Multi-Drone Collaborative Shepherding Through Multi-Task Reinforcement Learning

Guanghui Wang, Junkun Peng, Chenyang Guan, Jinhua Chen, and Bing Guo

***Abstract*—Robotic shepherding has become indispensable in animal husbandry and crowd management, offering a modern solution to traditional challenges. Drone Automated Shepherding leverages advanced maneuverability and an extensive field of view to improve the efficiency of these tasks, which are typically labor-intensive and time-consuming. Existing methods for managing large herds face significant challenges due to insufficient coordination among multiple drones and the complexities involved in simultaneously executing diverse shepherding tasks. This paper aims to enhance the execution of multiple shepherding tasks by optimizing drone coordination, designing optimal flight paths, and reducing flight time. To harness the potential of reinforcement learning, we develop a multi-drone collaborative shepherding environment that facilitates efficient drone training using a dense reward system. Additionally, we employ a multi-task deep reinforcement learning algorithm that enhances the sample efficiency and reward performance by leveraging shared information across tasks within this environment. Two specific tasks, driving and collecting, are used to assess the performance of our methodology. The effectiveness of our approach is measured against a classical solution named CTRL, examining metrics such as success rate, completion time, and flight path length. Results indicate that our approach significantly outperforms the CTRL in all measured metrics. Visualization of drone trajectories provides further evidence of our enhanced collaboration and efficiency in shepherding operations. Real-world experiments are conducted on a square of 300 m^3, where two drones utilize our method to guide four small autonomous vehicles from the starting area to the goal area within 30 seconds.**

***Index Terms*—Path planning for multiple mobile robots or agents, reinforcement learning, collaboration, shepherding.**

I. Introduction

SHEPHERDING, an operation pivotal in both animal husbandry [1] and crowd control [2], poses significant challenges. This paper focuses on sheep herding–a task that traditionally requires considerable human and animal labor. Historically, shepherds and sheepdogs perform this time-consuming task. However, the emergence of Drone Automated Shepherding (DAS) introduces an innovative approach, eliminating the need for human intervention and thereby reducing human error and associated risks.

Drones enhance shepherding by offering superior maneuverability and an extensive field of view, which are essential for effectively navigating diverse terrains. Shepherding is often delineated into two primary tasks, driving and collecting [3]. Driving requires a set of drones (the herders) to guide the flock of sheep (the targets) from a designated starting area to a specified goal area, while collecting involves retrieving stray targets to the main flock. The primary constraint for DAS is the herder's limited battery life [4], [5], which necessitates completing tasks both efficiently and effectively. Optimizing flight paths can not only conserve energy but also minimize flight time by enabling drones to fly shorter and more efficient routes. Reducing flight time can boost the overall productivity of the shepherding operation.

This paper addresses this critical issue by providing a strategy for ensuring the successful completion of all shepherding tasks while optimizing flight paths and minimizing flight time.

Rule-based and learning-based algorithms are two main categories of solutions for this shepherding problem. Rule-based algorithms [6], [7], [8] employ a set of predefined rules to model the interaction between herders and targets. These approaches are characterized by their ease of understanding and implementation. Furthermore, given a particular set of targets' behavior, they can offer consistent and predictable outcomes. However, their scalability is significantly limited due to the inherent difficulty in modeling the entirety of targets' behavior and correspondingly formulating herders' action rules. A few rule-based approaches [9], [10], [11] consider some uncertainty in sheep's models, but still struggle with high-dimensional state spaces. In real-world scenarios, flocks can consist of thousands of sheep, leading to an extremely complex and uncertain environment that is challenging to model. Conversely, learning-based algorithms [12], [13] have no need to fully model response rules of herders towards the target flock and empower herders to acquire adaptive herding behaviors through learning. They have the potential to leverage deep neural networks for processing high-dimensional input data and perform effectively in various environments. Particularly, algorithms based on deep reinforcement learning (DRL) enables herders to make successive decisions during the herding process, a capability that supervised

Received 13 May 2024; accepted 9 September 2024. Date of publication 25 September 2024; date of current version 10 October 2024. This article was recommended for publication by Associate Editor J. Liang and Editor A. Faust upon evaluation of the reviewers' comments. This work was supported by the Open foundation of State Key Laboratory of Plateau Ecology and Agriculture, under Grant KF-2023-ZZ-02. *(Guanghui Wang and Junkun Peng are co-first and contributed equally to this work.) (Corresponding author: Bing Guo.)*

Guanghui Wang, Chenyang Guan, and Bing Guo are with Qinghai University, Qinghai 810016, China (e-mail: ys220854060318@qhu.edu.cn; ys230855010370@qhu.edu.cn; 2019990066@qhu.edu.cn).

Junkun Peng and Jinhua Chen are with Tsinghua University, Beijing 100084, China (e-mail: pjk20@mails.tsinghua.edu.cn; chenjinh24@mails.tsinghua.edu.cn).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3468155, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3468155

learning methods are unable to achieve. More importantly, these methods are capable of adapting to diverse behaviors exhibited by the target flock through extensive interaction with the environment. Furthermore, methods based on DRL can be categorized into model-based and model-free approaches. Among these, the model-free approach simplifies the learning process by eliminating the need for an explicit model of the environment, making it more robust and adaptable to dynamic scenarios. Given their adaptability, this paper primarily explores algorithms based on model-free DRL for shepherding tasks.

