Automatic Parameter Tuning via Reinforcement Learning for Crowd Simulation with Social Distancing

Yiran Zhao and Roland Geraerts

Abstract— **Reinforcement learning (RL) has been applied to a variety of fields such as gaming and robot navigation. We study the application of RL in crowd simulation by proposing an automatic parameter tuning system based on Proximal Policy Optimization (PPO). The system can be used with any crowd simulation software to improve the quality of the simulation by automatically assigning parameters to each agent during the simulation. Our experiments indicate that the automatic parameter tuning system can reduce unexpected congestions in counterflow scenarios. In addition, by utilizing the improved commonly used crowd simulation algorithms and our parameter tunning system, we can represent social distancing behavior of pedestrians under COVID-19, where pedestrians comply to the suggested social distance when they have enough space to move while they reduce their social distances to others when there is limited space.**

I. Introduction

In the past decade, many crowd simulation algorithms have been developed to enhance the anticipation of movement and to come closer to real pedestrian behaviors, such as Density-based path planning [1], Optimal Reciprocal Collision Avoidance (ORCA) [2], Implicit Crowds [3] and C-nav [4]. To construct a complete crowd simulation, several algorithms need to be combined together. From global path planning to local collision avoidance, a plethora of parameters need to be tuned carefully according to different scenarios. A coarse selection of parameters may yield inflexible and unrealistic behaviors of pedestrian-agents. In most crowd simulation softwares, a long list of parameters is left for the user to determine, which is difficult to understand without professional experiences.

However, even with professional knowledge of the algorithms, it is hard to manually choose the best parameter combination for every simulation at each simulation step. For example, the social distance is a crucial factor for crowd simulation under COVID-19. In real life, pedestrians have initiatives to obey the social distance rule such as maintaining 1 m to 2 m distance in most countries. But the degree of compliance differs from their surrounding situations. Pedestrians tend to reduce their social distances from others when they see a high crowd density [5] [6] or counterflow [7]. Unfortunately, due to lack of a robust social distance simulation model, most simulations either completely ignore the active social avoidance or treat the social distance as a stiff constant under all situations [8] [9] [10] [11] [12], which may not represent the flexible pedestrian behavior correctly.

The authors are with the Department of Information and Computing Sciences at Utrecht University, Princetonplein 5, Utrecht, the Netherlands. Email: zhaoyiran182@gmail.com, r.j.geraerts@uu.nl

Inspired by reinforcement learning methods applied in local motion planning in recent years [13] [14] , we propose an auto-parameter tuning approach based on the Proximal Policy Optimization algorithm [15]. This approach can assign parameters, including the social distance preference, for the simulated agents dynamically according to their goals, positions, velocities and surrounded crowd densities. By utilizing reinforcement learning, the approach can assign parameter values to the simulated agents at each assigning step (1 second). In contrast to the existing parameter fitting methods [16] [17] [18], which only give one global and static optimal parameter set, our method offers dynamic parameter fitting for each agent constantly. Such a flexibility is beneficial for tunning some sensitive parameters such as "personal space" in the collision avoidance algorithm, which may vary at different places in one scenario. Our experiments indicate, compared to simulations with manually assigned parameters, that our approach generates more energy-efficient behaviors [19] with less travelling time and less congestions.

Our main contributions are: 1) a general deep reinforcement learning framework to automatically and dynamically tune the parameters for any simulation scenario, giving more intelligent simulation results, and 2) a specific model for simulating the social distancing behavior of pedestrians under COVID-19, based on Density-based path planning [1], Optimal Reciprocal Collision Avoidance (ORCA) [2], Social Groups Navigation (SGN) [20] and our automatic parameter tuning system.

II. Related work

Crowd simulation is a technique to simulate real pedestrians walking via controlling motions of virtual agents, which has been applied in flow control, risk evaluation and gaming fields. To improve the collision anticipation and to reduce the unexpected congestions, the Density-based path planning algorithm [1] and the ORCA algorithm [2] have been proposed. The combination of these two methods gives convincing motions that are smooth and collision-free. Social grouping is another important factor in crowd simulation. Family members, couples and friends walking together usually form a social group. A social group model, SGN, proposed by Jaklin et al. [20], simulates groups in both global path-planning and local collision-avoidance levels. The algorithm shows socially-friendly group navigation, with flexibility of group member movement.

