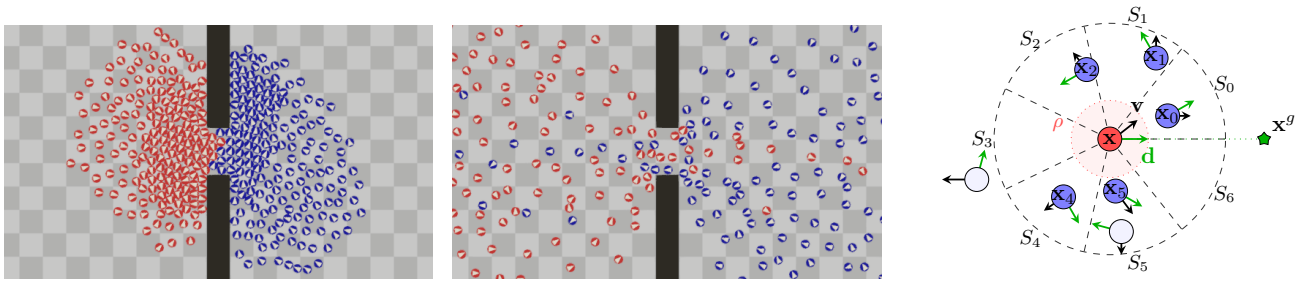
STAG: Smart Tools and Applications in Graphics (2025)
F. Maggioli, C. Romanengo, D. Cabiddu, and D. Giorgi (Editors)

Poster

Learning to Wait: Preventing Global Congestion from Local Observations in Real-Time Crowd Navigation

Irena Ruprecht[1], Florian Michelic[2] and Reinhold Preiner[1]

[1] Graz University of Technology, Institute of Visual Computing, Austria
[2] Fragment Garage, Independent Game Studio, Austria



(a) (b) (c)

Figure 1: *450 RL-trained agents in a bottleneck scenario after 60 s (red: move left to right, blue: vice versa). (a) Without waiting, agents cluster and get stuck. (b) With waiting, agents maintain flow and prevent congestion. (c) Each agent (red) observes its own state (velocity* $\mathbf{v}$*, distance to goal* $\mathbf{x}^g$*) and, for each sector (*S_0*–*S_6*), the nearest neighbor's relative position (*$\mathbf{x}_k - \mathbf{x}$*), velocity (*$\mathbf{v}_k$*), and desired direction (*$\mathbf{d}_k$*).*

Abstract

We present a real-time crowd simulation approach based on reinforcement learning (RL), addressing congestion prevention in confined spaces. We learn a local navigation policy that uses compact, fast-to-compute per-agent observations of a small set of neighbors, including their desired directions. Alongside goal progress and inter-agent spacing, we reward agents for waiting when neighbors ahead pursue similar goals. This formulation fosters global self-organization from purely local interactions. Preliminary results show reduced congestion and consistent goal attainment for large crowds with hundreds of agents.

CCS Concepts

• ***Computing methodologies*** → ***Real-time simulation; Multi-agent reinforcement learning;***

1. Introduction and Related Work

Real-time crowd simulation is important in games and immersive applications, where many agents move simultaneously. Agents typically balance goal-directed motion with collision avoidance. Social forces [HM95] work well in many cases, but high densities in confined spaces may still cause congestion and prevent progress. Addressing such global flow issues efficiently requires per-agent navigation decisions based on fast-to-compute information.

In previous work [RMEP24], we detect congestion from compact local neighbor sets, enabling agents to wait in response. In contrast, learning-based approaches train navigation policies directly. Data-driven models imitate real trajectories [ZLH*22], producing realistic behaviors but dataset-limited strategies. Reinforcement learning (RL) adapts agents via trial-and-error [LWL18], sometimes guided by real data [CPV*23]. However, most RL formulations focus on continuous goal pursuit and collision avoidance [LWL18] or energy efficiency [KKPC23]. Hence, congestion-preventive behaviors usually emerge only incidentally. We train agents to wait strategically using RL with compact, sector-based observations [RMEP24]. Our reward balances goal progress, inter-agent spacing, and waiting, enabling agents to effectively prevent congestion.

2. Method

Agents are trained using actor-critic Proximal Policy Optimization (PPO). The policy receives observations and outputs two scalar actions $\in [-1, 1]$, used as 2D inputs to a force-based motion controller.