In the view of the number of herders, most shepherding strategies currently focus on utilizing a single shepherd [14], [15], [16], [17], which poses challenges in managing larger flocks due to the shepherd's limited influence range. This limitation often results in excessive shepherding time as the shepherd attempts to cover all areas around the flock. Some recent approaches [18] have attempted to mitigate these issues by employing multiple herders, each responsible for a segment of the flock. These efforts face significant challenges due to poor collaboration between herders.

Moreover, existing methods often target one specific task like driving [17], [18], [19], with only a few considering the selection of multiple tasks as the action of herders [16] or developing independent policies for each task [15]. Such methods either restrict the herders from performing fine-grained actions or fail to capitalize on the correlations between different shepherding tasks, such as similar targets' responses to the herders' same action.

In response to these limitations, this paper advocates for the use of Multi-Task Deep Reinforcement Learning (MTDRL). MTDRL facilitates the simultaneous learning of multiple tasks and the optimization of each through the strategic utilization of shared information across tasks. This approach promises to enhance both the efficiency and effectiveness of DAS systems.

To address the raised shepherding problem, this paper introduces a novel environment designed for multi-drone collaborative shepherding. Unlike traditional methods that assign specific areas to each herder, our environment enables unrestricted flight for all herders across the entire area, facilitating collaborative herding of the target flock based on their velocities, thereby enhancing adaptability and operational efficiency. The movement of the target flock is simulated using the boids model [20], which also accounts for the flock's reaction to the approaching herders. Within this environment, we implement a dense reward function that incentivizes herders to herd targets towards the goal area efficiently, focusing on minimizing flight time and optimizing flight paths. Moreover, a MTDRL algorithm is applied within the context of the shepherding specifically for learning multiple shepherding tasks. This algorithm recognizes the commonalities between driving and collecting tasks, such as similar observation and action spaces, and shared dynamics such as the herders' repulsion from the targets and the overall flock behavior. Specific adaptations are developed to address challenges encountered in applying the MTDRL algorithm to shepherding tasks. These adaptations involve the design of suitable multiple shepherding tasks and modifications to the reward function to ensure that the model effectively advances toward optimal performance. Through the multi-drone collaborative shepherding environment and the MTDRL algorithm, multiple effective shepherding agents are trained and capable to achieve superior performance.

Experimental results demonstrate that our approach significantly outperforms existing methods in terms of success rate, completion time, and flight path length for both driving and collecting tasks. Additionally, trajectory visualizations illustrate effective collaboration among herders, thereby validating the enhanced efficacy of our approach. Our method is implemented and deployed in real life on two drones and four small autonomous vehicles. The evaluation results show that drones successfully complete the task of driving the small cars within 30 seconds.

II. Related Work

Solutions of Shepherding: Various researchers have developed abstract models to simulate shepherding behaviors. Reynolds et al. [20] first developed the boids model of the swarm by mimicking the flocking behavior of birds. Vo et al. [19] introduced a motion planning method based on the simulation of group behavior to guide the flock of agents through the environment populated with obstacles. Strombom et al. [3] defined two key behaviors–driving and collecting, and then demonstrated a heuristic shepherding method to achieve adaptive switching between these two behaviors. These realistic modeling of shepherding behavior is crucial for effectively transferring solutions to real-world herding problems. Furthermore, many methods are based on such modeling and subsequently design a series of rules to accomplish shepherding tasks. Lakshika et al. [21] proposed a framework featuring a specialized objective function, explicitly designed to efficiently co-evolve shepherding behaviors. Fujioka et al. [22] presented V-formation control and the other crowd control method to herd the targets to the goal area, considering the cohesion behavior of the target flock. Furthermore, Lee et al. [8] extended the rule-based method from one shepherd to multiple herders by developing a decentralized method to uphold the specific formation of shepherding agents based on the biologically-inspired control law. Concerning the uncertainty of environments, Mohanty et al. [10], [11] utilized control barrier functions to impose constraints on the velocities of the herders. However, the inherent complexity of defining exhaustive rules for every possible scenario often results in fragility in these rule-based methods, limiting their robustness and adaptability.

Machine Learning for Shepherding: The application of machine learning to shepherding has shown promising results. CK et al. [14] first presented a shepherding solution based on the SARSA algorithm and built the task environment based on the boids model [20]. Lellis et al. [18] introduced a control-tutored reinforcement learning algorithm, demonstrating that the fusion of feedback control and learning could attain complex control objectives. Zhi et al. [17] proposed a deep reinforcement learning (DRL) framework integrated with probabilistic roadmaps to guide a group of agents in obstacle-laden environments.

