IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received February 28, 2022, accepted April 26, 2022, date of publication May 3, 2022, date of current version May 11, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3172492

Controllable Swarm Animation Using Deep Reinforcement Learning With a Rule-Based Action Generator

ZONG-SHENG WANG[1], **CHANG GEUN SONG**[1], **JUNG LEE**[1], **JONG-HYUN KIM**[2], **AND SUN-JEONG KIM**[1]
[1]Department of Convergence Software, Hallym University, Chuncheon 24252, South Korea
[2]School of Software Application, Kangnam University, Yongin 16979, South Korea

Corresponding author: Sun-Jeong Kim (sunkim@hallym.ac.kr)

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology, under Grant NRF-2020R1A2C101519513; in part by the Hallym University Research Fund under Grant HRF-202108-002, and in part by the R&D Program for Forest Science Technology under Project 2021390A00-2123-0105 provided by the Korea Forest Service (Korea Forestry Promotion Institute).

ABSTRACT The swarm behavior in nature is a fascinating and complex phenomenon that has been studied extensively for decades. Visually natural swarm animation can be produced by the state-of-the-art rule-based method; however, it still suffers from the drawbacks of low control accuracy and instability in swarm behavior quality when controlled by the user. This study proposes a deep reinforcement learning (DRL) based approach to generate swarm animation that reacts to real-time user control with high quality. A rule-based action generator (RAG) adapted to the actor-critic DRL method is presented to enhance DRL's action exploration strategy. Various practical dynamic reward functions are also designed for DRL to train agents by rewarding swarm behaviors and penalizing misbehavior. The user controls the swarm by interacting with the swarm's leader agent, for example by directly changing its speed or orientation, or by specifying a path consisting of waypoints. The second aim of this study is to improve the scalability of the trained policy. This study introduces a new state observation quantity of DRL called the embedded features of swarm (EFS) for allowing the trained policy scaling to a more extensive system than it has been trained on. In the experiments, four different scenarios have been designed to evaluate the control accuracy and quality of the generated swarm behavior by metrics and visualization. Additionally, the experiment has compared the performance of the proposed dynamic reward functions with fixed reward functions. Experimental results show that the proposed approach outperforms state-of-the-art methods in terms of swarm behavior quality and control accuracy. Moreover, the proposed dynamic reward functions are more effective than the existing reward functions.

INDEX TERMS Computer graphics, swarm animation, deep reinforcement learning, actor-critic, path planning.

I. INTRODUCTION

Swarm behaviors, as commonly seen as a flock of birds, a herd of land animals, a school of fish, or even human crowds, are complex yet fascinating phenomena in nature and society. To investigate the swarm phenomena involves interdisciplinary subjects such as biology, sociology, psychology, physics, and computer science. Generally, swarms are composed of a large number of simple, individual, and homogeneous agents [1]. From a global perspective, swarms performance seems to be collective and self-organized, but their behaviors are stochastic from the perspective of individuals. Recently, numerous studies have been devoted to modeling swarm behaviors [2] which provide contributions to robotics [3], traffic simulation [4]–[6], network communication [7], navigation [8], [9], precision agriculture [10], and other fields. The stochastic and convergence properties of swarm behaviors are also widely used in designing meta-heuristic optimization algorithms [11]–[17] to solve NP-hard problems [18]–[20].

The associate editor coordinating the review of this manuscript and approving it for publication was Songwen Pei.

Advancements in the field of digital entertainment have evolved significantly in recent years. Generating controllable swarm animations [21]–[24], as a practical and challenging topic, has been studied for years. Due to the limitations of the computer systems, early digital contents had relatively been rough and straightforward. As computer graphics technology and hardware have advanced, visual effects have become more realistic [25], [26], and game content has grown more complicated. The study of swarm simulation has the potential to provide high-quality content for digital entertainment and social simulation. Therefore, modeling and simulating natural and social phenomena, such as swarm behaviors in animal groups or human crowd behavior, is important.

This study focuses on generating controllable and visually realistic swarm animations. The bulk of state-of-the-art studies [27]–[29] use rule-based methods. However, those methods still suffer from the drawbacks of low control accuracy and instability in swarm behavior quality, when they are controlled by the user in real time. Simultaneously, a surge of interest in data-driven animations [30]–[34] has been rekindled along with the development in deep learning technology. Thus, this study proposes a deep reinforcement learning (DRL) method which generates swarm animation that reacts naturally to real-time user control. DRL is basically a trial-and-error technique that gains experience by driving the agent to continuously interact with the environment. The challenge is that DRL requires extensive action explorations and more episodes for training agents in swarms since swarms are multi-agent systems with various interactions. To address this issue, this study proposes a rule-based action generator (RAG) to improve DRL's action exploration strategy. Another objective of this study is to scale the trained policy to function on a more extensive system than the original ones on which it has been trained. In simulations, the user controls the swarm of trained agents by interacting with the leader agent of the swarm. The experimental results show that the proposed method outperforms several state-of-the-art methods in terms of control accuracy and swarm behavior quality.

The main contributions of this study include the follows:

- A novel rule-based action generator (RAG) is proposed to improve the quality of generated swarm behaviors by enhancing DRL's action exploration strategy.
- A new state observation quantity of DRL called the embedded features of swarm (EFS), is introduced to improve the scalability of the swarm created by the trained policy.
- Several dynamic reward functions using quadratic or exponential function properties are designed for DRL to train agents by rewarding swarm behaviors and penalizing misbehaviors.

The rest of this paper has been organized in the following way. Section 2 provides a review of the literature and relevant work. A detailed description of the proposed method is presented in Section 3. Section 4 is devoted to discussing the experimental results. The conclusion and future works are summarized in the last section.

II. RELATED WORK

Many papers have been published on the swarming phenomenon [35]–[40] in nature since the 1980s. Following the pioneering works of Reynolds [41] and Vicsek *et al.* [42] in the late 1980s and early 1990s, numerous studies have been devoted to modeling swarm behaviors [2]. The majority of studies can be categorized as rule-based methods [27]–[29], [41]–[44] which prescribe a variety of physical [27], [29] or kinematic rules [28] for positioning agents to preserve swarm behaviors. Moreover, as artificial intelligence has advanced in recent years, data-driven methods for swarm simulation have been considered [32], [45]–[49].

A. RULE-BASED METHODS

Rule-based simulation methods are modeling approaches that refer to using a set of rules indirectly specifying a mathematical model [50], [51]. To model swarm behaviors, earlier studies [41], [42] have proposed rules that constrain the behavior of individuals in a swarm. Reynolds [41] proposed that swarm behaviors follow three primitive rules [41]: *separation* refers to the prevention of static collisions between agents in the swarm; *alignment* specifies velocity matching of agents to others in the swarm; *cohesion* indicates the propensity of agents to gravitate toward the swarm's center of mass (COM). Later, the collection of potential steering behaviors is expanded to include goal-seeking, obstacle avoidance, direction-finding, and fleeing [52]. Vicsek *et al.* [42] presented the idea of self-propelled particles (SPP) in 1995 as a special case of Reynolds' model [41]. An SPP swarm is modeled by a set of particles and the heading of each particle is updated using a local rule taking the average of its own heading plus the headings of its "neighbors". The work of Tu *et al.* [53] simulated the underwater environment and realistically emulates the appearance, movement, and behavior of fishes driven by their state and intentions. Zaera *et al.* [54] attempted but failed to create the Newtonian kinematics-based controller for three-dimensional schooling behavior using inertia and drag. Ward *et al.* [43] are more successful than Zaera *et al.* [54] by proposing a fitness function dependent on each agent's virtual energy levels, as it was challenging to explicitly encode entire swarm behaviors in the fitness function. Furthermore, a tremendous amount of studies [55]–[62] have demonstrated the efficiency of rule-based methods.

This work focuses on studies [21]–[24], [63]–[67] related to creating swarm animations with these rule-based methods. Anderson *et al.* [68] investigated an iterative sampling approach for generating group animations of predefined formations. Nonetheless, the approach was considered to be quite computationally expensive for real-time rendering, and they did not take into account managing obstacles avoidance during certain phases. Xu *et al.* [69] proposed a shape-constrained swarm animation system to enforce static and deforming shape constraints on the spatial distribution of a swarm. Klotsman *et al.* [70] provided a biologically motivated technique for modeling and animating bird flocks in flight with line formations, which produces plausible and realistic-looking flock animations. Wang *et al.* [71] proposed a flock morphing technique for creating special morphing effects between two arbitrary 3D objects. Zheng *et al.* [72] adopted geometric constraint idea for smooth formation animation of regulated crowds. Wolinski *et al.* [73] presented a method to compute optimal parameters for rule-based or physically based multi-agent simulation algorithms. As the most recently rule-based methods for producing swarm behaviors, Chen *et al.* [27] defined a distance-based rule to sort agents by three separate sphere zones, and presented a flock morphing method of flying insect swarms with predefined shape constraints. On the basis of the work of Chen *et al.*, In this study, a rule-based action generator (RAG) was designed for deep reinforcement learning (see Section III-C3).

B. DATA-DRIVEN METHODS

Along with the development in deep learning technology, data-driven algorithms have been introduced into computer graphics for rendering [74]–[77], simulation [78]–[80], and geometry processing [81]–[89]. In recent studies, there has been a vast amount of renewed interest in data-driven animations [30]–[34]. Several studies have learned patterns from domain-specific data (such as developed datasets or real-world data) to simulate complex phenomena. Simultaneously, deep reinforcement learning is becoming a trendy framework for generating complex animations.

1) DEEP REINFORCEMENT LEARNING (DRL)

Reinforcement learning (RL) is an algorithm to solve a Markov decision process (MDP), which is a mathematical formulation on sequential decision-making problems. Deep reinforcement learning (DRL) [90]–[94] offers a deep neural networks-based mechanism for the agent to make behavioral decisions by interacting with the environment in lack of reference data. DRL was commonly used for robotics in the early studies. Works from [95]–[98] have shown success in controlling robots or agents with different morphology. DRL methods have also been successfully applied to games in a discrete virtual environment [99]–[101]. Recent studies of producing animations by DRL focus on crowd simulation [102] and character motion control [103].