Several papers related to simulating crowds under the COVID-19 pandemic have been published. In papers [10] [11] and [12], the authors estimate the social exposure via

considering the physical contact and the distances between simulated agents. These methods have a common limitation in which the initiative of social avoidance is not taken into account. In the pandemic time, most people try to keep a safe social distance from others [6]. The above models may lose correctness to simulate the pedestrians movements under COVID-19. On the other hand, some simulations such as [8] and [9] adopt a constant social distance such as 1 m or 1.5 m, ignoring the social distance preferences of pedestrians under different situations, while pedestrians intend to reduce their social distances when they encounter a dense crowd [5] [6] or a counterflow [7] in real life.

Reinforcement learning is developing rapidly in recent years. There are several algorithms of path planning and collision avoidance based on reinforcement learning have been proposed. Socially Adaptive Path Planning [14] was developed for a robotic vehicle planning socially adaptive paths in dynamic environments. This was done by learning an optimal trajectory by following experts with an inverse reinforcement learning procedure, and integrating the local plan to the global planed path. The IMARL [21] algorithm gives a combination of a multi-agent reinforcement learning process and an improved social-force model. However, the existing approaches always focus on only one aspect of steering. They can be advanced alternatives for collision avoidance or path planning, but they are difficult to integrate into a complete crowd simulation framework. For setting up a complex simulation, a user still needs to tune the parameters carefully.

Our method gives a general system for training a scenario by automatic and dynamic parameter tuning, which can be easily combined with existing crowd simulation frameworks. Without knowing the details of each algorithm of the crowd simulation, our method can find a proper solution for a scenario that optimizes the travelling time of agents, i.e., the time that agents need to reach their goals. Among the various reinforcement learning algorithms, we select Proximal Policy Optimization (PPO) [15] as the training policy for our automatic parameter tunning system because it can deal with the continuous state and action spaces, updates the policy quickly and shows general feasibility.

III. Automatic parameter tuning sytem

In this section, we introduce our automatic parameter tuning system. The system is a centralized system, which gathers information of all simulated agents and gives a global optimal solution. At each parameter assigning step (usually 1 second), the system assigns a set of parameters for each simulated agent according to its properties and surrounded environment. Any tunable parameter in the simulation can be trained with this system, such as the agent speed and the agent visual field. We show an application example in Section IV, where the parameter of planning density weight and a parameter related to collision avoidance radius are trained to simulate the social distancing behavior realistically.

The automatic parameter tuning system is modelled as a standard reinforcement learning problem, where in a step t, state s_t of each simulated agent is sent to the system as input, and a mapped policy $\pi_q(a_t|s_t)$ for each tunable crowd simulation parameter q is calculated by the system and then sent back to the agent. The policy $\pi_q(a_t|s_t)$ represents a probability density function of action a_t given s_t. Applying a sampled value from the policy to the parameter, the system responds with a new state s_{t+1} and an immediate reward r_{t+1} in the next step $t+1$. The goal of the training is that the system maximizes the cumulative reward of the corresponded parameter, called return R, via improving a policy gradually.

Let's denote the set of all tunable parameters in the crowd simulation as Q. For each tunable parameter $q \in Q$, we use an independent network to represent π_q and an independent memory buffer to store collected states, actions, rewards, etc. Instead of applying the policies to the agents every simulation execution step L (typically 0.1 second), the system only runs the policies with a specified period L_{run} (usually 1 second), where L_{run} is a multiple of L. The policies are trained by using a policy-gradient method, named PPO [15], in the end of each episode with period $L_{episode}$ ($L_{episode} > L_{run}$), and then the simulation restarts. The general structure of the system is shown in Algorithm 1.