Advances in MTDRL have been significant. Actor-Mimic [23] derived a multi-task policy from single-task DQN expert policies, which demonstrated its effectiveness in the Atari benchmark [24]. Multi-objective reinforcement learning [25],

[26] and policy search [27], [28] were explored as extensions of Actor-Mimic. Learning multiple tasks by a shared representation was theoretically proved to be better than learning a single task [29]. Currently, there is no work on applying MTDRL to shepherding tasks.

III. Method

In this section, we first introduce the multi-drone collaborative shepherding environment and then propose the multi-task reinforcement learning algorithm for two shepherding tasks.

A. Multi-Drone Collaborative Shepherding Environment

Detailed designs for the multi-drone collaborative shepherding environment, such as tasks, entities, rules, the observation space, the action space and the reward function, are as follows:

1) Tasks: The environment encompasses two primary tasks: driving and collecting.

Driving Task: This task requires herders to guide the flock towards a designated goal area. Herders continually monitor the positions of the targets and adjust their velocities to steer the flock appropriately. A significant reward is awarded once the targets successfully reach the goal area, marking the completion of the driving task.

Collecting Task: This task involves rounding up scattered targets into a specified collection area. Similar to the driving task, herders track the positions of individual target and modify their velocities to herd these scattered targets together. Completion of the collecting task is confirmed and rewarded when all targets are gathered within the goal area.

2) Entities: The environment consists of two primary entities: M herders $\mathbb{D} = \{d_1, \ldots, d_i, \ldots, d_M\}$ and N targets $\mathbb{S} = \{s_1, \ldots, s_j, \ldots, s_N\}$. The herders serve as the shepherding agents, while the targets represent the target animals to be guided or herded. The herders and targets are both modelled as points in simulation.

3) Rules: As is shown in Fig. 1, the environment incorporates two fundamental rules: the model of targets' movement and the herder's repulsion effect towards the targets [18]. The behavior of the target flock is governed by the boids model [20], a widely recognized algorithm in computer graphics and simulations for replicating flocking behavior. The boids model is based on three primary rules:

- Separation. This rule computes a velocity v_{sep} for each target, directing it away from the flock's center of mass. It helps maintain a safe distance between the targets in the flock, thereby reducing the risk of collisions and preventing congestion.
- Alignment. This rule generates a velocity v_{align} that aligns each target's movement direction with the average velocity of nearby flock members. It ensures coordinated movement across the flock, promoting streamlined group navigation.
- Cohesion. This rule produces a velocity v_{coh} that guides each target towards the center of nearby flock members. It encourages the targets to stay together, enhancing flock unity and maintaining group compactness.

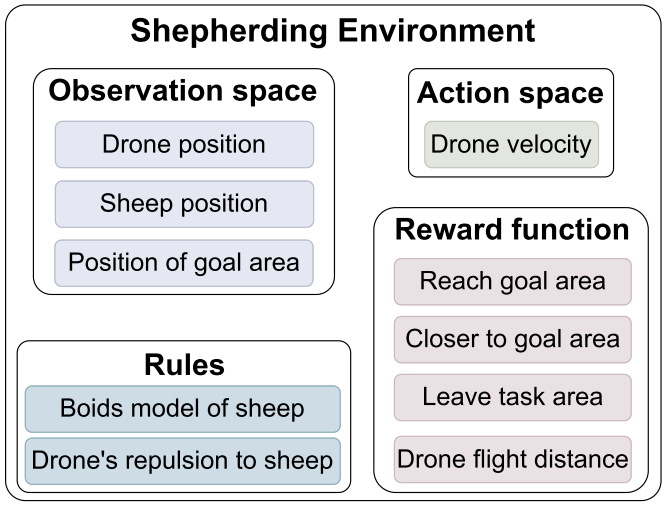


Fig. 1. Diagram illustrating the integrated framework of the multi-drone collaborative shepherding environment, detailing the components of the observation space, action space, reward function, and governing rules.

Considering a target, s_1, its velocity, v_{s_1}, is influenced by two main factors: interactions within the target flock and interactions with the herders. The velocity component produced by the former factor, denoted as v_{boids}, is a combination of three behavioral rules from the boids model:

$$v_{boids} = a_1 v_{sep} + a_2 v_{align} + a_3 v_{coh} \tag{1}$$

where a_1, a_2, a_3 are tuning hyperparameters that weight the influence of separation, alignment, and cohesion, respectively. These components are defined as follows:

$$v_{sep} = \sum_{j=2}^{N} \frac{x_{s_1} - x_{s_j}}{\|x_{s_1} - x_{s_j}\|_2^3} \cdot I(\|x_{s_1} - x_{s_j}\|_2 \leq G_1) \tag{2}$$

$$v_{align} = \frac{1}{N} \sum_{j=1}^{N} v_{s_j} \tag{3}$$

$$v_{coh} = \left(\frac{1}{N} \sum_{j=1}^{N} x_{s_j} \right) - x_{s_1} \tag{4}$$

Here, x_{s_j} and v_{s_j} represent the position and velocity of the target s_j, respectively. $\|x_{s_1} - x_{s_j}\|_2^3$ is the cube of the 2-norm of $x_{s_1} - x_{s_j}$. G_1 is the detection range for separation and $I(condition)$ is the indicator function, which equals 1 if the condition is true, otherwise 0. In other words, the separation component only affects the target s_1 from others within range G_1. To guide and herd the targets effectively, herders exert a repulsive force on the targets [18], which influences their movement:

$$v_{herder2target} = \sum_{i=1}^{M} \frac{x_{s_1} - x_{d_i}}{\|x_{s_1} - x_{d_i}\|_2^3} \tag{5}$$

where The z-axis velocity component of $v_{herder2target}$ is set to 0 directly, as the target cannot pass through the ground.