This study is motivated by a recent work of Lee *et al.* [31], which provided a deep reinforcement learning approach with a fixed reward function for navigating crowds in various complex scenarios. Crowd simulation refers to a process of simulating the movement of numerous characters or agents [104]. The key capability of crowd simulation is agents should reach goals without colliding with other agents or obstacles. However, the work of Lee *et al.* suffered from poor scalability and paid more attention to agents' collision avoidances rather than gathering agents in formation. In this paper, several practical dynamic reward functions instead of fixed reward functions were presented to enhance gathering agents in formation. To improve scalability, the embedded features of swarm (EFS) were introduced as a state observation of DRL. Research into crowd simulations has a long history. Macroscopic crowd simulation algorithms [105], [106] animate crowds at the global level, aiming to capture statistical quantities such as flows or densities. In contrast, microscopic algorithms [107]–[109] model interactions between individual pedestrians, with the emergence of movement patterns at the crowd level. Fridman *et al.* [110] adopted social comparison theory in their crowd behavioral model to account for the various social factors influencing human behavior. Wolinski *et al.* [109] presented a local interaction algorithm with a new context-aware, probabilistic motion prediction model to improve the quality of crowd simulations. Karamouzas *et al.* [111] proposed an optimization-based integration scheme for implicit integration of physics-based multi-agent animation. To date, several studies on crowd simulation have successfully applied deep reinforcement learning [112]–[115]. Chen *et al.* [116] brought attention mechanisms into DRL to train robots with swarm behaviors. Wang *et al.* [115] presented to apply DRL to the path planning-based crowd simulation.

In recent years, there has been a surge of interest in controlling the characters' motion [33], [117] by DRL. Park *et al.* [117] combined kinematic controllers with DRL to produce a responsive controller for biped agents. Chentanez *et al.* [118] proposed a deep reinforcement learning method that learns to control articulated humanoid bodies to imitate given target motions closely when simulated in a physics simulator. The work from Luo *et al.* [33] presented a motion synthesis based on imitation learning and introduces Generative Adversarial Networks (GAN) to adapt high-level controls. The work from Peng *et al.* [119] adapted DRL to learn robust control policies capable of imitating example motion clips, allowing the trained characters to react intelligently in interactive environments. Clegg *et al.* [120] introduced a model-free deep reinforcement learning method to automatically discovering robust dressing control policies by designing an appropriate input state space and a reward function. Moreover, interactively controlling physics-based characters in real-time is attainable by combining a motion matching technique and DRL-based control policy [121].

2) LEARNING FROM DOMAIN-SPECIFIC DATA

Research on producing animations by learning domain-specified datasets has been studied widely in recent years [122]. Xiang *et al.* [32] presented a data-driven framework, FASTSWARM, to model complex behaviors of flying insects based on real-world data and simulate plausible animations of flying insect swarms. Lee *et al.* [45] presented a crowd simulation method which use an agent model generated from real-world observations. Chao *et al.* [123] applied characteristics of drivers from an empirical video to an agent-based model. Boatright *et al.* [124] classified the contexts and learn the characteristics from a dataset. Charalambous *et al.* [125] presented a real-time synthesis method for crowd steering behaviors with the temporal perception pattern. Du *et al.* [126] investigated different ways of representing joint angles (Euler angle, quaternion, and exponential map) from motion capture data, concluding that Quaternion representation is more appealing for statistically modeling motion data. Gopalakrishnan *et al.* [127] presented a Gated Recurrent Unit (GRU)-based method for predicting and synthesizing human motion, and consider alternative metrics and human evaluation to deal with the uncertainty of the task. Kim *et al.* [128] presented a layered volumetric human body model composed of a data-driven inner layer and a physics-based external layer to produce realistic skin deformation due to interactions with the environment. In addition, the data-driven approaches are also widely used in traffic simulation [4]–[6], fluid simulation [129], [130], etc.

III. METHODOLOGY

The goal of this study is to create a data-driven framework that produces swarm behaviors and natural movements while following high-level user controls. This section describes the proposed methodology. The first part provides an overview of the proposed approach, followed by details about the training process, and eventually the trained policy is applied to the simulation phase.

A. METHOD OVERVIEW

The overview of the proposed approach is depicted in Figure 1, which contains the following main parts:

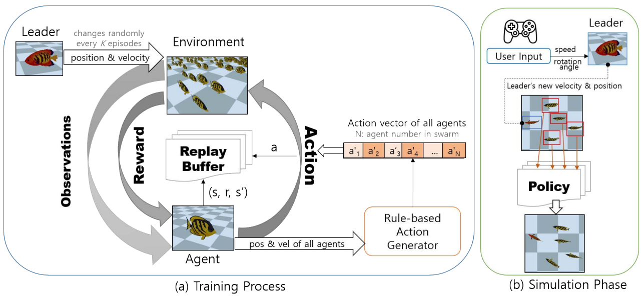


FIGURE 1. **Overview of the proposed method.**

Training process is driven by deep reinforcement learning (DRL). This study presents a rule-based action generator (RAG) to build an initial action vector of DRL for producing better swarm behaviors. This work also includes designing a series of reward functions that enable the trained policy to preserve swarm behaviors, as well as introducing embedded features of swarm (EFS) as quantities of state observations to improve the trained policy's scalability.

Simulation phase utilizes the trained policy to make agents of the swarm react to changes in the user-controlled leader agent.

B. SIMULATION PHASE

The movement of an agent in swarm is determined by its speed u and orientation (heading direction) q, where $u \in \mathbb{R}$ and $q \in \mathbb{R}^3$. During the simulation phase, the user controls the leader agent by adjusting its orientation and speed.

$$\begin{aligned} u' &= u + \Delta u \cdot h \\ q' &= q + \Delta q \cdot h \end{aligned} \tag{1}$$

where h represents the holding duration of press buttons, u' stands the new speed of the leader agent while q' indicates its new orientation (Euler angles). Moreover, Δu and Δq denote the minimum change of speed and orientation, respectively. At each time-step t, every agent makes a decision $a = \{\nu, \omega\}$ according to the trained policy.

$$\begin{aligned} u^i_{t+1} &= u^i_t + \nu_i \\ q^i_{t+1} &= q^i_t + \omega_i \end{aligned} \tag{2}$$

where ν denotes a change in speed and ω denotes a change in orientation.

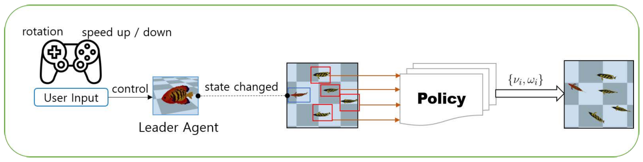


FIGURE 2. **Simulation phrase.**

C. TRAINING PROCESS

1) DEEP REINFORCEMENT LEARNING (DRL)

Generally, DRL is characterized as Markov decision process (MDP) [131] to address sequential decision-making problems. MDP is represented as $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is a set of environment's states, $\mathcal{A}$ is a collection of agent's actions (decisions), and $\mathcal{P}(s, a, s')$ is the probability of transitioning to the next state s' given the current state s and the action a. $\mathcal{R}(s, a, s')$ is a reward function that indicates the attractiveness of the agent state, and $\gamma \in [0, 1]$ is a discount factor that prevents the total of rewards from approaching infinite [132]. DRL finds an optimal policy $\pi(s)$ that maximizes the expectation on cumulative rewards $\eta(\pi)$.

$$\eta(\pi) = E_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right] \tag{3}$$

where $s_t \sim \mathcal{P}(s_{t-1}, a_t, s_t)$, $a_t \sim \pi(s_t)$, and $r_t = \mathcal{R}(s_{t-1}, a_t, s_t)$. The essence of DRL is the development of a value function, which is an approximation on $\eta(\pi)$. A state-action value function $Q_\pi(s, a)$ is considered to provide the cumulative reward when taking action a according to policy π. An appropriate value function should satisfy the Bellman optimality equation.

$$Q(s, a) = \mathcal{R}(s, a, s') + \gamma \max_{a' \in A'} Q(s', a') \tag{4}$$

where A' represents all potential actions in the next state s'.

This study utilizes an actor-critic method that is similar to Deep Deterministic Policy Gradient (DDPG) [133]. The benefits of DDPG include the reduction in training time, improved stability of the training process, and reduced sensitivity to hyperparameter changes. It combines Deterministic Policy Gradient (DPG) [134] with Deep Q-Network (DQN) [135]. DDPG extends DQN to continuous space with the actor-critic framework while learning a deterministic policy. The actor (policy) and the critic (value function) are represented as deep neural networks $\pi(s|\theta)$ and $Q(s, a|\phi)$, where θ and ϕ are parameter values of each network, respectively. The critic network Q evaluates state and action pairs, and it is learned by minimizing the mean-squared Bellman error (MSBE) loss L that measures how much Q satisfies the Bellman equation.

$$L = \frac{1}{M}\sum_{i}^{M}(r_i + \gamma Q(s_i', \pi(s_i')|\bar{\phi}) - Q(s_i, a_i|\phi))^2 \tag{5}$$

where γ is a discount factor, $\bar{\phi}$ is fixed parameter of the target network which is updated periodically to the current parameter to stabilize the learning. The actor network π decides what to do based on current state, and it is learned by the deterministic policy gradient ∇J.

$$\nabla J = \frac{1}{M}\sum_{i}^{M}(\nabla_a Q(s, a|\phi)|_{s=s_i, a=\pi(s_i)}\nabla_\theta \pi(s|\theta)|_{s=s_i}) \tag{6}$$

where ∇_a, ∇_θ are partial derivatives with respect to a, θ, respectively. A sketch of DDPG is illustrated in Figure 3. The proposed training process is described in Algorithm 1, and the structure of neural networks is described in detail in the next section (see in Section IV-A1).

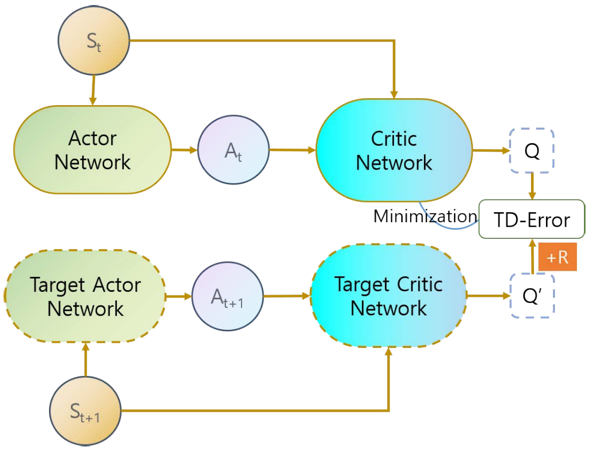


FIGURE 3. An overview of deep deterministic policy gradient.

