IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 17, 2019, accepted July 30, 2019, date of publication August 6, 2019, date of current version August 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2933492

Crowd Navigation in an Unknown and Dynamic Environment Based on Deep Reinforcement Learning

LIBO SUN, JINFENG ZHAI, AND WENHU QIN
School of Instrument Science and Engineering, Southeast University, Nanjing 210096, China

Corresponding authors: Libo Sun (sunlibo@seu.edu.cn) and Wenhu Qin (qinwenhu@seu.edu.cn)

This work was supported in part by the National Science Foundation of China under Grant 61300101, in part by the National Science and Technology Major Project of the Ministry of Science and Technology of China under Grant 2014ZX07405002, in part by the Fundamental Research Funds for the Central Universities under Grant 2242019k30043, and in part by the Key Research and Development Program of Jiangsu Province under Grant BE2017035.

ABSTRACT This paper presents an approach for solving the crowd navigation problem in an unknown and dynamic environment based on deep reinforcement learning. In our approach, we first make four leader agents learn how to reach their goals and avoid collisions with static and dynamic obstacles in an unknown environment by use of proximal policy optimization combined with Long short-term memory and a collision prediction algorithm. In the second stage, we make each leader agent arrive at a specific goal several times and record its trajectory as the guiding path so that the members in its group know how to reach their goals. We adopt the Reciprocal Velocity Obstacle algorithm to make agents not collide with others. Finally, we simulate the scenario of four groups moving towards their goals simultaneously using the Unity 3D engine. The experimental results demonstrate self-learning ability of a crowd who can reach their goals successfully in an unknown and dynamic environment.

INDEX TERMS Crowd simulation deep reinforcement learning, long short-term memory, navigation, proximal policy optimization.

I. INTRODUCTION

Crowd simulation has been gaining considerable attention due to its applications in entertainment, education, architecture, training, urban engineering and virtual heritage. Crowd simulation consists of many different components, including perception, path planning, behavior, locomotion and how to integrate them effectively. Path planning and decision making can guarantee that agents reach their goals without colliding with obstacles and other agents in an optimal way and it is a very important aspect of crowd simulation that researchers should put great effort into.

Simulating crowd motions realistically is complex and difficult. Classical social force models [1], cellular automata models [2], rule-based models [3], [4] are mainly concerned with the realism of crowd behaviors by simulating perception, memory, planning and emotion in every agent. Typical crowd simulations decouple global path planning and local collision avoidance. Global path planning takes the form of graph-based technique or static potential fields since the information of the environment is known and unchanged. Local collision avoidance often gets stuck and may not find a global and existent path because the information of the environment is unknown or known partially. Even Q-learning can find a path to the goal in an unknown environment, but the path is aesthetically unpleasant since the action space is discrete in most of reinforcement learning systems. That is, these methods are not suitable for dealing with navigating crowds in an unknown and dynamic environment in a natural way. Note that an unknown and dynamic environment means that agents don't know the information of the environment at all, including the positions of static as well as dynamic obstacles and the number of static as well as dynamic obstacles.

Recently, deep reinforcement learning (DeepRL)[23] has gained remarkable achievements in many research areas such as physics-based animation, robotics, computer vision and games. It aims at finding an optimal policy that maximizes cumulative rewards and therefore, it is quite suitable for

The associate editor coordinating the review of this manuscript and approving it for publication was Tai-hoon Kim.

solving problems with continuous and high dimensional states and actions. Inspired by deep reinforcement learning, we present an approach for crowd navigation in an unknown and dynamic environment. We first make four leader agents learn how to reach their goals and avoid the collisions with static and dynamic obstacles in an unknown environment by use of Proximal Policy Optimization (PPO)[35] combined with long short-term memory (LSTM)[34] and a collision prediction algorithm[38]. Then we make each leader agent arrive at a specific goal several times and record its trajectory as the guiding path so that the members in its group know how to reach their goals. Finally, we adopt the Reciprocal Velocity Obstacle (RVO) algorithm to make agents not collide with others and furthermore, we simulate the scenario of four groups moving towards their goals simultaneously.

The key contribution of this work can be summarized as follows: 1) The agents can interact with the environment through RL techniques. Furthermore, agents can learn how to avoid collisions with static obstacles in an unknown environment, even if the positions of static obstacles change, which shows that our agents adopt the self-learning ability of human beings. 2) The agents can avoid collisions with dynamic obstacles through predicting the probability of collisions and getting penalized when they collide with dynamic obstacles, which shows the adaptability of our approach in a complex environment. 3) The agents can reach any specified goal successfully in a collision-free path after the training, which shows the generality of our approach. 4) The integration of LSTM can reflect memory characteristics of human beings, which speeds up the learning of the policies for reaching the goals in collision-free paths. 5) Multi-agents with different goals can be trained simultaneously and satisfactory results are obtained, which shows the feasibility of training agents with multi-goals.

The remainder of this paper is organized as follows. In the next section, we review related work in crowd motion simulation. Section 3 describes the framework architecture of our approach. In Section 4, we present the policy presentation. Section 5 describes the learning algorithm for agents to learn how to reach their goals and avoid collisions with static and dynamic obstacles. In Section 6, we show the experimental results and make comparisons with other approaches. Finally, Section 7 draws conclusions and discusses future work.

II. RELATED WORKS

Virtual crowd simulation is a wide topic and the research on it covers many tangible aspects of human locomotive behavior such as the realism of the walking motion itself, collision avoidance, navigation, and local interactions between agents. This overview focuses on the crowd motion simulation. Crowd navigation attempts to reach specific goals while not colliding with other agents nor static or dynamic obstacles in the situated environment, which can be separated into local collision avoidance and global path planning. Global path planning finds an optimal path to the goal while avoiding the collisions with static obstacles based on the established map, in which the environment is known and unchanged. Local collision avoidance steers the preferred velocity away from collisions with other agents in which the environment is unknown or known partially.

Local collision avoidance methods have been proposed including rule-based methods [3], geometrically-based algorithms [5], grid-based methods [6], force-based methods [7], [8], Bayesian decision processes [9] and divergence-free flow tiles [2]. Recent work [10] builds on the adaptive use of algorithms in [11] and adopts machine learning to fit a model that determines which policy to use for a given decision. It analyzes the performance of an ''implicit'' approach to collision avoidance with a data-driven technique and states that a collision avoidance algorithm is necessary for a data-driven approach to steering. However, collision avoidance alone cannot properly model real crowds with a specific goal due to the possibility of getting stuck. Therefore, local methods are often combined with global path planning techniques. In general, global path planning has taken the form of graph-based techniques, such as navigation graphs [12], probabilistic maps [13], coarse graph-based roadmaps [14], [15], navigation meshes [16], [17] and Voronoi diagrams [18]. The main objective of these methods is to find a path free of obstacles between two points in the scene by a search algorithm (usually A*) based on the generated graph. Recent work [19] presents a real-time planning framework for multi-character navigation that enables the use of multiple heterogeneous problem domains of differing complexities for navigation in large, complex, dynamic virtual environments. It demonstrates realtime character navigation for multiple agents in a large-scale, complex, dynamic environment, with precise control and little computational overhead.

