architecture: The model (Fig. 2) follows the work [20] and [23]. There are three convolution layers and two fully connected layers, and each hidden layer followed by a rectifier nonlinearity (RELU) unit. Compared to the previous work [5], we add independent Gaussian noise for every weight in the fully-connected layers to aid efficient exploration. The parameters of noise are stored in the layer and can be trained as the same as the standard linear layer.

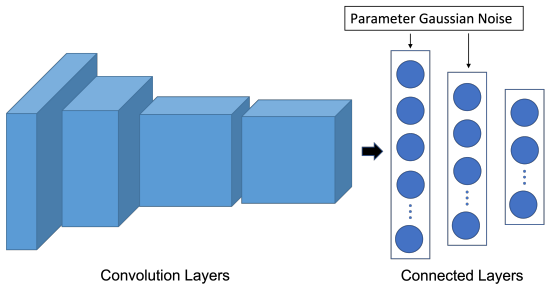

Fig. 2. Deep reinforcement learning architecture with Gaussian noise added for each weight in the fully-connected layers.

2) Observation Space: We assume that the shepherd robot can only have a local view of the world state centered on itself. Practically, 84×84 pixels image is used as input for the network. To meet the criteria of the Markov Decision Process domain, we maintain four frames from the past based on *frame stack technique* [25] and apply them for observation space of the current state.

Representing the Flock. An important question in learning the model is how to represent the group. In the previous work [5], we used a bounding circle but this simple representation does not capture the deformable quality of group sheep. Additionally, some work [8] represented the flock as individual agents which is limited to a small number of agents.

To address this problem, we propose to investigate a method that represents the contour of the group as a pixel blob in the intensity map λ in S_j with a bounding circle. The pixel blob of a single member g of a group is simply the set of pixels within the distance r of g, where r is the sensor range of g. The pixel blob of a group (called group blob) is therefore the union of all pixel blobs of its members and each pixel value represents the density within. The group blob may have several connected components if the group has separated into sub-groups. see Fig. 1 for the sheep representation in observation space.

Optimized Potential Field. Since we use the optimized potential function as a surrogate to navigate the sheep, the shepherd should know the information about the potential function. Therefore, we choose the average gradient direction of the flock to represent the potential field information. We use a black straight line to represent the gradient direction, and we connect the center of the flock and the point which has the minimum potential value in a circle range, but we do not want the line to interfere with the pixel blob too much, so we double the line's length in the same direction and cut the inner part. In normal cases, the larger portion of the gradient line is out of the bounding circle and pixel blob. In fact , the shepherd needs to know the difference between the expected direction of the flock and the current direction of the flock, so we add an average velocity of the flock as a grey line in the observation space follows the same policy. These two directions of information have different pixel values. Fig. 1 illustrates the observation space.

Environment information. In every local 84×84 frame, the central shepherd is represented as 3×3 pixels, the goal is represented as one pixel. obstacles and boundaries in the workspace are solid polygons. Different objects have different pixel values in the frame. See Fig. 1 for an instance.

3) Action Space: The action space follows the work [5], We consider semi-discrete actions for the shepherd robot. In each simulation step, the planner selects one of eight directions and applies a fixed magnitude force along that direction to the shepherd. To cover all the directions, a clipped Gaussian noise is applied to the force with a maximum deviation of $22.5°$.

4) Reward Function: The reward structure contains three parts: force projection reward, misbehavior reward and task completing reward.

Force Projection Reward. To guarantee that the sheep are advancing in the needed direction guided by the global potential field, we propose the force projection reward.

For every sheep, we compute the fear force $\vec{f}$ by the relative position of sheep and shepherd. In fact, based on the position of sheep, we can look up the global potential

field grid map and compute the force $\vec{a}$ by the gradient of the optimized potential field. Then a projected vector $\vec{r} = \frac{\vec{f}\cdot\vec{a}}{||\vec{a}||^2}\vec{a}$ can be calculated by the vector $\vec{f}$ onto vector $\vec{a}$. The magnitude and sign of the reward can be determined by the length and direction of the projected vector $\vec{r}$, namely, the sign of reward is identical to the dot product of vector $\vec{f}$ and vector $\vec{a}$. The numerical values used for the projection reward function is usually between -2 and $+2$.