DDPG is an off-policy model-free actor-critic method that stabilizes learning through experience replay and the frozen target network. Experience replay lets online reinforcement learning agents store and reuse previous experiences. The core idea of the proposed method is to initialize the experience replay buffer with action vectors generated by a rule-based action generator, and keep them until the leader agent is changed.

Algorithm 1 DDPG With RAG

```
1:  Initialize critic network $Q(s, a|\phi)$, actor network $\pi(s|\theta)$
    with parameters $\phi$ and $\theta$
2:  Initialize target network $Q'(s, a|\phi')$ and $\pi'(s|\theta')$ with
    parameters $\phi' \leftarrow \phi, \theta' \leftarrow \theta$
3:  Initialize replay buffer $E$ with size $S$, updating rate of the
    target network $\lambda$
4:  for episode = 1, Z do
5:      Initialize a random noise process $\Psi$ for action
        exploration
6:      Observe initial state $s_1$
7:      if episode mod K == 1 then
8:          Update leader agent's speed and orientation
            randomly
9:          Update all protection flags to false in $E$
10:     end if
11:     for t = 1, M do
12:         if episode mod K == 1 then
13:             Select action $a_t$ from Rule-based Action Genera-
                tor in Section III-C3
14:             Set the protection flag $f_t$ to true for preventing
                overwrite in $E$
15:         else
16:             Select action $a_t = \pi(s_t|\theta) + \Psi_t$ according to the
                policy and exploration noise
17:             Set the protection flag $f_t$ to false
18:         end if
19:         Execute action $a_t$ and obtain reward $r_t$ and observe
            new state $s_{t+1}$
20:         Store transition $(s_t, a_t, r_t, s_{t+1}, f_t)$ in replay
            buffer $E$
21:         Sample a random minibatch of $H$ transitions
            $(s_i, a_i, r_i, s_{i+1})$ from $E$
22:         Update critic by minimizing the loss in Eq. 5
23:         Update actor policy using Eq. 6
24:         Update the target networks:
                $\phi' \leftarrow \lambda\phi + (1-\lambda)\phi'$
                $\theta' \leftarrow \lambda\theta + (1-\lambda)\theta'$
25:     end for
26: end for
```

2) DATA COLLECTION

DRL is essentially a trial-and-error technique that gains experience by driving the agent to continuously interact with the environment. In this study, the trainer randomly assigns a new velocity and location to the leader agent every K episode. The follower agents keep exploring the action space with the same leader agent for K episodes.

The initial condition of follower agents in each episode is identical. In the training process, follower agents have two exploration policies for generating actions: one is a noise-based approach provided by the original DDPG [133], and the other is the proposed rule-based action generator (RAG) which is discussed in Section III-C3. During each episode, the trainer collects the follower agents' state observations and actions at each time step Δt. The collected experiences are then reformed into new trajectories, each of which is composed of agent-independent transitions and can be utilized by any agent during simulation. Lastly, the reformed trajectories are stored in the experience replay buffer. A new trajectory τ_i^k indicates the i-th agent's sequential actions in the k-th episode, and can be represented as follow:

$$\tau_i^k = \{s_t^i, a_t^i, r_k^{a_t^i}, s_{t+1}^i\}, for\ i = \{1, 2, \ldots N\},$$
$$t = \{1, 2, \ldots, M\}, k = \{1, 2, \ldots, K\} \quad (7)$$

where s_t^i and s_{j+1}^i represent the current and next state observation, a_t^i is the action taken by the i-th agent, and $r_k^{a_t^i}$ indicates the reward for action a_t^i. Moreover, N denotes the number of agents, M denotes the maximum steps in an episode, and K denotes the maximum episodes.

3) RULE-BASED ACTION GENERATOR (RAG)

The rule-based action generator (RAG) is inspired by the work from chen *et al.* [27], which is one of the most recently rule-based methods for producing swarm animation. The primary notion of RAG, as depicted in Figure 4, is to calculate the velocity of each agent in the next time step based on the current position and velocity, and convert it into a DRL-relevant action vector $a = \{v, \omega\}$. RAG sorts agents by three spheres, denoted by three radii R_{sep}, R_{ali} and R_{coh} ($0 < R_{sep} < R_{ali} < R_{coh}$). Specifically, R_{sep} represents the repulsion zone which is the private separation space for each agent; R_{ali} indicates the alignment region where the agents tend to align with their neighbors; and R_{coh} signifies an attractive zone where the agents are cohesion.

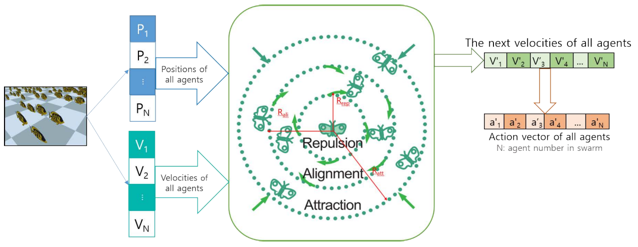


FIGURE 4. Rule-based action generator.

The force on i-th agent F_i consists of three components: the repulsion force F_i^{sep}, the alignment force F_i^{ali}, and the attraction force F_i^{coh}. The three forces are computed as follow:

$$F_i^* = \frac{w_*}{N_*} \sum_{j=1}^{N_*} \left(T(|p_{ij}|) \frac{p_{ij}}{|p_{ij}|} + (1 - T(|p_{ij}|)^2) \frac{v_{ij}}{|v_{ij}|} \right) \quad (8)$$

where $F_i^* \in \{F_i^{sep}, F_i^{ali}, F_i^{coh}\}$ signify the three types of forces for the i-th agent, and N_* indicate the numbers of neighbors when these three different forces are calculated, respectively. Furthermore, the constant parameters w_* denote the weights for the three forces. $p_{ij} = p_j - p_i$ is a vector from the position of i-th agent p_i to its j-th neighbor p_j. Similarly, $v_{ij} = v_j - v_i$ represents the velocity difference between the i-th agent and its j-th neighbor. The value of $T(|p_{ij}|)$ can be calculated as follows:

$$T(|p_{ij}|) = \begin{cases} -1, & 0 \le |p_{ij}| \le R_{sep}; \\ 0, & R_{sep} \le |p_{ij}| \le R_{ali}; \\ 1, & R_{ali} \le |p_{ij}| \le R_{coh}. \end{cases} \quad (9)$$

RAG collects the velocity of each agent every Δt time step. The following equation (Eq. 10) can be used to transform of the i-th agent's next action, $a_i = \{v_i, \omega_i\}$.

$$v_i = |v_i'| - |v_i|,$$
$$\omega_i = \arccos\left(\frac{v_i \cdot v_i'}{|v_i||v_i'|}\right) \quad (10)$$

where v_i is the current velocity of the i-th agent, and v_i' is the new velocity v_i' in the next time step, which can be easily calculated with the F_i. As shown in Figure 5, the generated action vectors and state observations are stored in an experience replay buffer. This study proposes to keep the RAG-generated experience in the experience replay buffer for K episodes or until the leader agent is changed.

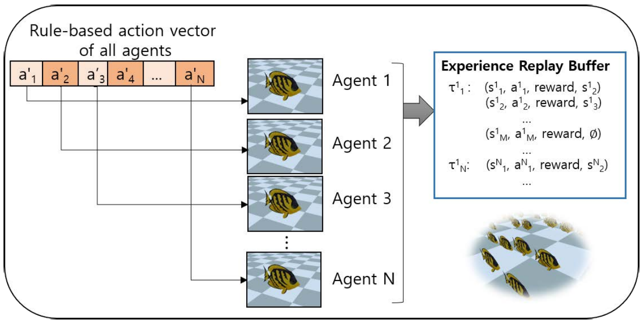


FIGURE 5. Rule-based experience collection.

D. OBSERVATIONS AND ACTIONS

The policy is a mapping from state observations to actions. The state observation $\mathcal{S}$ is constructed with local observations and embedded features of swarm (EFS).

Local observations of an agent, as defined in Eq. 11, include the agent's speed ($s_u \in \mathbb{R}$) and orientation ($s_q \in \mathbb{R}^3$), the vector from the position of the agent to the leader agent ($s_{lead}^p \in \mathbb{R}^3$), the distance between the agent and the leader agent ($s_{lead}^d \in \mathbb{R}$), the relative speed to the leader agent ($s_{lead}^u \in \mathbb{R}$).

$$\begin{aligned} s_u &= \hat{u} \\ s_q &= \hat{q} \\ s_{lead}^p &= p_{lead} - \hat{p} \\ s_{lead}^d &= ||s_{lead}^p||_2 \\ s_{lead}^u &= u_{lead} - \hat{u} \end{aligned} \quad (11)$$

where $\hat{u}$, $\hat{q}$ and $\hat{p}$ represent the current agent's speed, orientation, and position, respectively. Furthermore, in the 3d hemisphere of the agent's heading direction, 42 rays with the length of R_{coh} are cast at 30° intervals. This observation $s^d_{ray} \in \mathbb{R}^{42}$ stores distances between the agent and detected objects (eg., obstacles or other agents), however, the negative distance is only stored if the detected object is an obstacle.

Embedded features of swarm (EFS) is inspired by the mean embeddings presented by Hüttenrauch *et al.* [136]. It employs the average speed of neighbors $\bar{s_u}$, the mean square error (MSE) on orientation $\bar{s_q}$ and position $\bar{s_d}$ between the agent and its neighbors. The key advantage of using EFS is that it overcomes the limitation of a fixed number of neighbors during the simulation and enhances the scalability of the trained policy. The Eq. 12 shows the definition of EFS, where N_{nbr} indicates the number of neighbors.

$$\begin{aligned}\bar{s_u} &= \frac{1}{N_{nbr}} \sum_{i=1}^{N_{nbr}} u^i_{nbr} \\ \bar{s_q} &= \frac{1}{N_{nbr}} \sum_{i=1}^{N_{nbr}} ||q^i_{nbr} - \hat{q}||_2 \\ \bar{s_d} &= \frac{1}{N_{nbr}} \sum_{i=1}^{N_{nbr}} ||p^i_{nbr} - \hat{p}||_2 \end{aligned} \tag{12}$$

All quantities are concatenated into a single large state observation vector.

$$\mathcal{S} = \{s_u, s_q, s^p_{lead}, s^u_{lead}, s^d_{ray}, \bar{s_u}, \bar{s_q}, \bar{s_d}\} \in \mathbb{R}^{54} \tag{13}$$

The action vector ($\mathcal{A} \in \mathbb{R}^4$) of an agent is composed of a change in speed $\nu \in \mathbb{R}$ and a change in orientation $\omega \in \mathbb{R}^3$, as mentioned in Section III-B.