Reinforcement learning (RL) has become more popular and has been applied to mobile robots [20], pedestrian navigation [21] and crowd simulation [22], [25]. The classical approaches include Q learning, inverse reinforcement learning (IRL) and Multi-agent reinforcement learning (MARL). The main idea of these methods is to make agents find an optimal control policy that maximizes the reward received in the long term by interacting with the environment by trial and error. Q learning [23], [24] is mostly used in discrete environments, which makes the motion unrealistic, and furthermore, it may be infeasible to learn the Q-values in tabular form due to excessively large state space when there are many obstacles in the environment. IRL [25] is adopted to learn human-like navigation behavior based on example paths and it's also suitable for discrete environments. MARL [21] is used to learn local navigational behaviors to simulate virtual pedestrian groups. The environment is continuous but is quite simple with only five walls.

In recent years, the advent of deep learning has had a significant impact on many areas in machine learning and accelerated progress in RL by use of deep learning algorithms within RL defining the field of ''deep reinforcement learning'' (DRL) [26]. RL approaches lack the scalability and are inherently limited to fairly low-dimensional problems due to

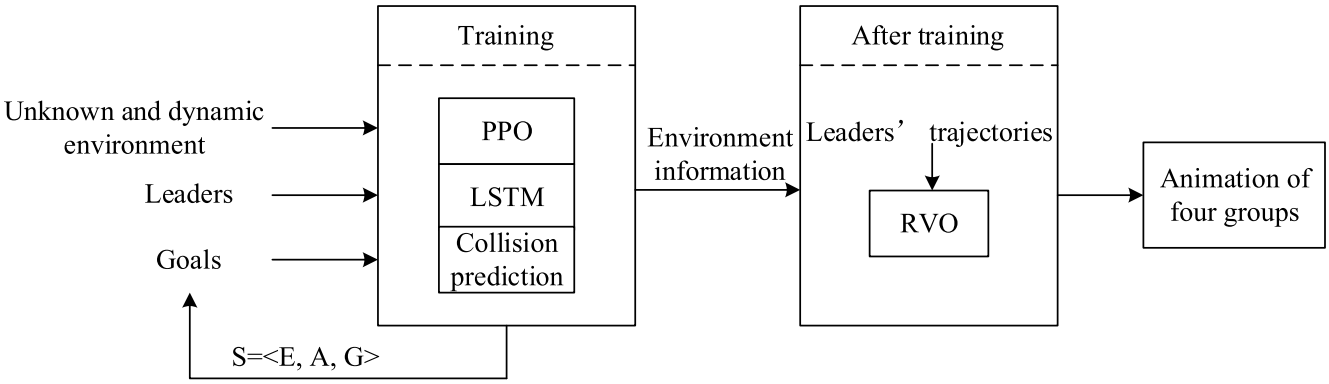

FIGURE 1. The framework architecture of our approach.

its memory complexity and computational complexity. Deep neural networks have powerful function approximation and representation learning properties, and therefore, DRL has provided new tools in dealing with high-dimensional continuous control problems. Among the recent work in the field of DRL, there have been two outstanding representatives. The first was the development of an algorithm that could learn to play a range of Atari 2600 video games at a superhuman level, directly from image pixels [27]. The second was the development of a hybrid DRL system, AlphaGo, that defeated a human world champion in Go [28], paralleling the historic achievement of IBM's Deep Blue in chess two decades earlier [29] and IBM's Watson DeepQA system that beat the best human Jeopardy! players [30]. Besides these two successful stories, DRL has shown promising results. Mnih *et al.* [31] propose a deep RL approach, where the parameters of the deep network are updated by multiple asynchronous copies of the agent in the environment. Feifei Li [32] proposes an actor-critic model whose policy is a function of the goal as well as the current state to address the issue of lacking generalization capability to new target goals; and furthermore, proposes AI2-THOR framework to address the data inefficiency problem for deep reinforcement learning. Wang *et al.* [33] develops a new approach to make agents adapt rapidly to new tasks by leveraging knowledge acquired through previous experience with related activities. Lee *et al.* [34] present an agent-based deep reinforcement learning approach for the navigation and they use only a simple reward function to make agents navigate in different scenarios. Xuebin *at al.* [35] presents a hierarchical learning-based framework for 3D bipedal walking skills that makes limited use of prior structure. It allows for easily-directable control over the motion style and is shown to produce highly robust controllers.

Our method is inspired by the deep reinforcement learning technique, which not only solves the problems that the action space is discrete and it is intractable when the state space is high dimensional in Q learning methods, but also finds feasible paths to reach their goals for crowds while avoiding collisions with static and dynamic obstacles through giving the reward and the penalty.

III. OVERVIEW

The focus of this work is to make the agents navigate successfully in an unknown and dynamic environment. It should not only self-learn the policies for collision avoidance with static and dynamic obstacles, but also plan smooth paths to arrive at their goals. Our approach can be divided into two phases to achieve the aforementioned objectives. The first phase is to train the agents to learn how to avoid the collisions with static and dynamic obstacles, which is realized by using the PPO algorithm combined with LSTM and the collision prediction algorithm. Note that in our paper, dynamic obstacles cannot avoid the collisions with agents inherently, I.e., the dynamic obstacle' trajectories cannot be modified. That is, agents must learn how to realize the collision avoidance. The second phase finds feasible paths to the goal for each of four leader agents and records their trajectories, and finally adopts the RVO algorithm to simulate the scenario of four groups moving towards their goals simultaneously.

The framework architecture is illustrated in FIGURE 1. The input of our approach are the positions of four leader agents, their corresponding goals and the environment including static and dynamic obstacles. In the training phase, the leader agent will get the reward if it reaches the goal or moves towards the goal continuously and get the penalty if it collides with any obstacles and walls or the probability of colliding with dynamic obstacles is greater than the specified threshold T_c if it doesn't change its velocity. The PPO algorithm is adopted to guarantee the stability of the learning and improve the robust performance of our approach. The integration of LSTM can improve the training results and reflect memory characteristics of human beings. The collision prediction algorithm can compute the probability of colliding with dynamic obstacles for agents and make agents learn how to avoid collisions with dynamic obstacles when they move towards their goals. After the training, four leader agents can reach their goals successfully and avoid static and dynamic obstacles. We make each leader agent arrive at a specific goal several times and record its trajectory as the guiding path for the members in its group to follow. Finally, we adopt the RVO algorithm [36] to make agents not collide with others. That is, the RVO algorithm predicts