Algorithm 1 Automatic parameter tuning

```
t ← L_run
while simulation not end do
    if t ≥ L_episode then
        for all q in Q do
            Calculate advantages
            Optimize policy π_q
            Clear replay buffer B_q
        end for
        Restart simulation
        t ← 0
    end if
    if t mod L_run = 0 then
        for all q in Q do
            Run policy π_q
            Update replay buffer B_q
        end for
    end if
    Perform simulation
    t ← t + 1
end while
```

A. State and Action

The state s_t represents the configuration of an agent and the information of its surrounding environment at step t. Considering the common features among most crowd simulation frameworks, state s_t of an agent is composed of the present position $\mathbf{p_t}$, the goal position $\mathbf{g_t}$, the velocity $\mathbf{v_t}$, the surrounding crowd density d_t, the number of surrounding neighbors o_t, and the values of all tunable parameters $k_{q,t}$, where the widely ranged variables such as positions are standardized to $[-1, 1]$.

The states of agents are shared between tunable parameters policy networks. The action $a(s_t)$, sampled from the policy $\pi_q(a_t|s_t)$ specifies the scalable value of parameter q for state s_t. For example, the value of the speed q_{sa} of an agent ranges from 0 to the maximum speed s_0. When we require a value of q_{sa} for state s_t, we sample a value $a(s_t)$ from the beta distribution of the q_{sa} policy (ranged from 0 to 1), and then assign $a(s_t) \cdot s_0$ to the parameter q_{sa}.

B. Reward and Advantage estimator

Reward $r(s_t, a(s_t), s_{t+1})$ represents the result of transferring from state s_t to s_{t+1} via action $a(s_t)$ at step t. According to the principle of least effort, the simulated pedestrians should avoid unnecessary detours and travel with short and fast paths [22] [23]. The simplest reward function is $r(s_t, a(s_t), s_{t+1}) = -1$ when one agent does not reach its goal. It can encourage a shorter travelling time from the departure position to the destination position of an agent. We can also take the social collision into account, as discussed in Subsection IV-B.

With rewards of agents in an episode, we can estimate the advantage $\mathcal{A}_t$ of taking an action $a(s_t)$ at step t, which is a measure of how much is $a(s_t)$ a good or bad decision compared with the expected return of s_t. That can be done by the equations in reference [15].

C. Neural network

A standard actor-critic architecture of the network is applied in our approach [24]. For each tunable parameter q, we build two networks. In both networks, the states are processed with two fully-connected layers with 512 and 256 units, respectively. The ReLu activation function is used following each fully-connected layer. The output layer of the critic network has only one unit, which of the actor network contains one or two units. The policy distribution can be a Gaussian distribution ($a \in (-\infty, \infty)$), a gamma distribution ($a \in (0, \infty)$) or a beta distribution ($a \in (0, 1)$). Empirically, both α and β (μ and σ) in the Gaussian and beta distribution can be trained, while we only train α in gamma distribution and set β to constant 1.0.

D. Replay buffer and Learning Policy

Compared with a single robot or agent, which has been intensively studied in the Robotics field, a plethora of agents in a crowd simulation offers abundant experience to learn within an episode. Since tracking the trajectories of every agent is too expensive, we limit the replay buffer size to N_c and only track the first N_c agents in the scene. Therefore, the replay buffer is naturally divided into maximum N_c batches, where each batch contains data of one agent during the episode. At step t, the system applies the policy of each tunable parameter q to each agent. The state s_t, the action $a(s_t)$, the reward r_t, the reached goal flag η_t, the output value of the critic network $V(s_t)$, and the logarithm of the action probability density $\rho(a(s_t))$ are collected and pushed into the corresponding batch in the memory buffer.

The policies are trained by the PPO method, with a clipped surrogate objective. In each training step, we optimize the critic and actor networks with batches in the replay buffer. The loss function of the i^{th} iteration is the combination of the expectation of the critic loss, the actor loss and the entropy, as explained by [15]. When optimizing the loss, the policy gradually approaches the optimum and the estimation of a state becomes increasingly precise.