1) REWARD FUNCTIONS

For generating swarm behaviors, a reward function $\mathcal{R}$ with five terms was dedicated to evaluate agents' actions during training.

$$\mathcal{R} = r_{spd} + r_{dir} + r_{leader} + r_{obs} + r_{eqnarray} \tag{14}$$

The r_{spd} term indicates that the agent's speed is more similar to its neighbors, the higher reward is received.

$$r_{spd} = w_{spd} \exp(-\lambda_{spd} \frac{1}{N_{nbr}} \sum_{i}^{N_{nbr}} (|\hat{u} - u_i|)) \tag{15}$$

where N_{nbr} denotes the number of neighbors, $\hat{u}$ is the speed of the current agent, and w_{spd} represents the weight of r_{spd}. Moreover, λ_{spd} indicates the rate of the reward's change, the greater it is, the more sensitive the reward's change is. The r_{dir} term represents that the agent receives a higher reward if its orientation resembles its neighbors'.

$$r_{dir} = w_{dir} \exp(-\lambda_{dir} \frac{1}{N_{nbr}} \sum_{i}^{N_{nbr}} (||\hat{q} - q_i||_2)) \tag{16}$$

where $\hat{q}$ denotes the orientation of the current agent, w_{dir} represents the weight of r_{dir}, and the meaning of λ_{dir} is similar to that of λ_{spd}. The r_{leader} term leds the agent following the leader agent by giving a higher reward if the agent's position is closer to the leader agent's repulsion zone.

$$r_{leader} = w_{leader} \exp(-\lambda_{leader} \cdot ||\hat{p} - p_{leader}||_2 + R_{sep})) \tag{17}$$

where w_{dir} represents the weight of r_{leader}, and the meaning of λ_{leader} is similar to that of λ_{spd}. $\hat{p}$ and p_{leader} denote the position of the current agent and the leader agent, respectively. The r_{obs} term checks if the agent is too close to obstacles.

$$r_{obs} = \exp((\lambda_{obs} \sum_{j}^{N_{obs}} ||\hat{p} - p_j||_2) - R_{sep}) - w_{obs} \tag{18}$$

where w_{obs} indicates a penalty if agent-obstacle collision occurred. The last term $r_{eqnarray}$ plays a role in keeping alignment with the agent's neighbors.

$$r_{eqnarray} = \frac{1}{N_{nbr}} \sum_{i}^{N_{nbr}} -\left(||\hat{p} - p_i||_2 - \frac{R_{ali} - R_{sep}}{2}\right)^2 + w_{dist} \tag{19}$$

where w_{obs} indicates the maximum reward when the agent keeps an appropriate distance from its neighbors, and the appropriate distance is determined by $\frac{R_{ali}-R_{sep}}{2}$, which lies in the middle of the alignment zone.

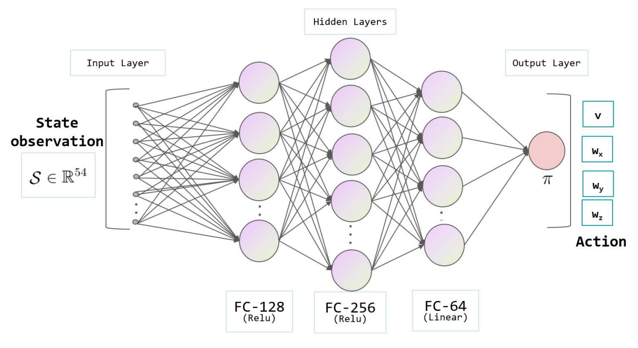
IV. EXPERIMENTAL RESULTS

This section starts by describing the configurations of experiments. Subsequently, the control accuracy, the swarm behavior quality, and the scalability of the proposed method are evaluated experimentally. Eventually, the comparison between the proposed dynamic reward function and the fixed reward method given in the previous study.

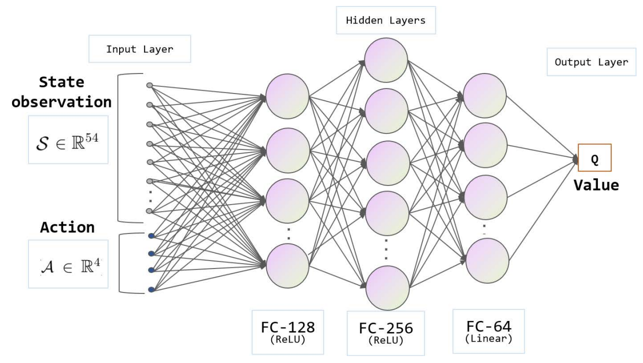
A. EXPERIMENT CONFIGURATIONS

1) NEURAL NETWORKS DESIGN

As illustrated in Figure 3, DDPG (Deep Deterministic Policy Gradient [133]) is composed of four neural networks: an actor network, a critic network, an actor target network, and a critic target network. The actor target network has the same structure as the actor network, while the critic target network has the same structure as the critic network. The structures of these two categories of neural networks used in the experiments are shown in Figure 6. Actor networks output an action for a given state. They are fed with state observations in the form of a 54-dimensional vector. Actor networks are composed of three fully connected layers: two of the layers have ReLU (Rectified Linear Unit [137]) activation functions with 128 and 256 neurons, respectively; the final layer has a linear output activation function with 64 neurons. The output of actor networks is a 4-dimensional action vector that contains the change in speed and orientation. Critic networks predict the value of a given action and state. They have a similar structure as actor networks except that their input includes an additional action vector from the actor network and the final outcome is a scalar value.



(a) Structure of Actor Networks



(b) Structure of Critic Networks

FIGURE 6. Neural networks design.

2) MODULAR CONTROL

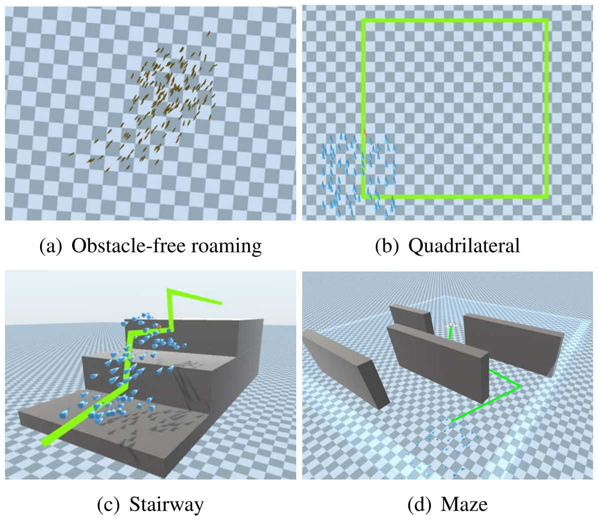
A navigation module, which is a specified path consisting of waypoints, is added to the simulation since it more precisely controls the leader agent than a joystick.

The waypoint of the navigation module contains the target position information and the target velocity information for the leader agent. The green lines in Figure 7 (b), (c), and (d) indicate the path of the navigation module.

3) SCENARIOS AND PARAMETERS

As shown in Figure 7, four different scenarios are dedicated in this study to evaluate the performance of the proposed method. *Obstacle-free roaming* is a scenario without obstacles in which the user uses the joystick to control the swarm. *Quadrilateral* scenario contains a quadrilateral path in an obstacle-free environment to control the swarm. *Stairway* scenario involves a stair-like obstacle and the swarm move along a specified stair-like path. *Maze* scenario is a complex environment with numerous obstacles.

Figure 8 shows an example of the training setup, where the orange cone in the training environment represents the leader agent and the blue cones represent follower agents. All parameters for the experiments are summarized in Table 1.



(a) Obstacle-free roaming (b) Quadrilateral

(c) Stairway (d) Maze

FIGURE 7. Experiment scenarios.

TABLE 1. Parameters of experiments.

Parameters	Values
Repulsion radius (R_{rep})	1.0
Alignment radius (R_{ali})	9.0
Attractive radius (R_{att})	18.0
Time step (Δt)	0.1
Learning rate (actor)	1e-4
Leaining rate (critic)	1e-3
Discount factor (γ)	0.98
Replay buffer size	10,000
Batch size (H)	300
Max episodes (Z)	50,000
Episodes to change leader (K)	500
Max step (M)	1,200
Target update rate (λ)	1e-3
Exploration noise (Ψ)	0.05
Change unit in speed (Δu)	0.1
Change unit in orientation (Δ q)	4°
w_{spd}	1.3
w_{dir}	1.0
w_{leader}	4.9
w_{obs}	16.0
w_{dist}	9.5

B. CONTROL ACCURACY

To evaluate the control accuracy of the proposed method, this experiment employs the average speed differential between the leader agent and 200 follower agents as a metric. The leader agent's speed varies from *0.5* to *1 unit/s*, while performing *90*- and *180*-degree turns in top, bottom, left, and right directions, respectively. This experiment produces ten recordings, each with one second of duration, for each control test. The metric is summarized in Table 2.

Simultaneously, the evaluations of control accuracy are also carried out in the *Quadrilateral* scenario, the

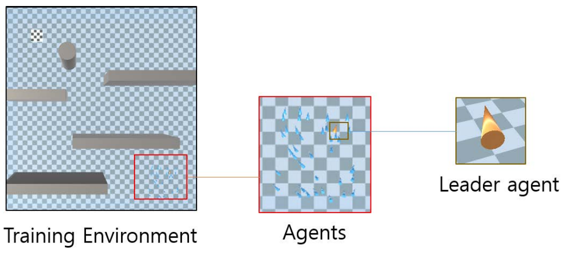


FIGURE 8. Training setup example.

TABLE 2. The average speed differential between agents and the leader agent.