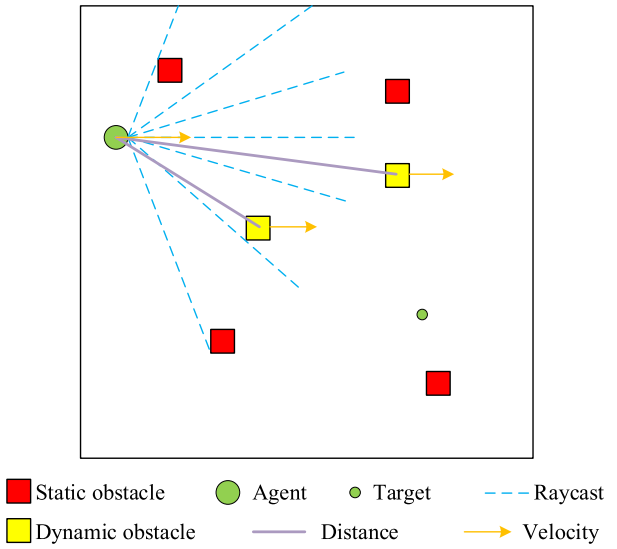

FIGURE 2. State representation.

whether a collision is going to happen at every time step. When there is no collision, the guiding path can lead agents to achieve their goals. However, once there is a predicted collision, the RVO algorithm takes the priority to make agents adapt motions so that the collision is resolved. In that case, our approach can simulate the scenario of several groups reaching their goals successfully in an unknown and dynamic environment.

IV. POLICY REPRESENTATION

A. STATE

As shown in FIGURE 2, a state s consists of features describing the configuration of the environment, the agent as well as its goal. The state of the agent is represented by its position p_a and velocity v_a. The state of the goal is represented by its position p_g. The state of the environment is determined by the existence of dynamic obstacles. If there are no dynamic obstacles, the state of the environment is represented by raycasting detected one of four kinds of possible objects from seven perspectives around agent's forward direction, in which four kinds of objects include agents, walls, static obstacles and goals; else the state of the environment is also represented by the velocities of obstacles and the distances between agents and dynamic obstacles.

B. ACTIONS

The action space is discrete in most of reinforcement learning systems such as Q learning, which makes the motion of the agent unrealistic. With the development of deep reinforcement learning, it becomes possible to deal with the problem with continuous and high-dimensional states and actions. In our approach, the action of agents $a = (v, w)$ is continuous, which is represented by forward movement v and rotation around the y axis w.

C. REWARD

In reinforcement learning, the reward function, $r(s, a, s')$, is used as a training signal to encourage or discourage behaviors. The reward function provides a scalar value reflecting the desirability of a particular state transition that is observed by performing an action a starting in the initial state s and resulting in a successor state s'.

The goal of reinforcement learning is to find a control policy that maximizes the expected value of the cumulative reward RR can be expressed as the time-discounted sum of all transition rewards r_i, from the current action up to a horizon T, in which T may be infinite.

$$R(s_0) = r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ldots + \gamma^i r_i + \ldots \gamma^T r_T \quad (1)$$

in which $r_i = r(s_i, a_i, s_i')$ and γ is a discount factor. $\gamma \in [0, 1)$ ensures that the cumulative reward is bounded.

In our approach, the agent will get the reward if it reaches the goal or moves towards the goal continuously on condition that the probability of colliding with dynamic obstacles is less than the specified threshold T_c. That is, the priority of avoiding the collision with dynamic obstacles is higher than that of moving towards the goal when its probability is greater than the specified threshold T_c. It is in accordance with that the priority of local collision avoidance is higher than that of global path planning. The agent will get the penalty if it collides with the obstacles including static and dynamic obstacles, walls, other leader agents or it spends too much time to search for the goal or the probability of colliding with dynamic obstacles is greater than the specified threshold T_c in which T_c is set to 0.6. The total reward is given by

$$R = r_{goal} + r_{approach} - (r_{c\_static} + r_{c\_dynamic} + r_{c\_walls} + r_{c\_agents} + r_{time} + r_{p\_dynamic}) \quad (2)$$

Note that we differentiate static obstacles from walls since the positions of the walls are fixed while the positions of static obstacles can be changed after the training and no post-processing is needed, which also shows the advantage of our approach Furthermore, we differentiate agents from dynamic obstacles since agents have the ability to avoid the collisions while dynamic obstacles don't have this ability so that agents need to learn how to avoid collisions with dynamic obstacles.

The first reward term checks whether the agent reaches its goal, and gives a reward if it does, in which w_{goal} is its weight.

$$r_{goal} = w_{goal} \quad (3)$$

The second reward term checks whether the agent moves towards its goal, and gives a reward if it does, in which $w_{approach}$ is its weight.

$$r_{approach} = w_{approach} \quad (4)$$

Note that these rewards are given if the probability of colliding with dynamic obstacles is less than the specified threshold T_c. Otherwise, the penalty is given to make agents learn not to collide with dynamic obstacles.

The first penalty term checks whether the agent collides with static obstacles, and gives a penalty if it has occurred, in which w_{c_static} is its weight and j is the number of this kind of collisions

$$r_{c\_static} = \begin{cases} \sum_{j} w_{c\_static} & if \text{ collision occurs} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

The second penalty term checks whether the agent collides with dynamic obstacles, and gives a penalty if it has occurred, in which $w_{c\_dynamic}$ is its weight and k is the number of this kind of collisions

$$r_{c\_dynamic} = \begin{cases} \sum_{k} w_{c\_dynamic} & if \text{ collision occurs} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

The third penalty term checks whether the agent collides with walls, and gives a penalty if it has occurred, in which w_{c_walls} is its weight and m is the number of this kind of collisions

$$r_{c\_walls} = \begin{cases} \sum_{m} w_{c\_walls} & if \text{ collision occurs} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

The fourth penalty term checks whether the agent collides with other agents, and gives a penalty if it has occurred, in which w_{c_agents} is its weight and n is the number of this kind of collisions.

$$r_{c\_agents} = \begin{cases} \sum_{n} w_{c\_agents} & if \text{ collision occurs} \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

The fifth penalty term checks whether the agent spends too much time for reaching its goal, in which w_{time} is its weight.

$$r_{time} = w_{time} \quad (9)$$

The last penalty term checks whether the probability of colliding with dynamic obstacles $p\_dynamic$ is greater than the specified threshold T_c, in which $w_{p\_dynamic}$ is its weight.

$$r_{p\_dynamic} = \begin{cases} w_{p\_dynamic} & if\ p\_dynamic > T_c \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

Note that in different situations, the values of the reward and the penalty are set differently.