The total velocity of the target s_1 is a weighted combination of influences from both the flock and the herders:

$$v_{s_1} = a_4 v_{boids} + a_5 v_{herder2target} \tag{6}$$

where a_4 and a_5 are hyperparameters that balance the forces from the flock and the herders, respectively.

By integrating these dynamics into our multi-drone collaborative shepherding environment, we ensure realistic interactions between the herders and targets. The boids model governs the natural flocking behavior, promoting separation, alignment, and cohesion among the targets, while the repulsive force from the herders allows for precise movement control, thereby facilitating efficient and effective shepherding operations.

4) Observation Space: The observation space outlines the information each herder accesses for decision-making. The observation space is structured as a matrix with dimensions $(M + N + 1, 3)$, where M is the number of herders, N is the number of targets, and '1' represents the goal area. This matrix includes:

- Positions of herders and targets. The three-dimensional coordinates of each herder and target are provided, enabling herders to evaluate the spatial distribution of the flock and adjust their shepherding strategies accordingly. The z-axis coordinate of the target is always zero, as the target cannot pass through the ground.
- Position of the goal area. This entry in the matrix helps herders ascertain the distance and direction to the target, facilitating effective navigation and herding towards the desired destination.

This comprehensive observation space allows herders to make informed decisions based on the complete dynamics of the environment, ensuring efficient management and control of the flock.

5) Action Space: The action space specifies the actions that herders can perform. In this environment, actions are defined as the velocities of the herders on the 3D space, to better approximate the real-world control of drones. The action space is therefore represented as a matrix with dimensions $(M, 3)$, where M is the number of herders. Each entry in the matrix is a velocity vector corresponding to a herder's movement in the 3D space. This action space enables the herders to effectively manage their proximity to the targets, influence the targets' movement, and strategically herd them toward the designated goal area.

6) Reward Function: The reward function is designed to optimize the herders' shepherding performance, including incentives and penalties across different aspects:

- Task Completion Reward ($r_{complete}$). This reward is granted when herders successfully herd the targets to the goal area or gather them within a designated area.
- Boundary Penalty (r_{leave}). Imposed if any herder or target exits the task area, this penalty encourages the herders to maintain the flock within designated boundaries, ensuring the task's confinement. The episode is terminated immediately upon violation.
- Proximity Penalty (r_{closer}). This penalty assesses the change in distance between each target and the goal area over two consecutive timesteps. If a target moves away from the goal area, a penalty is applied. To discourage herders from attempting to avoid penalties for moving away by remaining stationary, a lesser penalty is also applied when herders do not move, and is smaller than the penalty for straying. Additionally, to prevent herders from manipulating the system by moving back and forth near the goal area for undue rewards, a minimal penalty is consistently applied even as the flock approach the goal area. This encourages a more stable and direct approach to herding the targets towards the goal area.
- Travel Distance Penalty (r_{flight}). This penalty, based on the sum of the distances traveled by the herders, encourages efficient movement and optimal path finding to minimize unnecessary maneuvers.

The combined reward function can be formally expressed as:

$$r = k_1 r_{complete} + k_2 r_{leave} + k_3 r_{closer} + k_4 r_{flight} \tag{7}$$

where k_1, k_2, k_3, k_4 are tuning parameters weighting each term's significance. Detailed conditions and computations for each reward component are as follows:

$$r_{complete} = \begin{cases} r_1, & \text{if all targets reach the goal area} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$r_{leave} = \begin{cases} r_2, & \text{if any object exits the task area} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

$$r_{closer} = \sum_{j=1}^{N} (r_3 - \|x_{s_j} - x_{tar}\|_2 + \|x_{s_j}^{last} - x_{tar}\|_2) \tag{10}$$

$$r_{flight} = -\sum_{i=1}^{M} \|x_{d_i} - x_{d_i}^{last}\|_2 \tag{11}$$

where r_1, r_2, r_3 are specific reward or penalty magnitudes, x_{tar} is the position of the goal area, $x_{s_j}^{last}$ and $x_{d_i}^{last}$ are the positions of targets and herders in the previous timestep, respectively.