Control		Rule-based	Proposed method
90°	Top	0.1332	0.0974
	Bottom	0.1458	0.0842
	Left	0.1692	0.0528
	Right	0.1260	0.0644
180°	Top	0.2432	0.1337
	Bottom	0.2016	0.1287
	Left	0.3024	0.1551
	Right	0.2754	0.1698

Stairway scenario, and the *Maze* scenario. The initial speed is *0.5 units/s*, and the speed is increased by *0.2 units/s* at each waypoint. Table 3 presents the metrics of these three scenarios and the visual outcomes are shown in Figure 9. The results show that the proposed method has minor average speed differences, indicating that its control accuracy is greater than the rule-based method.

TABLE 3. The average speed differential on three different scenarios.

Scenarios	Rule-based	Proposed method
Quadrilateral	0.1136	0.0934
Stairway	0.3744	0.1127
Maze	0.3962	0.1895

C. SWARM BEHAVIOR QUALITY

Two metrics proposed by Eliot *et al.* [138] are utilized to evaluate the quality of produced swarm animation: *the cohesion/repulsion metric* and *the inter-agent distance metric*. The cohesion/repulsion metric is a distance-based metric that evaluates whether a swarm state is stable by analyzing the average number of repulsion occurrences and attraction occurrences. The inter-agent distance metric indicates how the agents are physically distributed, which only considers the inter-agent distances and their standard deviation. The average distance for the swarm can be calculated by Eq. 20.

$$\mu(P) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N_{nbr}^{i}} ||p_i - p_j^i||_2}{\sum_{i=1}^{N} N_{nbr}^{i}} \tag{20}$$

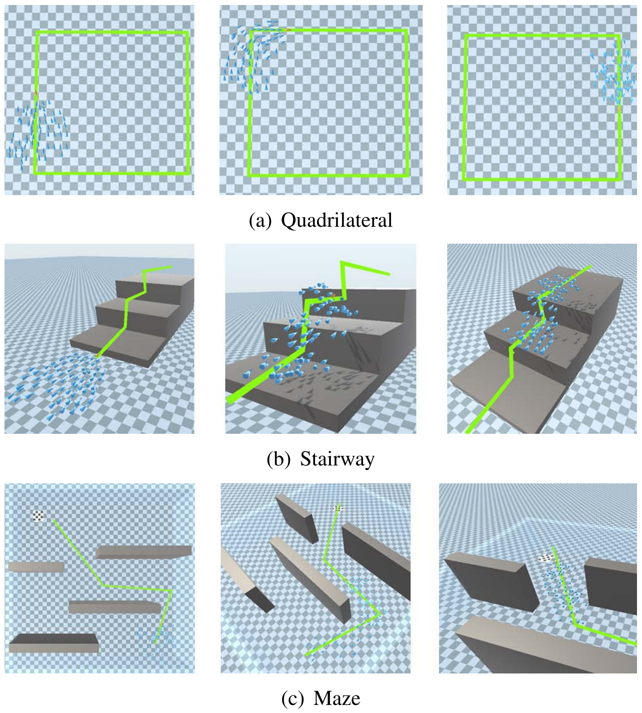


FIGURE 9. Visualization of control accuracy and swarm behavior quality evaluations on three different scenarios.

where P denotes the positions of all agents of the swarm, N_{nbr}^{i} represents the number of neighbor agents for the i-th agent, and p_j^i indicates the position of the j-th neighbor for the i-th agent. As defined in Eq. 21, the standard deviation explains the distribution within the swarm.

$$\sigma(P) = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N_{nbr}^{i}} (||p_i - p_j^i||_2 - \mu(P))^2}{\sum_{i=1}^{N} N_{nbr}^{i}}}. \tag{21}$$

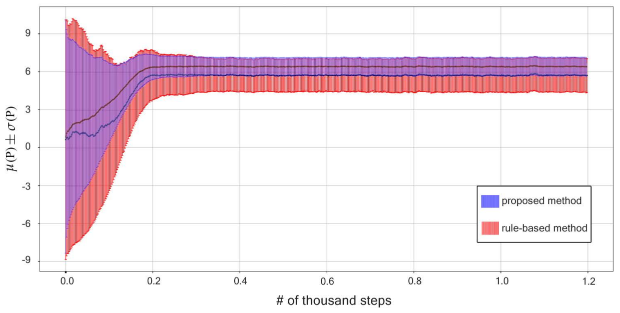


FIGURE 10. The results of inter-agent distance metric over time.

The simulation settings are identical to those mentioned in the previous section for the control accuracy. Table 4 presents the result of the cohesion/repulsion metric. What comes out in the table is that the swarm generated by the proposed method is more stable than the rule-based method since the number of cohesion occurrences is close to the number of repulsion occurrences. Figure 10 shows the evolution of the inter-agent distance metric over time. The clear trend of decreasing the standard deviation indicates that the inter-agent distances within the swarm are getting uniform.

TABLE 4. The result of cohesion/repulsion metric.

Control		Rule-based		Proposed method	
		repulsion	cohesion	repulsion	cohesion
90°	Top	103.0	70.3	61.7	68.1
	Bottom	99.7	48.8	54.7	61.6
	Left	109.1	74.2	69.9	76.7
	Right	95.8	49.0	50.3	53.4
180°	Top	150.6	129.3	78.9	84.6
	Bottom	146.1	57.0	97.6	111.4
	Left	161.7	85.5	96.6	101.6
	Right	149.1	46.3	84.2	82.7

D. SCALABILITY

This study analyzes the scalability of the trained policy with or without embedded features of swarming (EFS) by applying the policy, which was trained with 200 agents, to simulations with 200, 600, and 1,000 agents, respectively. The performance of each simulation is evaluated by computing the average rewards of each step. As shown in Figure 11, for the simulation with 600 agents, the trained policy without EFS is challenging to obtain acceptable average rewards, whereas the trained policy with EFS obtains almost as high average reward as the original simulation with 200 agents. Furthermore, the average rewards with EFS are still acceptable for the simulation with 1,000 agents. The visualization results of each simulation are shown in Figure 12.

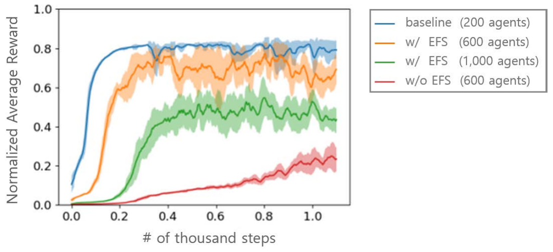


FIGURE 11. Normalized average rewards of the trained policy w/ or w/o EFS.

TABLE 5. Differences between the proposed method and existing methods.

Method	Lee et al.	Hüttenrauch et al.	Proposed method
Training algorithm	DDPG	TRPO	DDPG
On/off policy	Off-policy	On-policy	Off-policy
Rewards	Fixed	Dynamic	Dynamic
Experience Replay	Yes	No	Yes
Scalability	No	Yes	Yes
Movement (2D / 3D)	2D	2D	2D & 3D

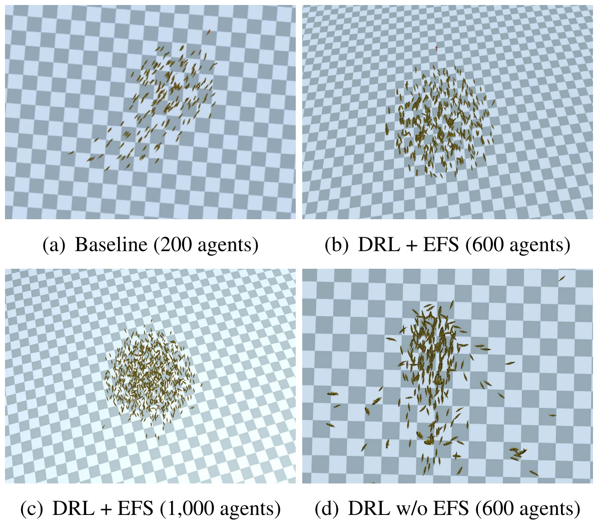

(a) Baseline (200 agents)
(b) DRL + EFS (600 agents)
(c) DRL + EFS (1,000 agents)
(d) DRL w/o EFS (600 agents)

FIGURE 12. Visualization of scalability simulation.

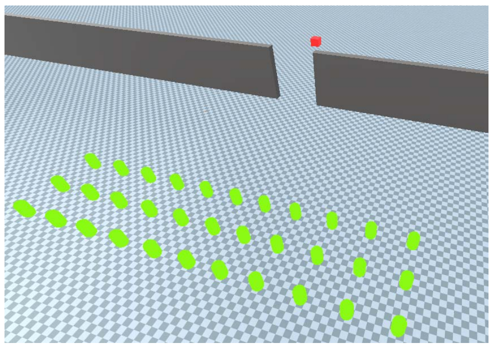


FIGURE 13. Crowd simulation for the evaluation of reward functions.

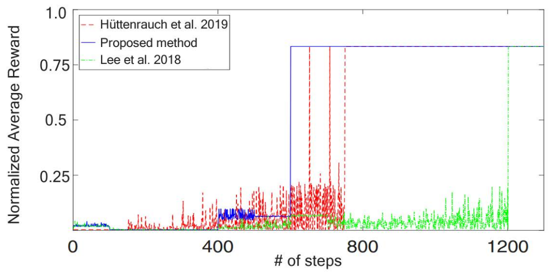


FIGURE 14. Comparison of three different reward functions.

E. DYNAMIC REWARDS VS FIXED REWARDS

In this section, the proposed dynamic reward function is compared with two existing reward functions: a fixed reward function proposed by Lee *et al.* [31] and a dynamic reward function for rendezvous and pursuit-evasion proposed by Hüttenrauch *et al.* [136]. The differences between three methods are summarized in Table 5. To evaluate the effectiveness of the reward functions, a two-dimensional crowd simulation was built. As shown in Figure 13, the red cube represents the target pursued by agents, and green items represent agents.

The crowd simulation aims to guide 30 agents through the gap to the side with the red cube.

Figure 14 compares the normalized average rewards obtained from three different reward functions. From the chart, it can be seen that the proposed method obtains high rewards earlier than the other two methods.