IV. Simulating the social distancing

Our goal is simulating the pedestrian behavior under a pandemic. While pedestrians are aware of complying to the social distance and still want to reach their goals as soon as possible, no approach can model the behavior to our knowledge. A much larger personal space than normal undoubtedly increases the difficulty of the simulation since unexpected congestions are easier to occur. For reducing congestions and modelling social groups, we apply and modify the following algorithms: Density-based path planning [1], ORCA [2] and SGN [20]. On the basis of the algorithms, we leverage the automatic parameter tuning system to train the parameters with two rewards: reaching the goal in a short time and complying to the social distance.

A. Representing the social distancing

For agent A, ORCA uses the sum of the body radii ($R_A + R_B$) of agent A and B to construct the velocity obstacle between the agents. With the velocity obstacle, ORCA can select the best velocity for A to avoid collisions. Moreover, ORCA also uses the body radius to check the existing collisions and gives a velocity to evacuate from the collision. In our simulation, we replace $R_A + R_B$ with $R_A + \psi_A + R_B$, where ψ_A is the social distance preference of A. So agent A can try to avoid "social collision" or try to get out of the "social collision" with B. The replacement is only applied when A and B are not in the same social group. In such a group, the members can always stay closer than other pedestrians and do not have the extra collision-avoidance distance. In our implementation, the social coherence is done by SGN. It should be noted that ψ_A is the social distance preference of A rather than the suggested social distance Ψ (like 1.5 m and 2 m). The social distance preference ψ is a trained parameter in the automatic parameter tuning system, with $\psi \in (0, \Psi)$.

B. Training the simulation

We train the social distance preference ψ and the density weight ω via the automatic parameter tuning system and keep other parameters constant. As explained in Subsection IV-A, the social distance preference ψ is the extra collision-avoidance distance for an agent at a moment. We want this value to be as close as possible to the suggested social distance Ψ, and keep flexibility for the agent to pass through a crowded area in a short time. Otherwise, if all agents insisted on complying to the suggested social distance, the path would be completely blocked and no one could pass. The policy of ψ is a beta distribution in which we train both α and β. Then a sampled value $a(s_t) \in (0, 1)$ from the distribution can be used to represent $\psi(s_t)$ via $\psi(s_t) =$

$\Psi a(s_t)$. We consider both the travelling time and the social collision in the reward function:

$$r(s_t, a(s_t), s_{t+1}) = -1 - \xi \cdot o(s_t), \tag{1}$$

where ξ is a constant during the training, representing the social distance awareness of the crowd. A high value of ξ yields the crowd complying to the social distance, while a low value of ξ represents a small collision avoidance radius so the agents have more freedom to move. $o(s_t)$ is the number of visible (usually represented as a 180-degree view cone) neighbors who locate inside the social radius of the agent. In this equation, -1 encourages an agent to reach its goal in a shorter time, and $-\xi \cdot o(s_t)$ encourages the agent to keep a social distance from its visible neighbors.

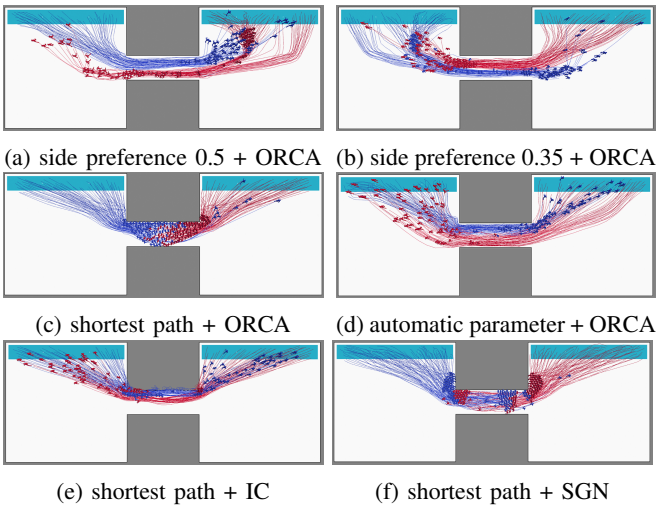
Except for the social distance preference, we train the policy of the density weight as well. The density weight ω is a parameter in the Density-based path planning method. When the simulator plans a global path for an agent on the navigation mesh, it calculates the cumulative cost of edges with ω, then the path with the minimum cost will be selected. We adopt a gamma distribution with trainable α and fixed $\beta = 1$ for the density weight parameter.