B. Multi-Task Deep Reinforcement Learning for Shepherding

We introduce the Multi-Task Deep Deterministic Policy Gradient (MTDDPG) algorithm, specifically tailored for the dual tasks of driving and collecting in shepherding. The Deep Deterministic Policy Gradient (DDPG) [30] is an off-policy actor-critic method that synergizes the benefits of policy-based and value-based approaches. It employs a deterministic policy, which facilitates more precise action selection than stochastic policies, by using an actor network to determine the optimal policy and a critic network to estimate the value function. We incorporate a shared network architecture, as proposed in [29], to construct the MTDDPG. This approach capitalizes on multi-task learning through shared representation across the tasks of driving and collecting. By training these tasks concurrently, MTDDPG identifies and leverages common features, enhancing the transfer of learned behaviors between tasks. This synergy not only boosts performance but also enhances the generalization capabilities of the herders. Although the targets and herders' dynamics is modelled to construct the simulation environment

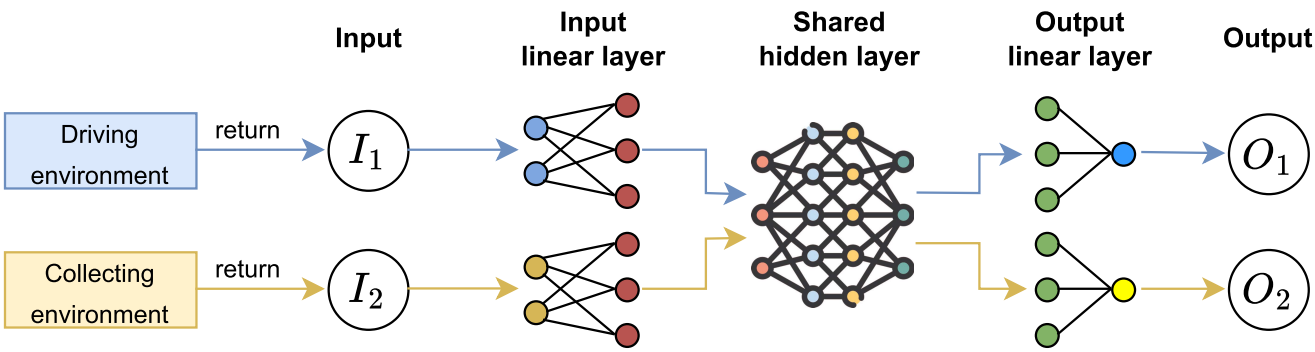


Fig. 2. Illustration of the shared neural network architecture used for dual shepherding tasks, highlighting the shared hidden layers that are common across tasks. This shared architecture facilitates efficient learning by leveraging common features in driving and collecting tasks.

necessary for training, MTDDPG algorithm is indeed model-free, allowing the agent to implicitly acquire the needed knowledge without requiring access to the underlying model during the decision-making and learning phase.

Architecture Details: The basic architecture of sharing neural network is shown in Fig. 2. Driving environment refers to the multi-drone collaborative shepherding environment we constructed for executing the driving task, while collecting environment refers to the environment for performing the collection task. The network inputs provided by different tasks pass through different input and output layers, as well as the same shared layers. This architecture allows the networks to process inputs and outputs for both tasks effectively, enabling the herders to execute complex shepherding maneuvers with greater efficiency. The MTDDPG employs both actor and critic networks, which follow above basic shared network architecture. Their detailed networks are as follows:

- Actor Network. This network utilizes several fully connected layers with ReLU activation functions to process the current environmental state. The output layer features the sigmoid activation function to ensure that the actions generated are within a valid range, making it suitable for continuous action spaces.
- Critic Network. Comprising fully connected layers with ReLU activation functions, the critic network integrates both the current state and the action taken to estimate the state-action value function, or Q-value. Its output layer delivers a single value estimation, providing feedback on the potential impact of the chosen actions.
- Target Networks. To promote stability in the training process, the MTDDPG employs target networks, which are periodically updated copies of the actor and critic networks using a soft update technique. These networks are crucial for generating stable target Q-values, which help in minimizing overestimation during training and enhancing the reliability of learning updates.

By leveraging the MTDDPG algorithm and its detailed network architecture, we enhance the herders' shepherding abilities in both driving and collecting tasks. This setup ensures that the herders coordinate their actions effectively, achieving optimal performance and successful task completion.

IV. Implementation

This section outlines the environment parameters and details the training and testing processes utilized in our study.

A. Environment Setting

The task environment is a 500 m x 500 m x 10 m area. The goal area, defined as a circle, has a randomly initialized center with a radius of 45 meters. Similarly, the starting area for driving tasks is also a circle with a randomly initialized center, but with a radius of 50 meters.