D. POLICY REPRESENTATION

Human beings have memory. Agents should also have memory when they explore an unknown environment. We integrate LSTM (Long short-term memory) [37] to reflect memory characteristics of human beings. LSTM is a special type of RNN (Recurrent Neural Network). LSTM is composed of a memory cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTM deals with the exploding and vanishing gradient problems that can be encountered when training

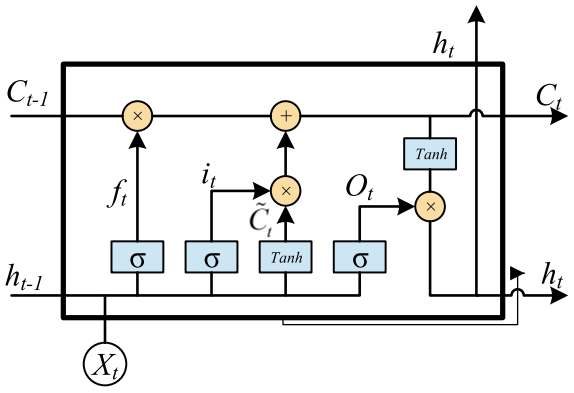

FIGURE 3. LSTM architecture.

traditional RNNs. The architecture of the LSTM unit is shown in FIGURE 3, in which C_t is cell state, h_t is the output, f_t is the forget gate's activation, i_t is the input gate's activation and o_t is the output gate's activation.

The forget gate controls the extent to which a value remains in the cell, and its output is f_t, which is given by:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (11)$$

in which σ is the sigmoid function.

The input gate controls the extent to which a new value flows into the cell, which is given by:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (12)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (13)$$

The old cell state C_{t-1} is updated into the new cell state C_t:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (14)$$

The output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit, which is given by:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (15)$$

$$h_t = o_t * \tanh(C_t) \quad (16)$$

To represent the policy, we use a recurrent neural network that maps a given state s and goal g to a distribution over action. The action distribution is modeled as a Gaussian. The inputs are processed by two hidden layers each composed of 128 LSTM hidden units, followed by a linear output layer. The value function is modeled by a similar network, with the exception of the output layer, which consists of a single linear unit.

V. LEARNING

A. PPO

Proximal Policy Optimization algorithm using the clipped surrogate objective [38] is adopted to train the agents to learn how to reach their goals and avoid collisions with static and dynamic obstacles. Considering that standard policy gradient methods perform one gradient update per data sample, a novel objective function was proposed to enable multiple epochs of

minibatch updates. The new methods, which are called Proximal Policy Optimization (PPO), have some of the benefits of trust region policy optimization (TRPO), but they are much simpler to implement, more general, and have better sample complexity.

In TRPO [39], an objective function is maximized subject to a constraint on the size of the policy update. Specifically,

$$\underset{\theta}{\text{maximize}}\ \hat{E}_t\left[\frac{\pi_\theta\left(a_t\left|s_t\right.\right)}{\pi_{\theta_{old}}\left(a_t\left|s_t\right.\right)}\hat{A}_t\right] \tag{17}$$

$$\text{subject to}\ \hat{E}_t\left[\mathrm{KL}\left[\pi_{\theta_{old}}\left(\cdot\left|s_t\right.\right),\pi_\theta\left(\cdot\left|s_t\right.\right)\right]\right]\le\delta \tag{18}$$

where π_θ is a stochastic policy, θ_{old} is the vector of policy parameters before the update, and $\hat{A}_t$ is an estimator of the advantage function at time step t. With the conjugate gradient algorithm, this problem can be approximately solved efficiently. The theory justifying TRPO actually suggests using a KL penalty instead of a constraint, i.e., solving the unconstrained optimization problem for some coefficient β

$$\underset{\theta}{\text{maximize}}\ \hat{E}_t\left[\frac{\pi_\theta\left(a_t\left|s_t\right.\right)}{\pi_{\theta_{old}}\left(a_t\left|s_t\right.\right)}\hat{A}_t - \beta\mathrm{KL}\left[\pi_{\theta_{old}}\left(\cdot\left|s_t\right.\right),\pi_\theta\left(\cdot\left|s_t\right.\right)\right]\right] \tag{19}$$

Let $r_t(\theta)$ denote the ratio of the probability, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, thus $r_t(\theta_{old}) = 1$. TRPO maximizes a "surrogate" objective:

$$L^{CPI}(\theta) = \hat{E}_t\left[\frac{\pi_\theta\left(a_t\left|s_t\right.\right)}{\pi_{\theta_{old}}\left(a_t\left|s_t\right.\right)}\hat{A}_t\right] = \hat{E}_t\left[r_t(\theta)\hat{A}_t\right] \tag{20}$$

The superscript CPI refers to conservative policy iteration [40], for the reason that maximization of L^{CPI} would lead to an excessively large policy update without a constraint, a novel objective function was proposed as follows and adopted as the new variant of PPO:

$$L^{CLIP}(\theta) = \hat{E}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ \mathrm{clip}\left(r_t(\theta),1-\varepsilon,1+\varepsilon\right)\hat{A}_t\right)\right] \tag{21}$$

Here, epsilon is a hyperparameter, say, $\varepsilon = 0.2$. In this way, we only ignore the change in probability ratio when it would make the objective improve, and we include it when it makes the objective worse. This objective implements a way to do a Trust Region update which is compatible with Stochastic Gradient Descent, and simplifies the algorithm by removing the KL penalty and need to make adaptive updates. This algorithm has been proved to display the best performance on continuous control tasks and almost matches ACER's performance on Atari, despite being far simpler to implement.

In our approach, PPO is based on the actor-critic algorithm in which actor attempts to maximize J_{PPO} while critic attempts to minimize L_{BL}, in which J_{PPO} is the expected sum of rewards and L_{BL} is loss function for updating the critic and furthermore, both of them are given in Algorithm 1. We maintain two networks, one for the policy π_θ and another for the value function V_ϕ, with parameters θ and ϕ respectively. The value function is updated using the temporal difference computed with the λ-return results in the TD (λ) algorithm. Similarly, the policy is updated using gradients computed from the surrogate objective, with advantages $\hat{A}_t$ computed using GAE (λ). Empirically, it has been proved that the λ-return shows better performance than simply using the $n$step return. The concrete learning process is illustrated in **Algorithm 1** in which M, B is the number of sub-iterations with policy and baseline updates given a batch of datapoints. Policy updates are performed after a batch of $N = 1024$ samples has been collected. $\lambda = 0.95$ is used for both TD (λ) and GAE (λ).

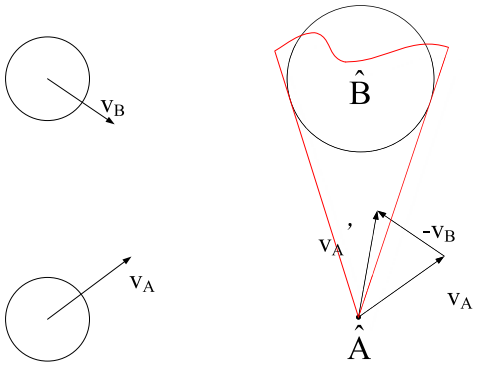

FIGURE 4. The collision cone CC_{AB}.

TABLE 1. Parameters.