To motivate exploration in large search space, several shepherd actions are penalized, such as the action that does not push the sheep for moving.

Violation Reward and Task Completing Reward. Two types of violation rewards are considered. Firstly, since we use group maximum radius as a term to find the optimal potential field, namely, maximum radius measures the cohesion of the flock, a big negative reward is given during the training if the radius is very large. We define the threshold as the sensor range (30 pixels). However, if the radius is too large (40 pixels), the training is terminated because the outlier sheep is not possible to move back to the larger group. Secondly, effective explorations are important for training, then the shepherd should always move around the flock. A negative reward is given if the shepherd is far away from the group. The threshold is 60 pixels and the termination threshold is 80 pixels this time.

If the termination threshold is met or the training episode exceeds the predefined time step, this episode is terminated and a failure is reported. A large reward (-20) is also given. Conversely, we give a huge positive reward ($+20$) when the flock arrives at the goal region and success is reported for the training episode.

V. Training Shepherding Controller

The training details are described in this section. We add controlled random noise in the environments and in the behavioral models during the learning process. In real-life scenarios, noise exists in many situations, for example, noise can be found in sensing the position of sheep or the location of the obstacle. Therefore, adding simulated noise to our model is beneficial for our method in adaptability.

The training parameters include: the learning rate is 0.0001, the reward discount factor is 0.99, and the size of the replay buffer is 25,000. The ADAM optimizer is used with training batches of size 32.

We apply the frame-skipping skill that the shepherd selects new actions on each $k = 5$th frame, then this selected action is not changed in the other skipped frames. Furthermore, we decide the maximum training time steps of the episode for different scenarios by the simple rule method [26].

To investigate the quality of the learned model, two types of obstacles in the scenario are selected: one is predefined obstacles, and the other is perturbed obstacles. All the simulated environments are 50×50 m^2 ($1\ m = 5$ pixels).

1) Predefined Obstacles: In this part, we focus on the scenario with fixed obstacles. Practically, the scenario (including in a range) is commonly predefined, then the flock move to one position for drinking, or move to another position for grazing. Thus two different predefined scenarios are created as shown in Fig. 3. The left figure shows the lattice environment and the right figure shows the pachinko environment. We set one position as the start position and several positions as different goals. We train one model for these two scenarios.

In these predefined environments, the group behavior parameters are fixed. The separation coefficient is 1, the cohesion coefficient is 5, the alignment coefficient is 10, and the fear coefficient is 10. The range of terminating time of every episode is between 1000 and 6000 based on different goals. It takes an average of 20 hours for training to converge.

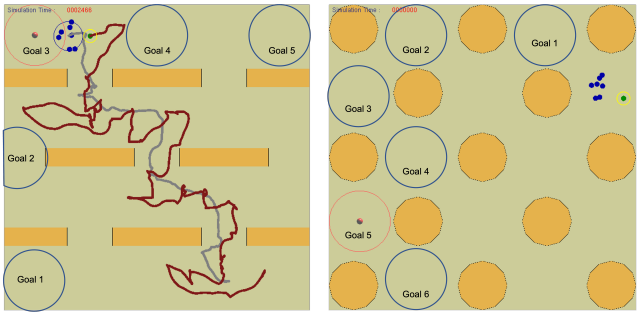

Fig. 3. The two fixed obstacles environments. Left: lattice environment and two paths generated by the proposed method, red: shepherd path, grey: sheep center path. Right: pachinko environment.

2) Perturbed Obstacles: By incorporating the randomness to predefined obstacles, we study perturbed obstacle models with random noise, namely, the U-turn model and the gap model are shown in Fig. 4.

The three-layer obstacle scenarios are created for training with these two models. We always have three U-turn patterns and can add at most three gap patterns during training. For example, in Fig. 4 (left), there are three gap patterns and U-turn patterns with six fences, and it changes to two-gap patterns and three U-turns patterns in Fig. 4 (right) when one median shorter fence is removed. For each fence, the starting location and orientation are set. During training, Gaussian noise is added to these set values. The maximum deviation of location is 2.5 pixels, and the maximum deviation of orientation is $8°$. The three shorter fences can vanish or appear randomly to add gap patterns in the training process. We alternate start and goal positions randomly selected from the top and bottom of the four regions divided by the three obstacles in the environment. The environments were created automatically by a script. Moreover, the number of sheep can be changed from 5 to 6 randomly.