V. Results

A. Counterflow

Counterflow is a typical pedestrian movement pattern. Literally, pedestrians moving in opposite directions form a counterflow, with spontaneous features such as lane-formation [25]. It is adopted as a benchmark scenario in the evacuation verification standard [26]. Compared with a one-directional movement scenario, a counterflow scenario is more challenging for a collision-avoidance method. Plenty of collision-avoidance algorithms leverage a counterflow scenario to verify their robustness [22] [27] [28].

We built a counterflow scenario which contains two square rooms (25m x 25m) connected with a bridge of 5m width. The start and goal positions are sampled randomly from two areas inside the rooms. After the simulation starts, 100 simulated pedestrians in each room move towards the opposite room with given pairs of start and goal positions. Based on the calculated global navigation mesh, we can either adopt the shortest path or the side preference route selection algorithm. The former finds the shortest path for each agent, but in the counterflow scenario, it results in an expected congestion because every agent selects the upper part of the bridge. The side preference method relies on the shape of each cell in the navigation mesh. A side preference value in the range $[-1, 0)$ yields agents preferring the left side and a value in the range $(0, 1]$ represents the right side. The side preference method can help to reduce the congestion on the bridge. However, agents need to detour in the rest part of the scenario. With the automatic parameter tuning system, we can make agents move with short paths in the rooms and form bidirectional lanes on the bridge, via only training the side preference parameter. The policy distribution of the side preference parameter is a beta distribution with trainable α and β, and the episode is set to 180 seconds.



(a) side preference 0.5 + ORCA (b) side preference 0.35 + ORCA

(c) shortest path + ORCA (d) automatic parameter + ORCA

(e) shortest path + IC (f) shortest path + SGN

Fig. 1: Trajectories at 40 s in the counterflow scenario in the six sub-scenarios. A blue trajectory represents an agent moving from Area A to Area B. As a contrast, a red trajectory is generated from an agent moving from Area B to Area A.

To compare the simulation results, six sub-scenarios were constructed. The first four sub-scenarios apply ORCA algorithm, i.e., (a) with the side preference path planning and a constant preference value of -0.5; (b) with the side preference path planning and a constant preference value of 0.35; (c) with the shortest path planning; (d) with the side preference path planning and dynamic preference values produced by the trained policy (trained by 5000 episodes). Sub-scenario (e) and sub-scenario (f) adopt the shortest path planning, while the former applies the Implicit Crowds (IC) collision avoidance [3] and the latter applies the Social Groups and Navigation (SGN) collision avoidance [20].

We ran each sub-scenario for 50 times and get the mean travelling time of the agents. The mean travelling time in second of the six sub-scenarios are 58.85, 55.99, 83.01, 45.05, 52.01 and 297.89, respectively. The trajectories at 40 s are shown in Figure 1. The result indicates the shortest travelling time and an emergent lane-formation agent behavior with the automatic parameter method, which combines the advantages of the shortest path and the side preference methods. This allows agents to pass through the bridge quickly and to follow short paths as well.

In conclusion, the automatic parameter tuning method brings variable side preference to the simulation, helping to solve the typical counterflow problem of the crowd simulation.

B. Crowd with social distancing

In this subsection, we simulated a small scenario that people cross the bridges to reach the opposite land (Figure 2). There is an obstacle in the left side and a pair of obstacles shaped as a gate in the right side. In the scenario, 80 agents are generated from Area A with a generation rate of 10 agents per second. Then they move to Area B and finally reach Area C. The generation and goal positions are randomly sampled from the areas. Among the 80 agents, 40

agents form 2-person groups and the remainder move individually. The goal of an agent is reaching the goal position as soon as possible, keeping a social distance from others and minimizing the social collision. Agents are expected to comply to the social distance of 1.0 m most of time while squeezing their social spaces in the narrow bridges to shorten the travelling time.