Parameters	Value
Learning rate	0.0003
Batch size	1024
Sequence length	64
Memory size	256
λ	0.95
γ	0.99
ε	0.2
w_{goal}	(100, 120)
$w_{approach}$	(5, 10)
w_{c_static}	(-20, -40)
w_{c_walls}	-5
w_{c_agents}	(-20, -30)
w_{time}	(-5, -10)
$w_{p\_dynamic}$	(-20, -40)
Dimension of the environment without dynamic obstacles	48
Dimension of the environment with dynamic obstacles	62

B. COLLISION PREDICTION

Agents should avoid collisions with the dynamic obstacles when they are moving in an unknown environment. Inspired by the concept of Velocity Obstacle (VO) [41], we compute the collision probability between the agent and the dynamic obstacle. The higher the collision probability is, the greater the penalty the agent will get. Consider the two circular objects, A and B, shown in FIGURE 4 at time t_0, with velocities v_A and v_B and radii r_A and r_B. Let A represent the agent, and B represent the dynamic obstacle. To compute the VO, we map B into the Configuration Space of A, by reducing

FIGURE 5. The process of four leader agents learning about the environment.

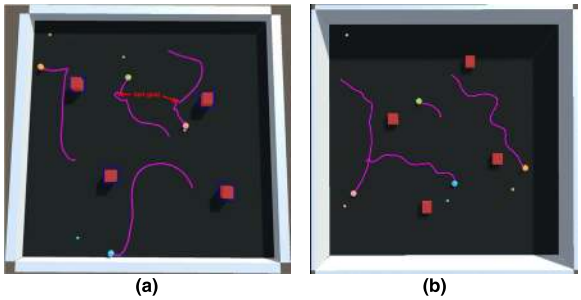

FIGURE 6. Four leader agents reach their goals successfully after the training.

Algorithm 1 Proximal Policy Optimization

for $i \in \{1, \cdots, N\}$ **do**
  Run policy π_θ for T timesteps, collecting $\pi_\theta \{s_t, a_t, r_t\}$
  Estimate advantages $\hat{A}_t = \sum_{t'>t} \gamma^{t'-t} r_{t'} - V_\phi(s_t)$
  $\pi_{\theta_{old}} \leftarrow \pi_\theta$
  for $j \in \{1, \cdots, M\}$ **do**
    $J_{PPO}(\theta) = \sum_{t=1}^{T} \min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t\right)$
    Update θ by a gradient method w.r.t. $J_{PPO}(\theta)$
  end for
  for $j \in \{1, \cdots, B\}$ **do**
    $L_{BL}(\phi) = -\sum_{t=1}^{T} \left(\sum_{t'>t} \gamma^{t'-t} r_{t'} - V_\phi(s_t)\right)^2$
    Update ϕ by a gradient method w.r.t. $L_{BL}(\phi)$
  end for
end for

A to the point $\hat{A}$ and enlarging B by the radius of A (r_A) to $\hat{B}$ We define the Collision Cone, CC_{AB}, as the set of colliding relative velocities between $\hat{A}$ and $\hat{B}$:

$$CC_{AB} = \{v'_A | \lambda'_A \cap \hat{B} \neq \phi\} \tag{22}$$

in which $v'_A = v_A - v_B$ and λ'_A is the line of v'_A. This cone is the planar sector with the apex in $\hat{A}$, bounded by the two tangents from $\hat{A}$ to $\hat{B}$ as shown in red in FIGURE 4. Any relative velocity that lies in this cone will cause a collision between A and B. Furthermore, the collision probability between agent A and dynamic obstacle B is also related to the time to the-time-to-collision (TTC), which is given:

$$t = \frac{\hat{A} - p_C}{v'_A} \tag{23}$$

where p_C is the intersection point between v'_A and $\hat{B}$.

The smaller t is, the higher the collision probability is. Therefore, the collision probability between agent A and dynamic obstacle B is set as follows:

$$p_c = \begin{cases} \min(1, \frac{1}{t}) & \text{if } v'_A \in CC_{AB} \\ 0 & \text{if } v'_A \notin CC_{AB} \end{cases} \tag{24}$$

Note that, we use the circumscribed circle of the cube to compute the collision probability since the static and dynamic obstacles are represented by cubes.

VI. RESULTS

All the experiments were run on a desktop with an NVIDIA GeForce 1080 Ti GT graphics card and an Intel Core 2 i7-8700K CPU (3.7GHZ) with 16 GB memory. Our virtual scenes are constructed through the Unity3D Engine. Parameters for the simulation and deep reinforcement learning are summarized in TABLE I.

FIGURE 5 shows the process of four leader agents learning how to avoid collisions with static obstacles, in which static obstacles are represented by cubes in red, walls are represented by the frames in white, four leader agents are represented by larger spheres in different colors and each goal is represented by a smaller sphere in the same color as the leader agent, leader agents' trajectories are represented by lines in deep pink and the final state representation is 48-dimensional. FIGURE 5(a) and FIGURE 5(b) show that four leader agents explore the environment randomly at the beginning of the training since they don't know the environment at all. FIGURE 5(c) shows that four leader agents can reach their goals while avoiding collisions with static obstacles through getting the reward when they approach their goals and the penalty when they collide with the obstacles and other leader agents, however, their paths to goals are longer and twisted in the middle phase of the training. FIGURE 5(d) shows that four leader agents can reach their goals while

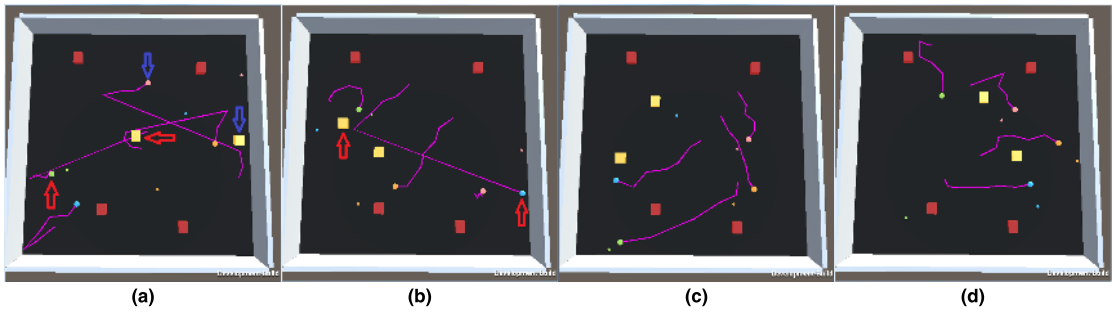

FIGURE 7. The process of four leader agents learning how to avoid dynamic obstacles.