To train a robust controller, the group behavior parameters are randomly changed from a clipped Gaussian distribution. Namely, the separation coefficient is selected between 0.5 and 1.5, the cohesion coefficient is chosen between 3 and 7, the alignment coefficient is selected between 8 and 12, and the fear force coefficient is chosen between 8 and 12. The optimal parameters of the potential field for perturbed obstacles are found when there is no Gaussian noise for all things in environments, then these parameters are not

changed during training.

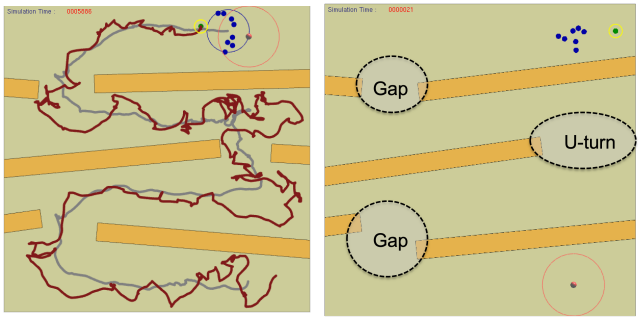

Fig. 4. The positions and orientations of the fences are perturbed to form 3 layers obstacles during training and testing.

VI. Evaluating the Learned Controller

In this section, we show the benefits of the proposed approach over the previous method [5] and the rule-based approaches. We study the effectiveness and quality of the learned controller in scenarios with noise. Two different rule-based methods are compared with our method, namely simple-rule and complex-rule methods [26]. In these two methods, a roadmap is used to provide the shepherd a rough idea of where to move the sheep. Additionally, the complex-rule method applies a safe zone method to approach the group and a stop-turn steering method [26] to guide the flock in a different direction. These two approaches can merge the outlier sheep back into a larger group. We also compare the state-of-the-art learning method (DRL + Roadmap) [5]. All experiments are conducted on a workstation with an NVIDIA TITAN X (Pascal) graphic card. The PC is equipped with an Intel Xeon 2.3 GHz CPU and 32Gb RAM.

A. Success rate in predefined environment

In this section, we focus on the success rate of the trained model in a predefined scenario with different goal areas. The group behavior parameters are identical to those used in the training. There are 50 tests for each approach and each goal area. The number of sheep is 7 in all tests, and the alignment coefficient is set between 8 to 12.

Fig. 5 shows the success rate of various methods for different goal areas in the lattice environment. In most cases, the proposed method performs better or remains constant compared to rule-based methods except goal 2 region. However, Fig. 5 shows the success rate of various methods for different goal regions in pachinko environments. In most cases, the proposed method is not as good as the complex rule methods, but it outperforms the simple rule method. Finally, the previous method (DRL + Roadmap) is not able to handle a large number of sheep, so the success rate is zero in all experiments.

B. Success rate in noisy perturbed obstacle environments

We focus on the success rate of the trained model in environments filled with noisy perturbed obstacles. We impose the same randomness into perturbed obstacle environments as

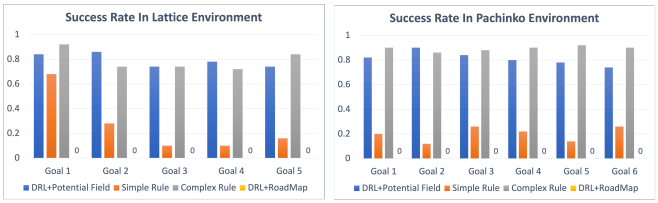

Fig. 5. Success rates of herding 7 sheep in lattice environments (left) and pachinko environments (right) with different goal areas.

the training process, such as changing the location orientation of each fence randomly and forming the U-turn patterns or gap patterns randomly.

This study helps us understand the performance of the proposed method in more pragmatic scenarios, such as noisy sensors or uncertain models.

To assess the trained model, the local behavior coefficients are chosen randomly similar to the values used in the training part, and four variations of the environment are designed for testing: 0-3 gaps are shown in Fig. 4.