V. CONCLUSION

This paper presents an improved deep reinforcement learning (DRL) method for generating swarm animations. The proposed method aims to generate high-quality swarm animations that respond to user controls in real time. The second goal of this study is to improve the scalability of trained policy even as the number of agents increases. Firstly, this study combined Deep Deterministic Policy Gradient (DDPG) with the rule-based action generator (RAG) to improve the swarm behavior quality by enhancing DDPG's action exploration strategy. The user controls the swarm by interacting with the swarm's leader agent. Four different scenarios have been designed to evaluate the control accuracy and quality of the generated swarm behavior by metrics and visualization in experiments. The experiment results show that the proposed approach outperforms state-of-the-art methods in terms of swarm behavior quality and control accuracy. Secondly, a new state observation quantity of DRL called the embedded features of swarm (EFS) is introduced as a state observation of DRL. In the experiment, the trained policy trained on a group of 200 agents has been applied to swarm with 600 agents and 1,000 agents, respectively. The experimental results show that the trained policy with EFS can scale to a more extensive system than it has been trained on. Finally, various practical, dynamic reward functions are designed for DRL. In the experiment, the proposed dynamic reward functions compared with existing fixed reward functions and dynamic reward functions, respectively. The experimental results show that the proposed dynamic reward functions are more effective than the existing reward functions.

Future studies will continue to expand the categories of animation, such as birds and insects, that the proposed method can produce. Currently, the presented state observations and reward functions are only defined for general swarm behaviors. The corresponding state observations and reward functions for different categories need to be implemented since different categories of swarm required different external influences to be considered. For example, the movement of bird flocks needs to take into account the speed of the wind, and insects have a more random movement of individuals in the swarm, as well as fish need to consider the speed of the current in their movement. Using the energy equation to model an extra velocity vector field in the environment would be a step in the right direction. Moreover, more study is needed to develop a deeper understanding of the effects of centralized training and decentralized training for multi-agent reinforcement learning on swarm animation. Centralized training proposes a virtual central controller to centrally train all agents of a swarm. The proposed method virtually is a decentralized training method in which both the actor network and the critic network are trained on each agent. Centralized training with decentralized execution is a hybrid architecture that trains the actor network on each agent and trains the critic network on a virtual central controller. Centralized training with decentralized execution will be a useful starting point for developing shape-constrained swarm animations, but it may potentially sacrifice scalability due to its architecture.

REFERENCES

[1] M. Dorigo and M. Birattari, "Swarm intelligence," *Scholarpedia*, vol. 2, no. 9, p. 1462, 2007.

[2] R. Bouffanais, *Design and Control of Swarm Dynamics* (SpringerBriefs in Complexity), 1st ed. Singapore: Springer, 2016.

[3] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm robotic behaviors and current applications," *Frontiers Robot. AI*, vol. 7, p. 36, Apr. 2020.

[4] Q. Chao, H. Bi, W. Li, T. Mao, Z. Wang, M. C. Lin, and Z. Deng, "A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving," *Comput. Graph. Forum*, vol. 39, no. 1, pp. 287–308, 2020.

[5] H. Bi, T. Mao, Z. Wang, and Z. Deng, "A data-driven model for lane-changing in traffic simulation," in *Proc. Symp. Comput. Animation*, 2016, pp. 149–158.

[6] D. Wilkie, J. Sewall, and M. Lin, "Flow reconstruction for data-driven traffic animation," *ACM Trans. Graph.*, vol. 32, no. 4, p. 89:1–89:10, Jul. 2013.

[7] C. Kolon and I. B. Schwartz, "The dynamics of interacting swarms," Naval Res. Lab. Washington, DC, USA, Tech. Rep. MR/6790–18-9782, Apr. 2018.

[8] J. N. Yasin, S. A. S. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, "Navigation of autonomous swarm of drones using translational coordinates," in *Advances in Practical Applications of Agents, Multi-Agent Systems, and Trustworthiness. The PAAMS Collection* (Lecture Notes in Computer Science), Y. Demazeau, T. Holvoet, J. M. Corchado, and S. Costantini, Eds. Cham, Switzerland: Springer, 2020, pp. 353–362.

[9] B. N. Sharma, J. Raj, and J. Vanualailai, "Navigation of carlike robots in an extended dynamic environment with swarm avoidance," *Int. J. Robust Nonlinear Control*, vol. 28, no. 2, pp. 678–698, Aug. 2017.

[10] D. Albani, J. IJsselmuiden, R. Haken, and V. Trianni, "Monitoring and mapping with robot swarms for agricultural applications," in *Proc. 14th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Aug. 2017, pp. 1–6.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.* Perth, WA, Australia: IEEE, Nov./Dec. 1995, pp. 1942–1948.

[12] C. A. C. Coello, G. B. Lamont, and D. A. van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems* (Genetic and Evolutionary Computation), 2nd ed. London, U.K.: Springer, 2007.

[13] C. Blum and X. Li, "Swarm intelligence in optimization," in *Swarm Intelligence: Introduction and Applications* (Natural Computing Series), C. Blum and D. Merkle, Eds. Berlin, Germany: Springer, 2008, pp. 43–85.

[14] S. Mirjalili, "Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Comput. Appl.*, vol. 27, no. 4, pp. 1053–1073, May 2016.

[15] M. Mafarja, I. Aljarah, A. A. Heidari, H. Faris, P. Fournier-Viger, X. Li, and S. Mirjalili, "Binary dragonfly optimization for feature selection using time-varying transfer functions," *Knowl.-Based Syst.*, vol. 161, pp. 185–204, Dec. 2018.

[16] Z.-M. Gao and J. Zhao, "An improved grey wolf optimization algorithm with variable weights," *Comput. Intell. Neurosci.*, vol. 2019, Jun. 2019, Art. no. e2981282.

[17] A. Bellaachia and A. Bari, "Flock by leader: A novel machine learning biologically inspired clustering algorithm," in *Advances in Swarm Intelligence* (Lecture Notes in Computer Science), Y. Tan, Y. Shi, and Z. Ji, Eds. Berlin, Germany: Springer, 2012, pp. 117–126.

[18] D. E. Knuth, "Postscript about NP-hard problems," *ACM SIGACT News*, vol. 6, no. 2, pp. 15–16, Apr. 1974.

[19] D. Karaboga and B. Basturk, "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," in *Foundations of Fuzzy Logic and Soft Computing* (Lecture Notes in Computer Science), P. Melin, O. Castillo, L. T. Aguilar, J. Kacprzyk, and W. Pedrycz, Eds. Berlin, Germany: Springer, 2007, pp. 789–798.

[20] A. R. Yıldız, B. S. Yıldız, S. M. Sait, S. Bureerat, and N. Pholdee, "A new hybrid Harris hawks-Nelder–Mead optimization algorithm for solving design and manufacturing problems," *Mater. Test.*, vol. 61, no. 8, pp. 735–743, Aug. 2019.

[21] Z. Cai, X. Chang, Y. Wang, X. Yi, and X.-J. Yang, "Distributed control for flocking and group maneuvering of nonholonomic agents," *Comput. Animation Virtual Worlds*, vol. 28, nos. 3–4, p. e1777, May 2017.

[22] C. S. Ho, "Flocking animation and modelling environment: FAME," M.s. thesis, Dept. School Comput. Eng., Nanyang Technol. Univ., Singapore, 2012.

[23] S. Podila and Y. Zhu, "A 3D animation tool for simulating fish escape behavior," in *Proc. 24th Int. Conf. Inf. Vis. (IV)*, Sep. 2020, pp. 757–760.

[24] J. Kim, Y. Seol, T. Kwon, and J. Lee, "Interactive manipulation of large-scale crowd animation," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 83:1–83:10, Jul. 2014.

[25] J.-H. Kim, J. Im, and J. Lee, "Post-processing framework for enhancing liquid surfaces in VFX pipeline," *IEEE Access*, vol. 9, pp. 91091–91103, 2021.

[26] J.-H. Kim and J. Lee, "Stable and anisotropic freezing framework with interaction between IISPH fluids and ice particles," *IEEE Access*, vol. 9, pp. 146097–146109, 2021.

[27] Q. Chen, G. Luo, Y. Tong, X. Jin, and Z. Deng, "Shape-constrained flying insects animation," *Comput. Animation Virtual Worlds*, vol. 30, nos. 3–4, May 2019, Art. no. e1902.

[28] L. D'Alfonso, A. Bono, and A. Filice, "A kinematic swarm model for vortex-like behavior around an uncertain target," in *Proc. 25th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*. Piscataway, NJ, USA: IEEE, Sep. 2020, pp. 435–440.

[29] X. Wang, J. Ren, X. Jin, and D. Manocha, "BSwarm: Biologically-plausible dynamics model of insect swarms," in *Proc. 14th ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, J. Barbič and Z. Deng, Eds., New York, NY, USA, 2015, pp. 111–118.

[30] M. Oshita, "Agent navigation using deep learning with agent space heat map for crowd simulation," *Comput. Animation Virtual Worlds*, vol. 30, nos. 3–4, May 2019, Art. no. e1878.

[31] J. Lee, J. Won, and J. Lee, "Crowd simulation by deep reinforcement learning," in *Proc. 11th Annu. Int. Conf. Motion, Interact., Games*, P. Charalambous, Ed. New York, NY, USA, Nov. 2018, pp. 1–7.

[32] W. Xiang, X. Yao, H. Wang, and X. Jin, "FASTSWARM: A data-driven framework for real-time flying insect swarm simulation," *Comput. Animation Virtual Worlds*, vol. 31, nos. 4–5, Jul. 2020, Art. no. e1957.

[33] Y.-S. Luo, J. H. Soeseno, T. P.-C. Chen, and W.-C. Chen, "Carl: Controllable agent with reinforcement learning for quadruped locomotion," in *Proc. ACM Trans. Graph.*, vol. 39, no. 4, pp. 38–41, 2020.

[34] S. Taylor, T. Kim, Y. Yue, M. Mahler, J. Krahe, A. G. Rodriguez, J. Hodgins, and I. Matthews, "A deep learning approach for generalized speech animation," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 93:1–93:11, Aug. 2017.

[35] H. R. Pulliam, "On the advantages of flocking," *J. Theor. Biol.*, vol. 38, no. 2, pp. 419–422, Feb. 1973.

[36] L. Landeau and J. Terborgh, "Oddity and the 'confusion effect' in predation," *Animal Behav.*, vol. 34, no. 5, pp. 1372–1380, Oct. 1986.

[37] B. L. Partridge, "The structure and function of fish schools," *Sci. Amer.*, vol. 246, no. 6, pp. 114–123, Jun. 1982.