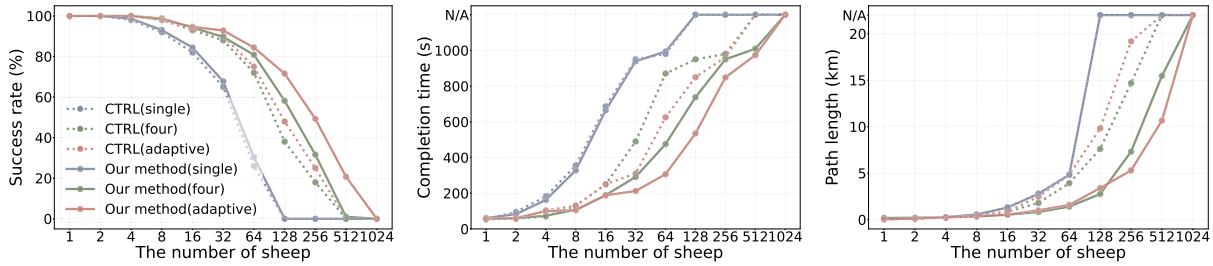
Boids Parameters: The default parameters for the boids rules in our environment are as follows: separation coefficient $a_1 = 100$, alignment coefficient $a_2 = 0.05$, cohesion coefficient $a_3 = 0.01$, flock influence coefficient $a_4 = 0.5$, herder influence coefficient $a_5 = 600$, and separation range $G_1 = 5$. The perturbations in the target's model at test time are considered to ensure that our method is robust and not overly sensitive to the target's model during training. During the testing phase, at the beginning of each episode, all boids parameters in the environment are set to the default parameters plus Gaussian noise. The noise for each parameter follows a Gaussian distribution with a mean of zero and a standard deviation equal to one-fifth of the default parameter.

Reward Parameters: For the collecting task, the default reward settings are $r_1 = 50$, $r_2 = -200$, $r_3 = -10$ with weighting factors $k_1 = 0.1$, $k_2 = 1$, $k_3 = 0.05$, and $k_4 = 2$. For the driving task, the parameters are set as $r_1 = 100$, $r_2 = -200$, $r_3 = -10$, $k_1 = 0.5$, $k_2 = 1$, $k_3 = 0.02$ and $k_4 = 2$. A global scaling factor $k_0 = 0.9$ is used to adjust the absolute value of the reward, stabilizing reward volatility across samples.

Other Parameters: The maximum speed of the herder is five meters per second, and the maximum speed of the target is three meters per second. We assume that for each step in the environment, one second has passed.

B. Training and Testing

Training: The model is trained using one NVIDIA RTX 2080ti GPU [31] on Python 3.8 with Pytorch 1.13 [32]. Depending on the experiment, different configurations of $< M, N >$ are used.



(a) Success rate of comparison methods on driving.

(b) Average completion time of comparison methods on driving.

(c) Average path length of comparison methods on driving.

(d) Success rate of comparison methods on collecting.

(e) Average completion time of comparison methods on collecting.

(f) Average path length of comparison methods on collecting.

Fig. 3. Overall performance of our method compared with CTRL. N/A indicates a success rate of zero for this group, so this group has no average completion time and average flight path length.

For each configuration, we accumulate 1 million training steps to develop the policy, which spends about four hours. The batch size for training is set to 256.

Testing: All methods are evaluated on a server equipped with two Intel @2.20 GHz Xeon Silver 4210 CPUs [33] and one NVIDIA RTX 2080ti GPU. Sensing and actuation noise are added during evaluation [16].

V. Experiments

This section outlines the comparative evaluation of our method's performance, focusing on multi-drone coordination and real-world performance.

A. Overall Performance

We assess our method's effectiveness in shepherding against the state-of-the-art (SOTA) solution.

1) Comparison Method: We benchmark against the control-tutored reinforcement learning (CTRL) algorithm [18], which alternates between control-tutor policy suggestions and decisions made by the reinforcement learning policy. CTRL also addresses the shepherding problem and has completed training in its own constructed environment. We compare the performance of the strategies trained in their respective environments between CTRL and our method under the same scenario.

2) Setup Variability:

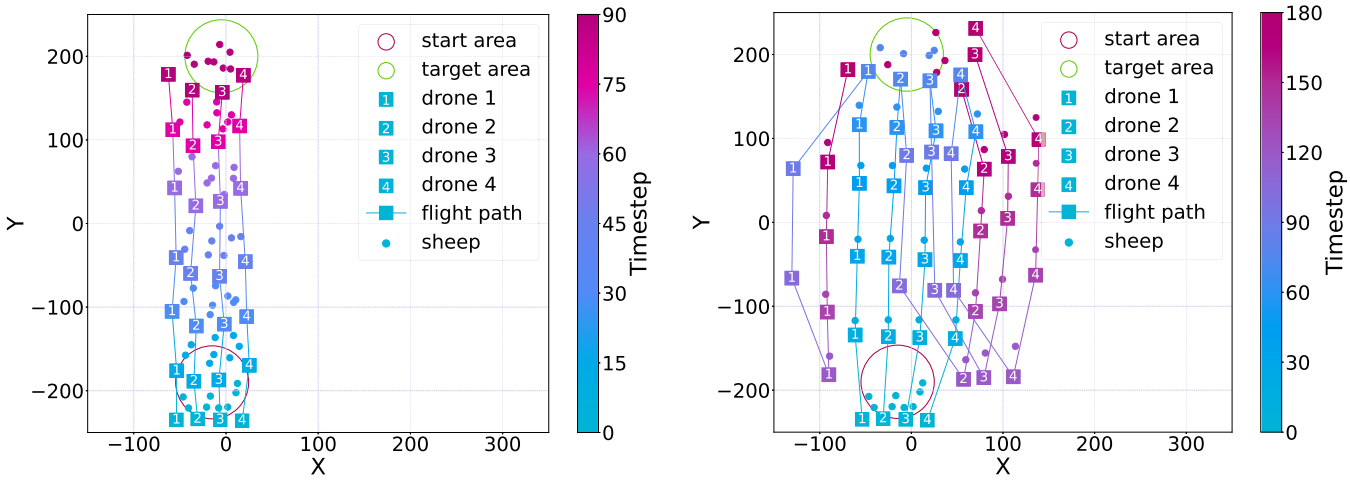
- The Number of Targets. Tests are conducted with varying flock sizes S, specifically 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, and 1024 targets.
- The Number of Herders. Herders are tested in configurations to evaluate scalability: single herder, four herders, adaptive herder count of $S/2^{\lfloor log_2 S/2 \rfloor}$.