There are 200 experiments for each approach and environment, and the terminating time of each episode is 11000 for 7 or 8 sheep. For 7 and 8 sheep, the alignment coefficient is set between 13 to 17.

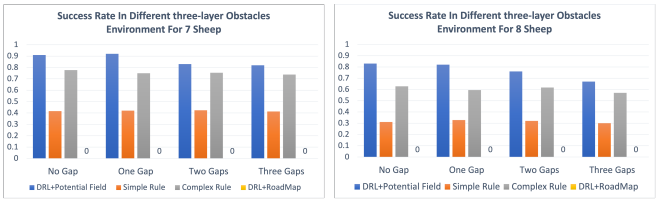

Fig. 6. Success rate of herding 7 (left) and 8 (right) sheep in different three-layer obstacles.

Fig. 6 (left) displays that the success rate of the proposed method (DRL+Potential Field) is about 80% in no gap or one gap situation for 7 sheep. When the number of gaps increased, the success rate decreased, but the proposed method still performs better than the rule-based methods. It is shown that the proposed method maintains about 10% higher than the complex rule methods. In fact , the previous method (DRL + Roadmap) only handled a limited number of sheep, so the success rate is zero in all experiments. Similarly, Fig. 6 (right) shows the success rate for 8 sheep. The success rate of the proposed method is about 65% to 85% for all cases, and it outperforms the complex-rule method by about 10%.

C. Robustness of model trained from perturbed obstacles

We assess the robustness of the model under uncertainty, especially, under tainted various behavioral models. The models are changed by their coefficients. In our tests, we fixed several coefficients and change one or two coefficients to see the difference, specifically, the fear force coefficient, the alignment coefficient, separation and cohesion pair coefficient. We also introduce the randomness in the behavior model. The experiments are conducted in the three-layer

obstacles scenarios with six fences. The fences are fixed to control variables. Similarly, the terminating time of each episode is 11000 for 8 sheep. We not only show the success rate of different coefficients but also show the control quality of the method, which includes maximum radius, completion time, path length and energy cost.

Success Rate for 3 sheep under different degrees of Fear Force. First, we show the experiments under different degrees of Fear Force. we compare our method to the previous method [5] using the same coefficients of cohesion, separation and alignment. The fear force coefficient is also between 5 and 25. The terminating time of each episode is 8000. Fig. 7 shows the proposed method preserves a higher success rate than the other two rule-based methods and the previous learning method. Even if the fear force coefficient is higher than 20, the proposed method can still maintain a success rate around 80%, which is larger than the previous learning method (70%), and the complex rule method (60%).

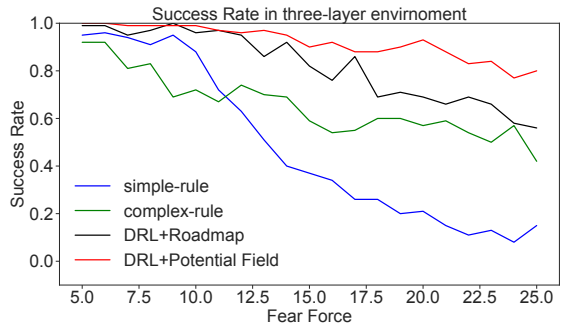

Fig. 7. Success rates for 3 sheep in a three-layer environment with different degrees of Fear Force. The x-axis represents fear force coefficients.

Success Rate for 8 sheep under different degrees of alignment. Here we show the experiments under different degrees of alignment. We come back to 8 sheep. Fig. 8 (left) shows that when the alignment coefficient increases, the success rate also increases. Alignment is one factor to measure the cohesion of the group. The proposed method has a little higher success rate than the complex rule method when the alignment coefficient is larger than 9.

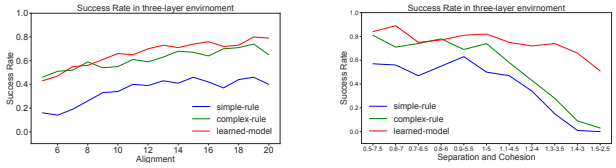

Fig. 8. Success rates for 8 sheep in a three-layer environment under different degrees of alignment (left) and separation and cohesion (right). The x-axis represents alignment coefficients (left) and the pair of separation coefficients and cohesion coefficients (right) .