[38] T. J. Pitcher, "Functions of shoaling behaviour in teleosts," in *The Behaviour of Teleost Fishes*, T. J. Pitcher, Ed. Boston, MA, USA: Springer, 1986, pp. 294–337.

[39] M. A. Elgar, "Predator vigilance and group size in mammals and birds: A critical review of the empirical evidence," *Biol. Rev.*, vol. 64, no. 1, pp. 13–33, Feb. 1989.

[40] A. Huth and C. Wissel, "The simulation of fish schools in comparison with experimental data," *Ecol. Model.*, vols. 75–76, pp. 135–146, Sep. 1994.

[41] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proc. 14th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, M. C. Stone, Ed. New York, NY, USA: Association for Computing Machinery, Jul. 1987, pp. 25–34.

[42] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Phys. Rev. Lett.*, vol. 75, no. 6, pp. 1226–1229, 1995.

[43] C. R. Ward, F. Gobet, and G. Kendall, "Evolving collective behavior in an artificial ecology," *Artif. Life*, vol. 7, no. 2, pp. 191–209, Apr. 2001.

[44] R. S. Olson, D. B. Knoester, and C. Adami, "Critical interplay between density-dependent predation and evolution of the selfish herd," in *Proc. 15th Annu. Conf. Genet. Evol. Comput. Conf. (GECCO)*, C. Blum and E. Alba, Eds. New York, NY, USA, 2013, p. 247.

[45] K. H. Lee, M. G. Choi, Q. Hong, and J. Lee, "Group behavior from video: A data-driven approach to crowd simulation," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation (SCA)*. Goslar, Germany: Eurographics Association, 2007, pp. 109–118.

[46] Y. Li, M. Christie, O. Siret, R. Kulpa, and J. Pettré, "Cloning crowd motions," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation (SCA)*. Goslar, Germany: Eurographics Association, 2012, pp. 201–210.

[47] J. Ren, X. Wang, X. Jin, and D. Manocha, "Simulating flying insects using dynamics and data-driven noise modeling to generate diverse collective behaviors," *PLoS ONE*, vol. 11, no. 5, May 2016, Art. no. e0155698.

[48] Q. Chao, Z. Deng, J. Ren, Q. Ye, and X. Jin, "Realistic data-driven traffic flow animation using texture synthesis," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 2, pp. 1167–1178, Feb. 2018.

[49] J. Ren, W. Xiang, Y. Xiao, R. Yang, D. Manocha, and X. Jin, "HeterSim: Heterogeneous multi-agent systems simulation by interactive data-driven optimization," *IEEE Trans. Vis. Comput. Graphics*, vol. 27, no. 3, pp. 1953–1966, Mar. 2021.

[50] J. R. Faeder, M. L. Blinov, and W. S. Hlavacek, "Rule-based modeling of biochemical systems with BioNetGen," in *Systems Biology* (Methods in Molecular Biology), I. V. Maly, Ed. Totowa, NJ, USA: Humana Press, 2009, pp. 113–167.

[51] S. Kim, C. Hoffmann, and J. M. Lee, "An experiment in rule-based crowd behavior for intelligent games," in *Proc. 4th Int. Conf. Comput. Sci. Converg. Inf. Technol.*, 2009, pp. 410–415.

[52] C. W. Reynolds, "Steering behaviors for autonomous characters," in *Proc. Game Developers Conf.*, vol. 1999. Princeton, NJ, USA: Citeseer, 1999, pp. 763–782.

[53] X. Tu and D. Terzopoulos, "Artificial fishes: Physics, locomotion, perception, behavior," in *Proc. 21st Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*. New York, NY, USA: Association for Computing Machinery, Jul. 1994, pp. 43–50.

[54] P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, "(Not) evolving collective behaviours in synthetic fish," in *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. Cambridge, MA, USA: MIT Press, 1996, pp. 635–644.

[55] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, no. 6803, pp. 487–490, Sep. 2000.

[56] Y. Liu, K. M. Passino, and M. M. Polycarpou, "Stability analysis of M-dimensional asynchronous swarms with a fixed communication topology," *IEEE Trans. Autom. Control*, vol. 48, no. 1, pp. 76–95, Jan. 2003.

[57] V. Gazi and K. M. Passino, "Stability analysis of swarms," *IEEE Trans. Autom. Control*, vol. 48, no. 4, pp. 692–697, Apr. 2003.

[58] R. O. Saber and R. M. Murray, "Flocking with obstacle avoidance: Cooperation with limited communication in mobile networks," in *Proc. IEEE Conf. Decision Control (CDC)*, vol. 2, Dec. 2003, pp. 2022–2028.

[59] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[60] D. Eui Chang, S. C. Shadden, J. E. Marsden, and R. Olfati-Saber, "Collision avoidance for multiple agent systems," in *Proc. 42nd IEEE Int. Conf. Decis. Control*, vol. 1, Dec. 2003, pp. 539–543.

[61] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Stability of flocking motion," Dept. Elect. Syst. Eng., Univ. Pennsylvania, Philadelphia, PA, USA, Tech. Rep. MS-CIS-03-03, 2003.

[62] R. O. Saber, "A unified analytical look at Reynolds flocking rules," Control Dyn. Syst., California Inst. Tech., Pasadena, CA, USA, Tech. Rep. CDS 03-014, 2003.

[63] H. P. H. Shum, T. Komura, M. Shiraishi, and S. Yamazaki, "Interaction patches for multi-character animation," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 114:1–114:8, Dec. 2008.

[64] J. Pettre, J.-P. Laumond, and D. Thalmann, "A navigation graph for real-time crowd animation on multilayered and uneven terrain," in *Proc. 1st Int. Workshop Crowd Simulation*, vol. 43. New York, NY, USA: Pergamon, 2005, p. 194.

[65] H. Noser and D. Thalmann, "The animation of autonomous actors based on production rules," in *Proc. Comput. Animation*, 1996, pp. 47–57.
[66] X. Tu, *Artificial Animals for Computer Animation: Biomechanics, Locomotion, Perception, and Behavior*. Springer, Berlin, Germany, Dec. 1999.
[67] A. Pilco, X. Luo, A. A. N. Newball, C. Zúñiga, and C. Lozano-Garzón, "Procedural animation generation technology of virtual fish flock," in *Proc. Int. Conf. Virtual Reality Vis. (ICVRV)*, Nov. 2019, pp. 233–237.
[68] M. Anderson, E. McDaniel, and S. Chenney, "Constrained animation of flocks," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, 2003, pp. 286–297.
[69] J. Xu, X. Jin, Y. Yu, T. Shen, and M. Zhou, "Shape-constrained flock animation," *Comput. Animation Virtual Worlds*, vol. 19, nos. 3–4, pp. 319–330, 2008.
[70] M. Klotsman and A. Tal, "Animation of flocks flying in line formations," *Artif. Life*, vol. 18, no. 1, pp. 91–105, Dec. 2011.
[71] X. Wang, L. Zhou, Z. Deng, and X. Jin, "Flock morphing animation," *Comput. Animation Virtual Worlds*, vol. 25, nos. 3–4, pp. 351–360, 2014.
[72] L. Zheng, J. Zhao, Y. Cheng, H. Chen, X. Liu, and W. Wang, "Geometry-constrained crowd formation animation," *Comput. Graph.*, vol. 38, pp. 268–276, Feb. 2014.
[73] D. Wolinski, S. J. Guy, A.-H. Olivier, M. Lin, D. Manocha, and J. Pettré, "Parameter estimation and comparative evaluation of crowd simulations," *Comput. Graph. Forum*, vol. 33, no. 2, pp. 303–312, May 2014.
[74] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, "Neural volumes: Learning dynamic renderable volumes from images," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 65:1–65:14, Jul. 2019.
[75] A. Keller, J. Křivánek, J. Novák, A. Kaplanyan, and M. Salvi, "Machine learning and rendering," in *Proc. ACM SIGGRAPH Courses*. New York, NY, USA: Association for Computing Machinery, Aug. 2018, pp. 1–2.
[76] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B. Goldman, and M. Zollhöfer, "State of the art on neural rendering," *Comput. Graph. Forum*, vol. 39, no. 2, pp. 701–727, 2020.
[77] Y. Zhang, W. Dong, C. Ma, X. Mei, K. Li, F. Huang, B.-G. Hu, and O. Deussen, "Data-driven synthesis of cartoon faces using different styles," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 464–478, Jan. 2017.
[78] M. A. Otaduy, B. Bickel, D. Bradley, and H. Wang, "Data-driven simulation methods in computer graphics: Cloth, tissue and faces," in *Proc. ACM SIGGRAPH Courses (SIGGRAPH)*. New York, NY, USA: Association for Computing Machinery, Aug. 2012, pp. 1–96.
[79] N. Jin, Y. Zhu, Z. Geng, and R. Fedkiw, "A pixel-based framework for data-driven clothing," *Comput. Graph. Forum*, vol. 39, no. 8, pp. 135–144, Dec. 2020.
[80] D. Fan, L. Yang, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, "Reinforcement learning for bluff body active flow control in experiments and simulations," *Proc. Nat. Acad. Sci. USA*, vol. 117, no. 42, pp. 26091–26098, Oct. 2020.
[81] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "MeshCNN," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–12, Aug. 2019.
[82] F. Milano, A. Loquercio, A. Rosinol, D. Scaramuzza, and L. Carlone, "Primal-dual mesh convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds. Red Hook, NY, USA: Curran Associates, 2020, pp. 952–963.
[83] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Red Hook, NY, USA: Curran Associates, Dec. 2017, pp. 5105–5114.
[84] A. Sinha, J. Bai, and K. Ramani, "Deep learning 3D shape surfaces using geometry images," in *Computer Vision—ECCV 2016* (Lecture Notes in Computer Science), B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 223–240.
[85] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.
[86] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov, "Shape Google: Geometric words and expressions for invariant shape retrieval," *ACM Trans. Graph.*, vol. 30, no. 1, pp. 1:1–1:20, Jan. 2011.
[87] D. Ezuz, J. Solomon, V. G. Kim, and M. Ben-Chen, "GWCNN: A metric alignment layer for deep shape analysis," *Comput. Graph. Forum*, vol. 36, no. 5, pp. 49–57, Aug. 2017.
[88] L. Gao, Y.-K. Lai, J. Yang, L.-X. Zhang, S. Xia, and L. Kobbelt, "Sparse data driven mesh deformation," *IEEE Trans. Vis. Comput. Graphics*, vol. 27, no. 3, pp. 2085–2100, Mar. 2021.
[89] Z.-S. Wang, J. Lee, C. G. Song, and S.-J. Kim, "Data-driven point sampling with blue-noise properties for triangular meshes," in *Proc. 3rd Int. Conf. Comput. Sci. Softw. Eng. (CSSE)*. New York, NY, USA: Association for Computing Machinery, May 2020, pp. 77–82.
[90] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent complexity via multi-agent competition," in *Proc. Int. Conf. Learn. Represent.*, Feb. 2018, pp. 1–12.
[91] X. B. Peng, G. Berseth, and M. Van de Panne, "Terrain-adaptive locomotion skills using deep reinforcement learning," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 81:1–81:12, Jul. 2016.
[92] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2015, pp. 1889–1897.
[93] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," Aug. 2017, *arXiv:1707.06347*.
[94] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, Nov. 2018.
[95] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, Jan. 2019, Art. no. eaau5872.
[96] J. Won and J. Lee, "Learning body shape variation in physics-based characters," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 207:1–207:12, Nov. 2019.
[97] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3389–3396.
[98] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange, "Reinforcement learning for robot soccer," *Auton. Robots*, vol. 27, no. 1, pp. 55–73, Jul. 2009.
[99] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[100] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[101] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[102] M. Xu, H. Jiang, X. Jin, and Z. Deng, "Crowd simulation and its applications: Recent advances," *J. Comput. Sci. Technol.*, vol. 29, no. 5, pp. 799–811, Sep. 2014.
[103] H. Y. Ling, F. Zinno, G. Cheng, and M. Van De Panne, "Character controllers using motion VAEs," *ACM Trans. Graph.*, vol. 39, no. 4, pp. 40:1–40:12, Aug. 2020.
[104] D. Thalmann, "Crowd simulation," in *Encyclopedia of Computer Graphics and Games*, N. Lee, Ed. Cham, Switzerland: Springer, 2016, pp. 1–8.
[105] R. Narain, A. Golas, S. Curtis, and M. C. Lin, "Aggregate dynamics for dense crowd simulation," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–8, Dec. 2009.
[106] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1160–1168, Jul. 2006.
[107] T. Kretz and M. Schreckenberg, "The F.A.S.T.-model," in in *Proc. Int. Conf. Cellular Automata*, Sep. 2006, pp. 712–715.
[108] M. Lhommet, D. Lourdeaux, and J.-P. Barthès, "Never alone in the crowd: A microscopic crowd model based on emotional contagion," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol.*, Aug. 2011, pp. 89–92.
[109] D. Wolinski, M. C. Lin, and J. Pettré, "WarpDriver: Context-aware probabilistic motion prediction for crowd simulation," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 164:1–164:11, Nov. 2016.
[110] N. Fridman and G. A. Kaminka, "Towards a cognitive model of crowd behavior based on social comparison theory," in *Proc. AAAI*, 2007, pp. 731–737.
[111] I. Karamouzas, N. Sohre, R. Narain, and S. J. Guy, "Implicit crowds: Optimization integrator for robust crowd simulation," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 136:1–136:13, Jul. 2017.