3) Performance Metrics: For each $< M, N >$ configuration under each task, each method is executed for 100 episodes, and the following metrics are calculated based on these episodes:

- Success Rate. Defined as the ratio of successful episodes to total episodes. An episode is considered successful if the herders guide the entire flock to the goal area or collect all dispersed targets.
- Average Completion Time. The mean time taken across successful episodes to complete the shepherding tasks, reflecting the method's efficiency. The duration of one step is one second.
- Average Path Length. Calculated as the average total distance traveled by the herders during successful episodes, indicating the effectiveness of the herder navigation strategy.

4) Evaluation Results: Fig. 3 presents the comparative performance of our method and the CTRL algorithm across the driving and collecting tasks.

Single vs. Multiple Herders: With a single herder, the performance of both methods is similar. However, in scenarios utilizing multiple herders, our method consistently outperforms CTRL in both tasks. Notably, as the number of targets increases, although success rates decline for both methods, our method sustains higher success rates than CTRL. This underscores the efficacy of our multi-drone coordination, a feature lacking in CTRL, particularly evident as our method remains effective with up to 512 targets, whereas CTRL's success rate diminishes to zero in such large-scale scenarios.



(a) The trajectory of our method.

(b) The trajectory of CTRL.

Fig. 4. In the scenario of four herders and eight targets, the trajectory of each comparative method on XY plane on the driving task. Different colors represent the herders or targets at different timesteps.

Completion Time and Path Length: Across various tasks and herder configurations, our method achieves competitive or even shorter completion time compared to CTRL, demonstrating its efficiency. Furthermore, our method tends to result in shorter path lengths relative to CTRL, indicating more efficient herder navigation and optimized task execution.

B. Multi-Drone Coordination Study

This study highlights the superior coordination capabilities of our method compared to the CTRL algorithm, which lacks explicit coordination mechanisms. With Fig. 4, we illustrate this through a visual analysis of previous experimental data from the driving task involving four herders and eight targets.

For our method, the four herders strategically form a diamond formation, enabling them to collaboratively and efficiently herd all eight targets directly to the goal area. Conversely, for the CTRL, each herder independently herds four targets towards the goal area, then returns to collect the remaining four. CTRL's performance arises from its strategy of having a herder drive one target into the goal area at a time. This sequential approach reflects a lack of coordination, leading to increased flight time and higher resource consumption.

The visual data clearly show that our method's coordination mechanism significantly enhances efficiency. By coordination between herders, our herders minimize the overall herding time and resource use, contrasting sharply with the disjointed and inefficient process observed with CTRL herders. This visual evidence underscores our method's effectiveness in executing multi-drone collaboration and optimizing the completion of shepherding tasks.

C. Real-World Experiment

To demonstrate the potential applicability of our proposed method in real-world shepherding tasks, we conduct the hardware experiment using real drones and small autonomous vehicles in an indoor environment.

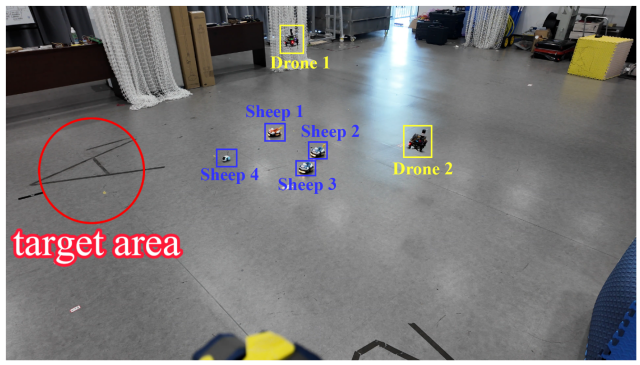


Fig. 5. A snapshot from the real-world experiment.

The experiments are conducted within a controlled environment, specifically a classroom space measuring 10 m x 10 m x 3 m. Four ground vehicles that serve as targets, and two drones equipped with Jetson NX [34], are utilized for real-world experiments. The primary task involves driving, where two drones are required to guide the four vehicles from a designated starting area to a goal area. The starting area and the goal area are marked on the floor, with a distance of more than 8 meters between them. Drones' policy is first trained in a simulated environment and subsequently deployed on the real-world drones.

Fig. 5 illustrates a snapshot from the experimental process. As a result, the two drones are able to drive the four vehicles to the goal area in just 30 seconds. A demonstration video of the experiment, which showcases the successful completion of the task, is included in the multimedia attachments submitted with this paper.