Success Rate for 8 sheep under different degrees of separation and cohesion. Here we show the experiments under different degrees of separation and cohesion. The number of sheep is 8. Fig. 8 (right) shows that when the group is coherent, the proposed method has a little higher success rate than the complex rule method. However, when the flock becomes more difficult to control (low cohesion and high separation), the model shows a significant difference as opposed to the rule-based methods.

Completion Time. One of the control qualities can be evaluated by the required completion steps or time for shepherding. Fig. 9 (left) indicates the average completion steps of successful runs under different degrees of separation and cohesion. The average completion time of the three methods increased when the sheep becomes more difficult to herd. When the sheep are more coherent, the proposed approach has fewer average completion steps than the simple rule method, but more average completion steps than the complex rule method. When the sheep are less coherent (high separation and low cohesion), the proposed controller has a shorter time than the other two approaches.

From our observation, the proposed maintain "passive" behavior to herd the sheep, namely, when the group becomes scattered, the controller does not move too much, it waits until the group becomes coherent by themselves through group behaviors forces. This phenomenon explains that the proposed method has a longer average completion time compared to the complex rule method in most cases.

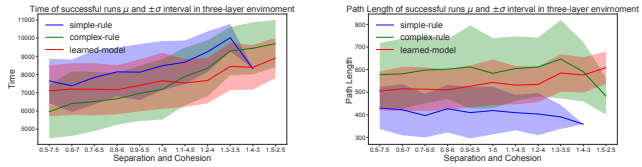

Fig. 9. Completion time (left) and Path Length (right) of success tests for 8 sheep under different degrees of separation and cohesion in a three-layer scenario.

Path Length. The path length can also be used to measure the quality of control. Fig. 9 (right) shows the average path length of successful cases under different degrees of separation and cohesion. When the sheep is more coherent, the proposed method has a longer average path than the simple-rule method but has shorter path lengths than the complex-rule method. When the sheep is less coherent, the rule-based methods have a shorter path length than the proposed method, but they have a lower success rate in these cases, see Fig. 8 (right). Additionally, our method has a smaller standard deviation of path length than the complex-rule method in all cases.

Maximum Radius. When we construct an optimal potential field for shepherding, maximum radius is one of the terms for optimization. Our model guided by optimal potential should show the same quality.

Fig. 10 (left) shows the maximum radius of sheep in successful cases under different degrees of separation and cohesion. The figure shows that the trained model maintains a significantly smaller maximum radius on average than the other two rule-based methods.

Energy Cost. When focusing on the energy cost, we apply a criterion called energy cost in unit time to measure the control quality. It is computed from the division of path

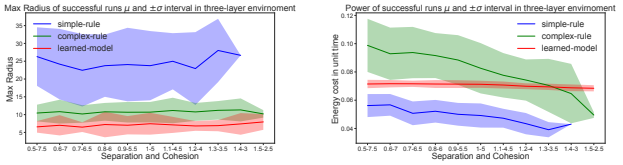

Fig. 10. Maximum radius (left) and Energy cost in unit time (right) of success tests for 8 sheep under different degrees of separation and cohesion in a three-layer scenario.

length by completion time in a successful case. Fig. 10 (right) shows the average energy cost in unit time (power) of successful cases under different levels of separation and cohesion. Our method has less energy cost in unit time than the complex-rule method but more than the simple-rule method when sheep is more coherent. When the sheep are less coherent, the rule-based method has less energy cost in unit time but they have a low success rate in these cases, see Fig. 8 (right). The figure also shows our method maintains a stable energy cost in unit time under all cases, it has the smallest standard deviation of energy cost in unit time than the other two rule-based methods.

VII. Conclusion

To allow the learned controller to herd a larger group, we proposed to construct an optimal potential field of obstacle-filled environment as the surrogate to train the controller in a deep reinforcement learning framework. In addition, a new and powerful data representation is introduced to increase the accuracy of the learned controller. Using pixel blob to represent the group in observation space and potential field as a surrogate, our new method outperforms the previous method in herding a large number of sheep. The experiments show that the new method is robust to noise in group behaviors and environments. The test results show that the proposed approach preserves a higher probability in herding tasks and maintains better control quality than the traditional rule-based methods. However, when the number is larger than 10, the performance of the proposed method decreases. We plan to investigate strategies to allow multiple shepherds in a learning framework to address this scalability issue.