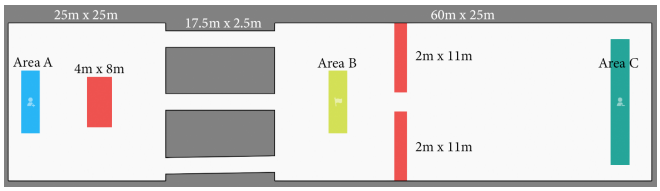


Fig. 2: Layout of the social distancing scenario

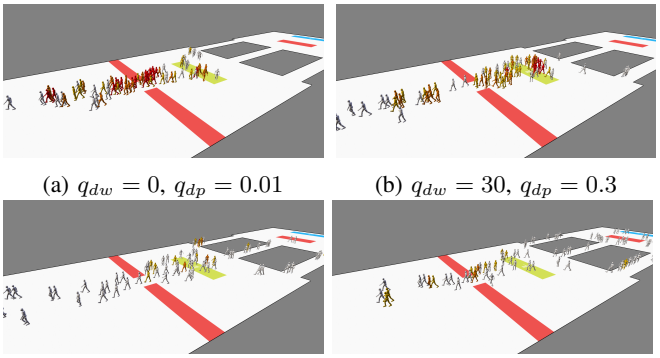
We trained both the density weight and the social distance preference parameter for the scenario for 5000 episodes. The policies of the density weight and the social distance preference are explained in Section IV-B, in which the social distance awareness ξ, an extra punishment of the social distance preference reward, is set to 1.0 and the episode is set to 240 s.

We compared two types of simulations. One simulation type uses the automatic parameters while another simulation type uses constant parameters. Both types of simulations apply ORCA. There are 9 scenarios constructed with the constant parameters. To measure the effect of social distancing, we counted the social collision times by summing the number of all neighbors of each agent, who are visible for the agent and socially collided with the agent, during the whole simulation. We ran each simulation 50 times. The constant density weight is noted as q_{dw}, and the constant social distance preference is noted as q_{dp}. In Table I, we show the mean travelling time and the mean social collision times in the constant parameter scenarios, where we observe that the travelling time increases as the social distance preference increases, and the social collision times is never smaller than 50000. However, with the automatic parameters, the mean travelling time is 102.01 seconds and the mean social collision times is only 32368.1, giving the agents a reasonable balance between shortening the travelling time and complying to the social distance. The screenshots of the simulations are presented in Figure 3 and Figure 4, where the agents who have zero, one, two, and at least three social collisions with their visible neighbors are colored in white, yellow, orange, and red, respectively. Figure 3 shows the comparison between simulations with three sets of constant parameters and the simulation with automatic parameters at 60 s. It can be observed that the scenario with automatic parameters have less social collisions. Figure 4 shows the difference between the simulation with a big social distance preference and the one with automatic parameters at 90 s. In the former simulation, agents form an unexpected congestion in front of the gate-shaped obstacles, while in the latter simulation, agents do not have the trouble of passing through the obstacles. Because of the dynamic parameter selection, our approach can find a nearly optimal solution under different crowd situations.

	$q_{dw} = 0$	$q_{dw} = 30$	$q_{dw} = 100$
$q_{dp} = 0.01$	84.95 / 143676	82.49 / 116530	84.89 / 89101.4
$q_{dp} = 0.3$	91.63 / 96196.1	87.25 / 74596.2	90.31 / 69715.4
$q_{dp} = 0.8$	105.57 / 54737.2	123.11 / 77020	118.25 / 57904

TABLE I: Simulation results of combination between the constant density weight and the constant social distance preference. The number on the left of a cell is the mean travelling time in seconds and the number on the right of a cell is the mean social collision times.