[112] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2010, pp. 981–986.

[113] F. Martinez-Gil, M. Lozano, and F. Fernández, "Strategies for simulating pedestrian navigation with multiple reinforcement learning agents," *Auton. Agents Multi-Agent Syst.*, vol. 29, no. 1, pp. 98–130, Jan. 2015.

[114] L. Torrey, "Crowd simulation via multi-agent reinforcement learning," in *Proc. 6th AAAI Conf. Artif. Intell. Interact. Digit. Entertainment (AIIDE)*. Stanford, CA, USA: AAAI Press, Oct. 2010, pp. 89–94.

[115] Q. Wang, H. Liu, K. Gao, and L. Zhang, "Improved multi-agent reinforcement learning for path planning-based crowd simulation," *IEEE Access*, vol. 7, pp. 73841–73855, 2019.

[116] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *Proc. Int. Conf. Robot. Autom. (ICRA)*. Piscataway, NJ, USA: IEEE, May 2019, pp. 6015–6022.

[117] S. Park, H. Ryu, S. Lee, S. Lee, and J. Lee, "Learning predict-and-simulate policies from unorganized human motion data," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 205:1–205:11, Nov. 2019.

[118] N. Chentanez, M. Müller, M. Macklin, V. Makoviychuk, and S. Jeschke, "Physics-based motion capture imitation with deep reinforcement learning," in *Proc. 11th Annu. Int. Conf. Motion, Interact., Games*. New York, NY, USA: Association for Computing Machinery, Nov. 2018, pp. 1–10.

[119] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 143:1–143:14, Jul. 2018.

[120] A. Clegg, W. Yu, J. Tan, C. K. Liu, and G. Turk, "Learning to dress: Synthesizing human dressing motion via deep reinforcement learning," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 179:1–179:10, Dec. 2018.

[121] K. Bergamin, S. Clavet, D. Holden, and J. R. Forbes, "DReCon: Data-driven responsive control of physics-based characters," *ACM Trans. Graph.*, vol. 38, no. 6, pp. 206:1–206:11, Nov. 2019.

[122] H. Yu, T. Komura, and J. J. Zhang, "Data-driven animation technology (D2AT)," in *Proc. SIGGRAPH Asia Workshops*. New York, NY, USA: Association for Computing Machinery, Nov. 2017, pp. 1–4.

[123] Q. Chao, J. Shen, and X. Jin, "Video-based personalized traffic learning," *Graph. Models*, vol. 75, no. 6, pp. 305–317, Nov. 2013.

[124] C. D. Boatright, M. Kapadia, J. M. Shapira, and N. I. Badler, "Context-sensitive data-driven crowd simulation," in *Proc. 12th ACM SIGGRAPH Int. Conf. Virtual-Reality Continuum Appl. Ind.*, New York, NY, USA, 2013, pp. 51–56.

[125] P. Charalambous and Y. Chrysanthou, "The PAG crowd: A graph based approach for efficient data-driven crowd simulation," *Comput. Graph. Forum*, vol. 33, no. 8, pp. 95–108, Dec. 2014.

[126] H. Du, M. Manns, E. Herrmann, and K. Fischer, "Joint angle data representation for data driven human motion synthesis," *Proc. CIRP*, vol. 41, pp. 746–751, Jan. 2016.

[127] A. Gopalakrishnan, A. Mali, D. Kifer, L. Giles, and A. G. Ororbia, "A neural temporal model for human motion prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12116–12125.

[128] M. Kim, G. Pons-Moll, S. Pujades, S. Bang, J. Kim, M. J. Black, and S.-H. Lee, "Data-driven physics for human soft tissue animation," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 54:1–54:12, Jul. 2017.

[129] Q. Chen, Y. Wang, H. Wang, and X. Yang, "Data-driven simulation in fluids animation: A survey," *Virtual Reality Intell. Hardw.*, vol. 3, no. 2, pp. 87–104, Apr. 2021.

[130] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks," in *Proc. Int. Conf. Mach. Learn.*, Nov. 2020, pp. 8459–8468.

[131] M. L. Puterman, Ed., *Markov Decision Processes* (Wiley Series in Probability and Statistics). Hoboken, NJ, USA: Wiley, Apr. 1994.

[132] M. Wiering and M. van Otterlo, Eds., *Reinforcement Learning: State-of-the-Art* (Adaptation, Learning, and Optimization), vol. 12. Heidelberg, Germany: Springer, 2012.

[133] T. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *Proc. 4th Int. Conf. Learn. Represent. (ICLR)* San Juan, Puerto Rico, Jan. 2016, pp. 2–4.

[134] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2014, pp. 387–395.

[135] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *Proc. NIPS Deep Learn. Workshop*, 2013. [Online]. Available: https://arxiv.org/abs/1312.5602

[136] M. Hüttenrauch, S. Adrian, and G. Neumann, "Deep reinforcement learning for swarm systems," *J. Mach. Learn. Res.*, vol. 20, no. 54, pp. 1–31, 2019.

[137] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML Workshop Deep Learn. Audio, Speech Lang. Process.*, 2013, p. 3.

[138] N. Eliot, D. Kendall, and M. Brockway, "A new metric for the analysis of swarms using potential fields," *IEEE Access*, vol. 6, pp. 63258–63267, 2018.

ZONG-SHENG WANG received the B.S. and M.S. degrees in computer engineering and the Ph.D. degree in convergence software from Hallym University, Chuncheon, South Korea, in 2012, 2014, and 2021, respectively. Since September 2021, he has been a Researcher Fellow with Hallym University. His current research interests include computer graphics, virtual reality rehabilitation, deep reinforcement learning, and optimization algorithms.

CHANG GEUN SONG received the B.S. degree in computer science and statistics from Seoul National University, in 1981, the M.S. degree in computer science from the KAIST, in 1983, and the Ph.D. degree in electrical engineering and computer science (EECS) from The University of Oklahoma, in 1992.

He was the Senior Vice President of Industry-University Cooperation and also the Director of the Industry Academic Cooperation Foundation, Hallym University, South Korea. He has been a Visiting Professor with Imperial College London, in 1995, and the Georgia Institute of Technology, USA, in 1996. He is currently a Professor with the School of Software, Hallym University. He has authored more than 50 technical papers in international journals and conferences and has published 28 domestic journals and more than 110 conference papers. He is the inventor of 13 patents and applications. His research interests include virtual reality/augmented reality (VR/AR), 3D computer graphics, HCI, and algorithm and scientific computation.

JUNG LEE is currently an Associate Professor with the Department of Convergence Software, Hallym University. His current research interests include augmented/virtual reality, fluid animation, and computer graphics.

JONG-HYUN KIM received the B.A. degree from the Department of Digital Contents, Sejong University, in 2008, and the M.S. and Ph.D. degrees from the Department of Computer Science and Engineering, Korea University, in 2010 and 2016, respectively. He is currently an Associate Professor with the School of Software Application, Kangnam University. His research interests include physics-based simulation and natural phenomenon modeling and virtual production.

SUN-JEONG KIM is currently a Professor with the Department of Convergence Software, Hallym University. Her research interests include geometric modeling, scientific visualization, virtual reality, augmented reality, and GP-GPU programming.