References

[1] B. Bat-Erdene and O.-E. Mandakh, "Shepherding algorithm of multi-mobile robot system," in *2017 First IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2017, pp. 358–361.

[2] 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 Transactions on Robotics*, vol. 34, no. 4, pp. 901–915, 2018.

[3] L. Chaimowicz and V. Kumar, "Aerial shepherds: Coordination among uavs and swarms of robots," in *Distributed Autonomous Robotic Systems 6*. Springer, 2007, pp. 243–252.

[4] J. Schubert and R. Suzic, "Decision support for crowd control: Using genetic algorithms with simulation to learn control strategies," in *MILCOM 2007-IEEE Military Communications Conference*. IEEE, 2007, pp. 1–7.

[5] J. Zhi and J.-M. Lien, "Learning to herd agents amongst obstacles: Training robust shepherding behaviors using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4163–4168, 2021.

[6] N. K. Long, K. Sammut, D. Sgarioto, M. Garratt, and H. A. Abbass, "A comprehensive review of shepherding as a bio-inspired swarm-robotics guidance approach," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 4, pp. 523–537, 2020.

[7] H. T. Nguyen, T. D. Nguyen, M. Garratt, K. Kasmarik, S. Anavatti, M. Barlow, and H. A. Abbass, "A deep hierarchical reinforcement learner for aerial shepherding of ground swarms," in *International Conference on Neural Information Processing*. Springer, 2019, pp. 658–669.

[8] R. Vaughan, N. Sumpter, J. Henderson, A. Frost, and S. Cameron, "Experiments in automatic flock control," *Robotics and autonomous systems*, vol. 31, no. 1-2, pp. 109–117, 2000.

[9] J. F. Harrison, C. Vo, and J.-M. Lien, "Scalable and robust shepherding via deformable shapes," in *International Conference on Motion in Games*. Springer, 2010, pp. 218–229.

[10] D. Strömbom, R. P. Mann, A. M. Wilson, S. Hailes, A. J. Morton, D. J. Sumpter, and A. J. King, "Solving the shepherding problem: heuristics for herding autonomous, interacting agents," *Journal of the royal society interface*, vol. 11, no. 100, p. 20140719, 2014.

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

[12] B. Bennett and M. Trafankowski, "A comparative investigation of herding algorithms," in *Proc. Symp. on Understanding and Modelling Collective Phenomena (UMoCoP)*, 2012, pp. 33–38.

[13] O. B. Bayazit, J.-M. Lien, and N. M. Amato, "Roadmap-based flocking for complex environments," in *10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings.* IEEE, 2002, pp. 104–113.

[14] C. Vo, J. F. Harrison, and J.-M. Lien, "Behavior-based motion planning for group control," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 3768–3773.

[15] S. Elsayed, H. Singh, E. Debie, A. Perry, B. Campbell, R. Hunjel, and H. Abbass, "Path planning for shepherding a swarm in a cluttered environment using differential evolution," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2020, pp. 2194–2201.

[16] R. A. Licitra, Z. D. Hutcheson, E. A. Doucette, and W. E. Dixon, "Single agent herding of n-agents: A switched systems approach," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 14 374–14 379, 2017.

[17] A. Pierson and M. Schwager, "Controlling noncooperative herds with robotic herders," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 517–525, 2017.

[18] K. Elamvazhuthi, Z. Kakish, A. Shirsat, and S. Berman, "Controllability and stabilization for herding a robotic swarm using a leader: A mean-field approach," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 418–432, 2020.

[19] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[21] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[22] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[23] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, *et al.*, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[24] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, and W. Burgard, *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.

[25] M. G. Bellemare, J. Veness, and M. Bowling, "Investigating contingency awareness using atari 2600 games," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[26] J.-M. Lien, O. B. Bayazit, R. T. Sowell, S. Rodriguez, and N. M. Amato, "Shepherding behaviors," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 4. IEEE, 2004, pp. 4159–4164.