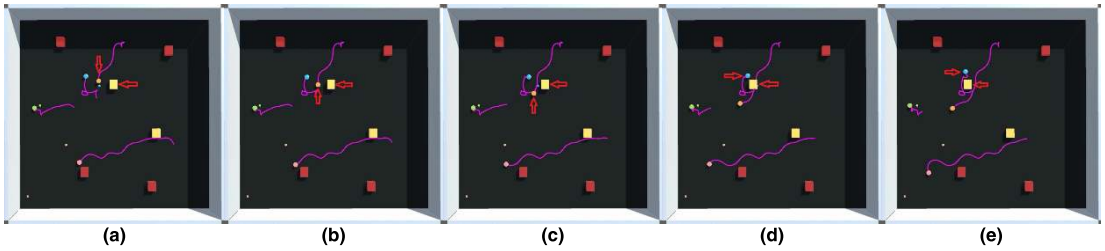

FIGURE 8. The process of four leader agents avoiding collisions with dynamic obstacles successfully.

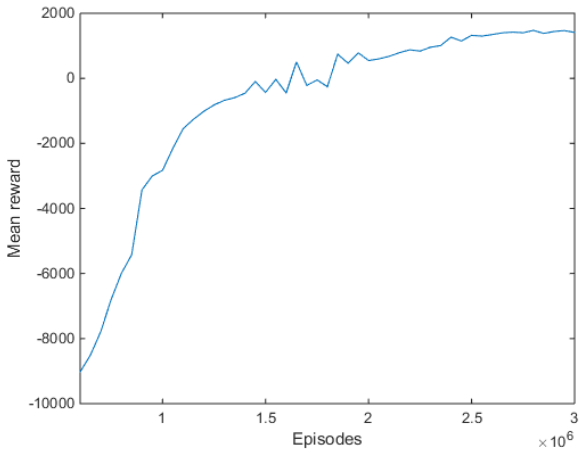

FIGURE 9. The learning curve in an unknown and dynamic environment.

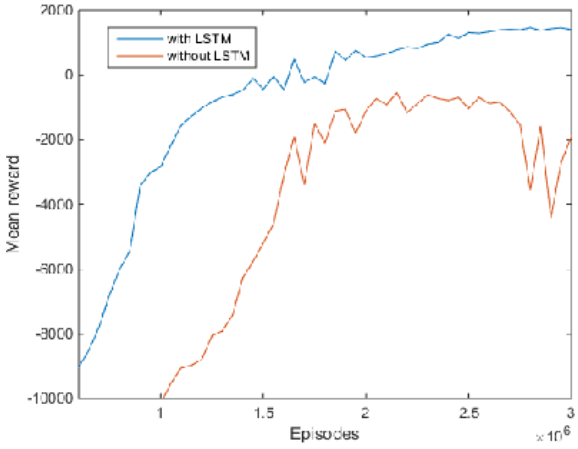

FIGURE 10. The comparison of learning curves with and without LSTM.

avoiding collisions with static obstacles in a much shorter path in the late phase of the training.

In FIGURE 6(a), the static obstacles have been bounded by bold lines in blue after the training is finished and four leader agents have learned how to avoid collisions with static obstacles well. As a result, four leader agents find smooth paths to reach their goals successfully while not colliding with static obstacles or the other leader agents. Note that the breakpoints in the trajectories of the leader agents shown in green and pink are the last goals for themselves. In FIGURE 6(b), when the positions of static obstacles change, no training is needed and four leader agents can also find smooth paths to reach their goals successfully while not colliding with static obstacles or the other leader agents.

FIGURE 7 shows the process of four leader agents learning how to avoid collisions with dynamic obstacles, in which dynamic obstacles are represented by cubes in yellow and they are moving back and forth, and correspondingly, the final state representation is 62-dimensional. In FIGURE 7(a) and 7(b), four leader agents don't know how to avoid dynamic obstacles so that collisions happen between leader agents and dynamic obstacles at the beginning of the training. The red arrows and dark blue arrows label the collision pairs. FIGURE 7(c) and 7(d) show that four leader agents know how to avoid collisions with dynamic obstacles through getting penalties when the collision probability is greater than the specified threshold T_c or they collide with dynamic obstacles.

FIGURE 8 shows the process of four leader agents avoiding collisions with dynamic obstacles successfully. In FIGURE 8(a), 8(b) and 8(c), agent shown in orange moves towards its goal and when the collision probability with the dynamic obstacle is high, it will speed up to avoid the collision. In FIGURE 8(d) and 8(e), the goal of the agent shown

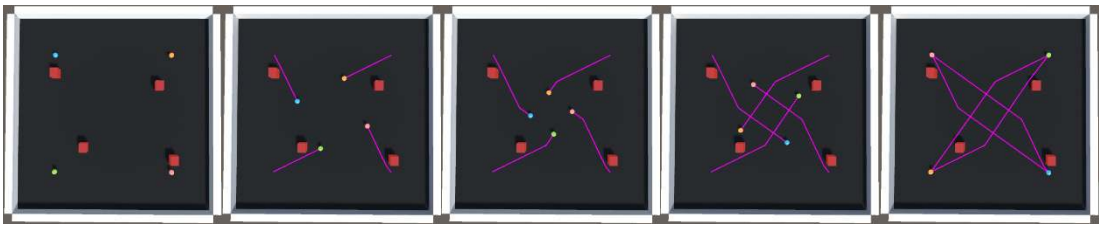

FIGURE 11. The process of four leader agents moving towards their goals.

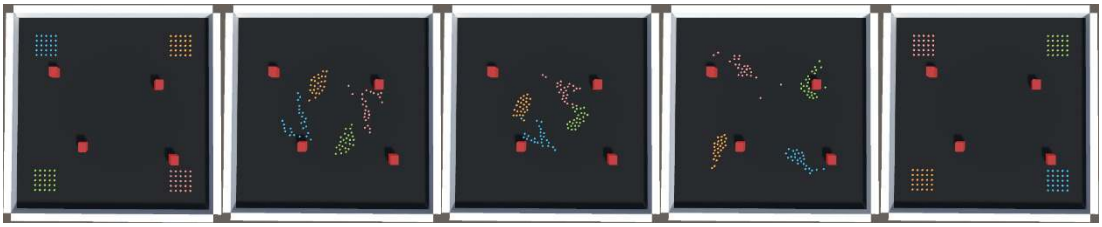

FIGURE 12. The process of four groups of agents moving towards their goals.

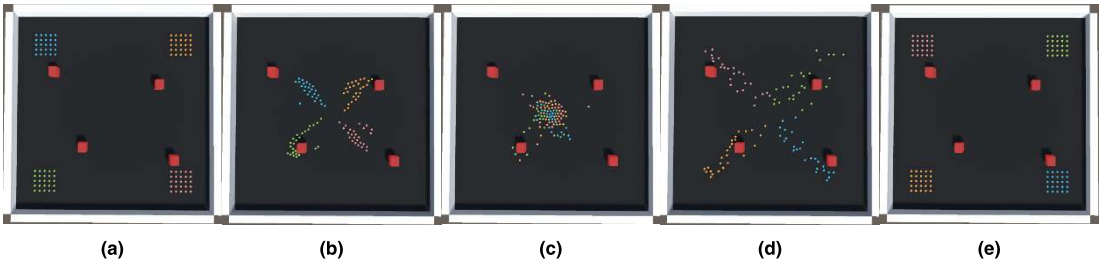

FIGURE 13. The process of four groups of agents moving towards their goals using navigation mesh combined with RVO.

in blue is very close to a dynamic obstacle and therefore, the agent shown in blue changes its moving direction to avoid the collision. Note that the agent and the dynamic obstacle who are going to collide are labeled by the red arrows.