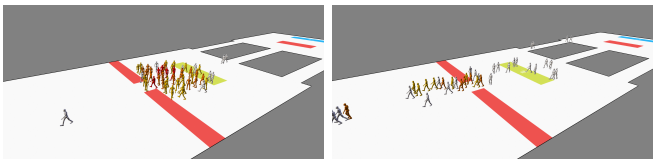
Our experiment reveals, via the automatic parameter tuning, that the simulated social distancing is more energy-efficient than only using the constant parameters. It can simulate the initiative of pedestrians to comply to the social distance, as well as violating the social distance when the crowd density is high.



(a) $q_{dw} = 0$, $q_{dp} = 0.01$ (b) $q_{dw} = 30$, $q_{dp} = 0.3$

(c) $q_{dw} = 100$, $q_{dp} = 0.8$ (d) automatic parameters

Fig. 3: Agents in the social distancing scenarios at 60s.



(a) $q_{dw} = 100$, $q_{dp} = 0.8$ (b) automatic parameters

Fig. 4: Agents in the social distancing scenarios at 90s.

VI. Conclusion

We presented an automatic parameter tuning system for applying dynamic parameters in crowd simulation, based on the reinforcement learning algorithm Proximal Policy Optimization (PPO). Such a framework can easily be embedded in a complex crowd simulation software, solving the problem of unexpected congestions and reducing the travelling time. Moreover, to simulate the social distancing behavior of crowds under the COVID-19 pandemic, we extended existing crowd simulation algorithms, i.e., Density-based path planning, Optimal Reciprocal Collision Avoidance (ORCA) and Social Groups and Navigation (SGN).

Next, we combined the algorithms with the automatic parameter tuning system. Experiments indicated that the automatic parameter tuning method can represent the crowd behavior more energy-efficient than the traditional constant parameter methods, in respect of resolving congestions in counterflow and simulating social distancing.

Although the experiments illustrated the ability of the framework of enhancing the crowd simulation performance, there are still some limitations: 1) the method has only been validated via small crowd simulation scenarios, but not big scenarios with thousands of agents; 2) training multiple policies with different reward functions may yield an unstable training process which requires a long converging time; 3) for training the social distance preference, the hyperparameter social distance awareness should be wisely chosen. A too big or too small value will lead to long training time.

Our main objective of future work is representing the state of an agent in a more abstract way and increasing the universality of the trained policies. It would be interesting that we can train a policy for one scenario and apply the same policy for other scenarios. Besides, we wish to validate the simulation result via the automatic parameter tuning method by real-world experiments.

REFERENCES

[1] W. G. Van Toll, A. F. Cook IV, and R. Geraerts, "Real-time density-based crowd simulation," *Computer Animation and Virtual Worlds*, vol. 23, no. 1, pp. 59–69, 2012.

[2] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Optimal reciprocal collision avoidance for multi-agent navigation," in *Proc. of the IEEE International Conference on Robotics and Automation, Anchorage (AK), USA*, 2010.

[3] I. Karamouzas, N. Sohre, R. Narain, and S. J. Guy, "Implicit crowds: Optimization integrator for robust crowd simulation," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–13, 2017.

[4] J. Godoy, S. J. Guy, M. Gini, and I. Karamouzas, "C-nav: Distributed coordination in crowded multi-agent navigation," *Robotics and Autonomous Systems*, vol. 133, p. 103631, 2020.

[5] L. S. Liebst, P. Ejbye-Ernst, M. de Bruin, J. Thomas, and M. R. Lindegaard, "Mask-wearing and social distancing: Evidence from a video-observational and natural-experimental study of public space behavior during the covid-19 pandemic," 2021.

[6] C. A. Pouw, F. Toschi, F. van Schadewijk, and A. Corbetta, "Monitoring physical distancing for crowd management: Real-time trajectory and group analysis," *PloS one*, vol. 15, no. 10, p. e0240963, 2020.