VI. Conclusion

In this paper, we introduce a robust solution to the challenges of drone automated shepherding by utilizing the multi-drone collaboration and the multi-task deep reinforcement learning algorithm. Our experimental setup includes a variety of

scenarios with different numbers of sheep and drones, demonstrating that our approach consistently surpasses the CTRL baseline in terms of success rates, completion time, and flight path length. These results emphasize the advantages of our multi-drone coordination, which significantly enhances operational efficiency. In the real-world experiments, our method successfully facilitates two drones in guiding four small autonomous vehicles to the goal area within 30 seconds, thereby demonstrating its potential applicability in real shepherding tasks. Looking forward, we aim to explore the scalability of our method to handle larger flocks and more diverse environments. The versatility of our techniques holds promise for broader applications, such as crowd control and search and rescue operations, potentially revolutionizing the field of multi-robot systems.

REFERENCES

[1] 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.

[2] 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.

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

[4] F. Leng, C. M. Tan, and M. Pecht, "Effect of temperature on the aging rate of Li ion battery operating above room temperature," *Sci. Rep.*, vol. 5, no. 1, 2015, Art. no. 12967.

[5] S. J. Kim, G. J. Lim, and J. Cho, "Drone flight scheduling under uncertainty on battery duration and air temperature," *Comput. Ind. Eng.*, vol. 117, pp. 291–302, 2018.

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

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

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

[9] J. Grover, N. Mohanty, C. Liu, W. Luo, and K. Sycara, "Noncooperative herding with control barrier functions: Theory and experiments," in *Proc. IEEE 61st Conf. Decis. Control*, 2022, pp. 80–86.

[10] N. Mohanty, J. Grover, C. Liu, and K. Sycara, "Distributed multirobot control for non-cooperative herding," in *Proc. Int. Symp. Distrib. Auton. Robotic Syst.*, 2022, pp. 317–332.

[11] J. S. Grover, "System identification and control of multiagent systems through interactions," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, 2023.

[12] M. Baumann and H. Buning, "Learning shepherding behavior." Ph.D. dissertation, University of Paderborn, Paderborn, Germany, 2016.

[13] 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.

[14] C. K. Go, B. Lao, J. Yoshimoto, and K. Ikeda, "A reinforcement learning approach to the shepherding task using SARSA," in *Proc. Int. Joint Conf. Neural Netw.*, 2016, pp. 3833–3836.

[15] H. T. Nguyen et al., "Continuous deep hierarchical reinforcement learning for ground-air swarm shepherding," 2020, *arXiv:2004.11543*.

[16] E. Debie, H. Singh, S. Elsayed, A. Perry, R. Hunjet, and H. Abbass, "A neuro-evolution approach to shepherding swarm guidance in the face of uncertainty," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2021, pp. 2634–2641.

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

[18] F. D. Lellis, F. Auletta, G. Russo, P. d. Lellis, and M. d. Bernardo, "An application of control-tutored reinforcement learning to the herding problem," in *Proc. 17th Int. Workshop Cellular Nanoscale Netw. Appl.*, IEEE, 2021, pp. 1–4.

[19] C. Vo, J. F. Harrison, and J.-M. Lien, "Behavior-based motion planning for group control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 3768–3773.

[20] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proc. 14th Annu. Conf. Comput. Graph. Interactive Techn.*, 1987, pp. 25–34.

[21] E. Lakshika, M. Barlow, and A. Easton, "Co-evolving semi-competitive interactions of sheepdog herding behaviors utilizing a simple rule-based multi agent framework," in *Proc. IEEE Symp. Artif. Life*, 2013, pp. 82–89.

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

[23] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," 2015, *arXiv:1511.06342*.

[24] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, 2013.

[25] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1312–1320.

[26] M. Andrychowicz et al., "Hindsight experience replay," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5055–5065.

[27] Z. Yang, K. E. Merrick, H. A. Abbass, and L. Jin, "Multi-task deep reinforcement learning for continuous action control," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 3301–3307.

[28] Y. Teh et al., "Distral: Robust multitask reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4499–4509.

[29] C. D'Eramo, D. Tateo, A. Bonarini, M. Restelli, and J. Peters, "Sharing knowledge in multi-task deep reinforcement learning," 2024, *arXiv:2401.09561*.

[30] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[31] Titan RTX, 2020. [Online]. Available: https://www.nvidia.cn/deep-learning-ai/products/titan-rtx/

[32] "PyTorch 1.13 release, including beta versions of functorch and improved support for Apple's new M1 chips," 2022. [Online]. Available: https://pytorch.org/blog/PyTorch-1.13-release/

[33] "Intel Xeon Silver 4210 CPU," 2020. [Online]. Available: https://www.intel.cn/content/www/cn/zh/products/sku/193384/intel-xeon-silver-4210-processor-13-75m-cache-2-20-ghz/specifications.html

[34] Nvidia Jetson Xavier NX, 2024. [Online]. Available: https://www.nvidia.cn/autonomous-machines/embedded-systems/jetson-xavier-nx/