In FIGURE 9, we evaluate the performance by measuring the mean reward the agents get every 2000 episodes. From FIGURE 9, we can see that the agents get higher rewards with the increase of the iterations. The curve rises with a fast rate at the beginning of the training; the curve rises with a slower rate when the episodes reach 1500k; and finally, the curve becomes steady when the episodes reach 2500k.

In FIGURE 10, we evaluate the performance by measuring the mean reward the agents get every 2000 episodes without and with LSTM. From FIGURE 10, we can see that the agents get higher reward with LSTM than without LSTM, which demonstrates the advantage of our approach.

FIGURE 11 shows the process of four leader agents moving towards their goals. FIGURE 11(a) shows the initial positions of four leader agents and the goal of each leader is the opposite of the corner in a diagonal line. In FIGURE 11(b), 11(c) and 11(d), each leader agent approaches its goal continually and avoids collisions with other leader agents and static obstacles. In FIGURE 11(e), each leader agent reaches its goal successfully and the trajectory of each leader agent has been recorded as the guidance path.

FIGURE 12 shows the process of four groups of agents moving towards their goals based on the trajectories recorded in FIGURE 11, in which the group members are represented in the same color as their leader. FIGURE 12(a) shows the initial positions of four groups of agents and the goal of each group is the opposite of the corner in the diagonal lines where they are situated. In FIGURE 12(b) and 12(c), each group of agents moves along the leader agent's trajectory while avoiding collisions with other groups of agents and furthermore, there is no congest at the center of the environment and the agents in the same group show the flocking behavior. In FIGURE 12(d), each group of agents moves towards their goal continually after the collisions are resolved. FIGURE 12(e) shows that four groups of agents reach their goals successfully finally.

In TABLE II, we do the comparisons among classical path planning approaches and our approach. Compared with other works related to global path planning and local collision avoidance, we solve the problem of navigating in an unknown and dynamic environment based on PPO combined with the LSTM and a collision prediction algorithm. To be more specifically, compared with traditional approaches for

TABLE 2. Comparisons among classical path planning approaches and our approach.

	Environment is known or unknown	Positions and number of static obstacles can change	Can avoid collisions with dynamic obstacles	Reach the goal	Action space
Global path planning	known	yes, but the graph must be re-generated	no	yes	continuous
Local collision avoidance	unknown or partially known	yes	yes	Sometimes it may get stuck and cannot find the goal	continuous
Reinforcement learning	unknown	yes, no training is needed	yes	yes	discrete
Our approach	unknown	yes, no training is needed	yes	yes	continuous

global path planning, which plan the paths according to the established graphs, such as navigation graphs[12], probabilistic maps[13], coarse graph-based roadmaps[14], [15], our approach can make agents reach their goals without knowing the environment in prior. That is, the prerequisite for global path planning is that the environment is known and the graphs are generated before planning while agents can learn how to reach their goals and avoid collisions with static obstacles in our approach and we don't need to establish the graph for the environment. Furthermore, when the environment changes, such as the positions of static obstacles change, our approach doesn't need to train agents again and can make agents arrive at their goals in collision-free paths while traditional global path planning approaches need to regenerate a new graph to guarantee that agents don't collide with static obstacles and reach their goals. Compared with traditional approaches for local collision avoidance, such as geometrically-based algorithms [5], grid-based methods [6], force-based methods [7], [8], our approach can make agents avoid collisions with dynamic obstacles through getting the reward and the penalty. Compared with the work focusing on navigation based on Q-learning, we make the action space continuous and generate more smooth paths while the paths generated by Q learning are often jagged since its action space is discrete. Compared with the work focusing on crowd simulation based on deep reinforcement learning, we make agents reach any random or specified goals successfully after the training, which shows the scalability of our approach. Furthermore, the recorded trajectory as the guidance path makes four groups of agents approach their goals in an uncongested way. As a comparison, we run the same experiment using the approach with no guidance paths which combines navigation mesh as global path planning with RVO algorithm as local collision avoidance, and obtain the result shown in FIGURE 13, in which four groups of agents congest at the center of the environment and the same group of agents scatter out and cannot show the flocking behavior.

VII. CONCLUSION

We present a novel approach for solving crowd navigation in an unknown and dynamic environment. The proposed method is inspired by the deep reinforcement learning technique. We state that global path planning and local collision avoidance cannot deal with the problem of navigating in an unknown and dynamic environment and furthermore, most of reinforcement learning systems such as Q learning cannot produce realistic motions due to its discrete action space. Therefore, we first combine PPO with LSTM and a collision prediction algorithm to train four leader agents to learn how to reach their goals and avoid collisions with static and dynamic obstacles, and then we make each leader agent arrive at a specific goal several times and record its trajectory as the guiding path so that the members in its group know how to reach their assigned goals. Finally, we adopt the RVO algorithm to make agents not collide with others and furthermore, we simulate the scenario of four groups moving towards their goals simultaneously. We evaluated our approach and demonstrated the achieved improvements as follows: 1) our approach can make agents navigate successfully in an unknown and dynamic environment based on deep reinforcement learning; 2) our approach can plan smooth paths for crowds since the action space is continuous; 3) our approach can reflect the characteristics of human beings including self-learning ability, the communication and the memory; 4) our approach can train multi-agents with different goals simultaneously and obtain good results. The main objective of our future work is to extend our method to deal with navigating articulated characters in a complex environment and develop a more general and improved navigation model considering the personality, the emotion and the mood of individual agents for crowd simulation. That is why we don't compare the results derived from our proposed navigation approach with real human crowd motions.

REFERENCES

[1] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, pp. 487–490, Sep. 2000.
[2] S. Chenney, "Flow tiles," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, Aug. 2004, pp. 233–242.
[3] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM Comput. Graph.*, vol. 21, no. 4, pp. 25–34, Jul. 1987.
[4] S. R. Musse and D. Thalmann, "Hierarchical model for real time simulation of virtual human crowds," *IEEE Trans. Vis. Comput. Graphics*, vol. 7, no. 2, pp. 152–164, Apr./Jun. 2001.
[5] O. B. Bayazit, J.-M. Line, and N. M. Amato, "Better group behaviors in complex environments using global roadmaps," in *Proc. 8th Int. Conf. Artif. Life*, Dec. 2002, pp. 362–370.
[6] R. Narain, A. Golas, S. Curtis, and M. C. Lin, "Aggregate dynamics for dense crowd simulation," in *Proc. ACM SIGGRAPH Asia Papers*, Dec. 2009, pp. 1–8.
[7] A. Sud, R. Gayle, E. Andersen, S. Guy, M. Lin, and D. Manocha, "Real-time navigation of independent agents using adaptive roadmaps," in *Proc. ACM Symp. Virtual Reality Softw. Technol.*, Nov. 2007, pp. 99–106.
[8] P. Charalambous, I. Karamouzas, S. J. Guy, and Y. Chrysanthou, "A data-driven framework for visual crowd analysis," *Comput. Graph. Forum.*, vol. 33, no. 7, pp. 41–50, Oct. 2014.