[7] T. Blanken, C. Tanis, F. Nauta, F. Dablander, B. Zijlstra, R. Bouten, Q. Oostvogel, M. Boersma, M. van der Steenhoven, F. van Harreveld *et al.*, "Smart distance lab: A new methodology for assessing social distancing interventions," 2020.

[8] S. Comai, S. Costa, S. M. Ventura, G. Vassena, L. Tagliabue, D. Simeone, E. Bertuzzi, G. Scurati, F. Ferrise, and A. Ciribini, "Indoor mobile mapping system and crowd simulation to support school reopening because of covid-19: a case study," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 44, pp. 29–36, 2020.

[9] Á. G. García, A. Caciagli *et al.*, "Crowd control in plazas constrained to social distancing," *arXiv preprint arXiv:2005.07038*, 2020.

[10] T. Harweg, D. Bachmann, and F. Weichert, "Agent-based simulation of pedestrian dynamics for exposure time estimation in epidemic risk assessment," *arXiv preprint arXiv:2007.04138*, 2020.

[11] E. Ronchi and R. Lovreglio, "Exposed: An occupant exposure model for confined spaces to retrofit crowd models during a pandemic," *Safety Science*, vol. 130, p. 104834, 2020.

[12] Y. Xiao, M. Yang, Z. Zhu, H. Yang, L. Zhang, and S. Ghader, "Modeling indoor-level non-pharmaceutical interventions during the covid-19 pandemic: a pedestrian dynamics-based microscopic simulation approach," *arXiv preprint arXiv:2006.10666*, 2020.

[13] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1343–1350.

[14] B. Kim and J. Pineau, "Socially adaptive path planning in human environments using inverse reinforcement learning," *International Journal of Social Robotics*, vol. 8, no. 1, pp. 51–66, 2016.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[16] G. Berseth, M. Kapadia, B. Haworth, and P. Faloutsos, "Steerfit: Automated parameter fitting for steering algorithms," in *Simulating Heterogeneous Crowds with Interactive Behaviors*. AK Peters/CRC Press, 2016, pp. 229–246.

[17] D. Wolinski, S. J. Guy, A.-H. Olivier, M. Lin, D. Manocha, and J. Pettré, "Parameter estimation and comparative evaluation of crowd simulations," in *Computer Graphics Forum*, vol. 33, no. 2. Wiley Online Library, 2014, pp. 303–312.

[18] A. Binch, G. P. Das, J. P. Fentanes, and M. Hanheide, "Context dependant iterative parameter optimisation for robust robot navigation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3937–3943.

[19] G. K. Zipf, *Human behavior and the principle of least effort: An introduction to human ecology*. Ravenio Books, 2016.

[20] N. Jaklin, A. Kremyzas, and R. Geraerts, "Adding sociality to virtual pedestrian groups," in *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology*, 2015, pp. 163–172.

[21] 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. 73 841–73 855, 2019.

[22] I. Karamouzas, P. Heil, P. Van Beek, and M. H. Overmars, "A predictive collision avoidance model for pedestrian simulation," in *International workshop on motion in games*. Springer, 2009, pp. 41–52.

[23] S. J. Guy, J. Chhugani, S. Curtis, P. Dubey, M. C. Lin, and D. Manocha, "Pledestrians: A least-effort approach to crowd simulation." in *Symposium on computer animation*, 2010, pp. 119–128.

[24] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2018.

[25] T. Kretz, A. Grünebohm, M. Kaufman, F. Mazur, and M. Schreckenberg, "Experimental study of pedestrian counterflow in a corridor," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2006, no. 10, p. P10001, 2006.

[26] E. Ronchi, E. D. Kuligowski, P. A. Reneke, R. D. Peacock, and D. Nilsson, *The process of verification and validation of building fire evacuation models*. US Department of Commerce, National Institute of Standards and Technology ..., 2013.

[27] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[28] Y. Tajima, K. Takimoto, and T. Nagatani, "Pattern formation and jamming transition in pedestrian counter flow," *Physica A: Statistical Mechanics and its Applications*, vol. 313, no. 3-4, pp. 709–723, 2002.