[9] R. A. Metoyer and J. K. Hodgins, "Reactive pedestrian path following from examples," *Vis. Comput.*, vol. 20, no. 10, pp. 635–649, Dec. 2004.
[10] C. D. Boatright, M. Kapadia, J. M. Shapira, and N. I. Badler, "Generating a multiplicity of policies for agent steering in crowd simulation," *Comput. Animation Virtual Worlds*, vol. 26, no. 5, pp. 483–494, Sep./Oct. 2015.
[11] S. Singh, M. Kapadia, G. Reinman, and P. Faloutsos, "Footstep navigation for dynamic crowds," *Comput. Animation Virtual Worlds*, vol. 22, nos. 2–3, pp. 151–158, Apr./May 2011.
[12] 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*, Nov. 2005, p. 194.
[13] M. Sung, L. Kovar, and M. Gleicher, "Fast and accurate goal-directed motion synthesis for crowds," in *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, Jul. 2005, pp. 291–300.
[14] O. B. Bayazit, J. M. Lien, and N. M. Amato, "Better group behaviors in complex environments using global," *Artif. Life*, vol. 8, no. 8, p. 362, 2003.
[15] S. Rodriguez and N. M. Amato, "Roadmap-based level clearing of buildings," in *Motion Games* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2011, pp. 340–352.
[16] R. Oliva and N. Pelechano, "Automatic generation of suboptimal navmeshes," in *Motion Games* (Lecture Notes in Computer Science). Berlin, Germany: Springer, 2011, pp. 328–339.
[17] L. Sun, L. Ding, and W. Qin, "Planning feasible and smooth paths for simulating realistic crowd," in *Intelligent Computing Theories Methodologies* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2015, pp. 498–509.
[18] A. Sud, E. Andersen, S. Curtis, M. Lin, and D. Manocha, "Real-time path planning for virtual agents in dynamic environments," in *Proc. IEEE Virtual Reality Conf.*, Mar. 2007, pp. 91–98.
[19] M. Kapadia, A. Beacco, F. Garcia, V. Reddy, N. Pelechano, and N. I. Badler, "Multi-domain real-time planning in dynamic environments," in *Proc. 12th ACM SIGGRAPH/Eurographics Symp. Comput. Animation*, Jul. 2013, pp. 115–124.
[20] W. D. Smart and L. P. Kaelbling, "Effective reinforcement learning for mobile robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2002, pp. 3404–3410.
[21] F. Martinez-Gil, M. Lozano, and F. Fernández, "Multi-agent reinforcement learning for simulating pedestrian navigation," in *Proc. Int. Workshop Adapt. Learn. Agents*. Berlin, Germany: Springer, 2011, pp. 54–69.
[22] L. Casadiego and N. Pelechano, "From one to many: Simulatin groups of agents with RL controllers," in *Proc. Int. Conf. Intell. Virtual Agents*, Aug. 2015, pp. 119–125.
[23] L. Khriji, F. Touati, K. Benhmed, and A. Al-Yahmedi, "Mobile robot navigation based on Q-learning technique," *Int. J. Adv. Robot. Syst.*, vol. 8, no. 1, pp. 45–51, Jan. 2011.
[24] S. Li, X. Xu, and L. Zuo, "Dynamic path planning of a mobile robot with improved Q-learning algorithm," in *Proc. IEEE Int. Conf. Inf. Autom.*, Aug. 2015, pp. 409–414.
[25] 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.
[26] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," Aug. 2017, *arXiv:1708.05866*. [Online]. Available: https://arxiv.org/abs/1708.05866
[27] 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, Feb. 2015.
[28] 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, Jan. 2016.
[29] M. Campbell, A. J. Hoane, Jr., and F.-H. Hsu, "Deep blue," *Artif. Intell.*, vol. 134, nos. 1–2, pp. 57–83, Jan. 2002.
[30] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, and C. Welty, "Building watson: An overview of the deep QA project," *AI Mag.*, vol. 31, no. 3, pp. 59–79, 2010.
[31] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[32] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May/Jun. 2017, pp. 3357–3364.
[33] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," Nov. 2016, *arXiv:1611.05763*. [Online]. Available: https://arxiv.org/abs/1611.05763
[34] J. Lee, J. Won, and J. Lee, "Crowd simulation by deep reinforcement learning," in *Proc. 11th Annu. Int. Conf. Motion, Interact., Games*, Nov. 2018, Art. no. 2.
[35] X. Peng, G. Berseth, K. Yin, and M. Van De Panne, "DeepLoco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Trans. Graph.*, vol. 36, no. 4, Jul. 2017, Art. no. 41.
[36] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Conf. Robot. Autom.*, May 2008, pp. 1928–1935.
[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 2012.
[38] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," Jul. 2017, *arXiv:1707.06347*. [Online]. Available: https://arxiv.org/abs/1707.06347
[39] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," in *Proc. 32nd Int. Conf. Mach. Learn.*, vol. 37, Jul. 2015, pp. 1889–1897.
[40] S. Kakade, J. Langford, "Approximately optimal approximate reinforcement learning," in *Proc. ICML*, vol. 2, Jul. 2002, pp. 267–274.
[41] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, Jul. 1998.

LIBO SUN received the Ph.D. degree from the School of Computer Science and Technology, Tianjin University, in January 2012. She was a Visiting Scholar of the Graphics Laboratory, University of Pennsylvania, from November 2009 to August 2011, where she was a Postdoctoral Researcher, from December 2015 to January 2017. She is currently an Associate Professor with the School of Instrument Science and Engineering, Southeast University, China. Her research interests include computer animation, virtual reality, and crowd simulation.

JINFENG ZHAI received the bachelor's degree from the School of Instrument Science and Engineering, Southeast University, China, in 2017, where she is currently pursuing the master's degree. Her research interests include computer animation, virtual reality, and crowd simulation.

WENHU QIN received the Ph.D. degree from the School of Instrument Science and Engineering, Southeast University, in 2005, where he is currently a Professor. He has more than 30 journal papers, 10 conference papers, and a book. He has five patents. His research interests include vehicle safety, virtual reality, crowd simulation, and road traffic accident reconstruction.