DOI: 10.2312/stag.20251341

Observations. As shown in Fig. 1c, each agent observes its own state - velocity $\mathbf{v}$ and scalar distance to the goal $\mathbf{x}^g$ - and the closest neighbor in each sector [RMEP24], defined by relative position, velocity, and desired direction. Sectors are aligned with the agent's desired direction $\mathbf{d}$ for rotation-invariant learning. All vectors input to the network are normalized and their magnitude is passed as separate scalar $\in [0,1]$. Velocities $\mathbf{v}$ are scaled by the agent's maximum speed and distances by the observation radius.

Reward. The agent's reward depends on its situation. *At the goal* ($g = 1$), it is rewarded for proximity to the goal position (R^{g}) and for standing still (R^{w}). *En route* ($g = 0$), it receives a spacing reward (R^{s}), considering the nearest of N detected neighbors, and full R^s if $N = 0$; if a neighbor with a similar desired direction is detected ahead ($w = 1$), it is rewarded for slowing or standing (R^{w}), otherwise ($w = 0$) for goal-directed motion with a nonlinear speed bonus (R^{p}). Formally, the reward is defined as follows:

$$R^{\mathrm{s}} = \min_{k \in N} \|\mathbf{x}_k - \mathbf{x}\|, \quad R^{\mathrm{p}} = \left(\frac{\hat{\mathbf{v}} \cdot \mathbf{d} + 1}{2}\right)^2 \left(1 - (1 - \|\mathbf{v}\|)^4\right),$$
$$R^{\mathrm{w}} = 1 - \|\mathbf{v}\|, \qquad R^{\mathrm{g}} = 1 - \|\mathbf{x}^g - \mathbf{x}\|,$$

$$R = \min(R^{\mathrm{w}}, R^{\mathrm{g}})\, g + \Big(\min(R^{\mathrm{s}}, R^{\mathrm{w}})\, w + \min(R^{\mathrm{s}}, R^{\mathrm{p}})\,(1 - w)\Big)(1 - g),$$

where $\hat{\mathbf{v}}$ is the normalized velocity; $g = 1$ if $\|\mathbf{x}^g - \mathbf{x}\| < 0.2$m; $w = 1$ if a neighbor k triggers waiting by $\mathbf{d}_k \cdot \mathbf{d} > 0.9$ (similar desired direction) and $\frac{\mathbf{d}_k + \mathbf{d}}{\|\mathbf{d}_k + \mathbf{d}\|} \cdot \frac{\mathbf{x}_k - \mathbf{x}}{\|\mathbf{x}_k - \mathbf{x}\|} > 0.5$ (is ahead). Distances are normalized by the maximum observation range, spacing by a radius ρ (Fig. 1c), and velocities $\mathbf{v}$ by maximum speed, yielding sub-rewards $\in [0,1]$. Scaling R by a constant 0.01 improves training stability.

Training Setup. Agents are trained with Unity ML-Agents using PPO (key hyperparameters in Tab. 1). Simulations run at 50 FPS (0.02 s per step) with a 2-step (0.04 s) decision period, during which actions are held constant. Each 50 s episode involves 32 agents in one of two scenarios: crowd-passing, where two opposing blocks of 16 agents swap positions (with or without a bottleneck); and circle-crossing, where agents repeatedly move between opposing points.

Parameter	Value and description
Batch size	512 — updates per gradient step
Buffer size	10,240 (≈ 410 s of experience)
Learning rate	3×10^{-4} — linear decay
Clipping ϵ	0.2 — PPO clipping threshold (linear decay)
GAE λ	0.95 — advantage estimation
Discount γ	0.99 — future reward weighting
Entropy coeff. β	0.02 — encourages exploration (linear decay)
Network	3×128 — hidden layers/units, shared critic
Time horizon	128 (≈ 5.1 s trajectory segment)
Max steps	40M — total training experience

Table 1: *Key PPO hyperparameters.*

3. Evaluation and Results

We study the impact of the waiting reward R^{w} via ablation. Two reward variants are compared: a baseline combining only spacing and goal progress ($R' = \min(R^{\mathrm{s}}, R^{\mathrm{p}})$) and the full reward including strategic waiting (Section 2). Crowds of 450 agents are tested in passing scenarios distinct from training, with and without bottlenecks. Evaluation results in Fig. 2 use deterministic inference, tracking average speed, inter-agent spacing, and goal achievement.

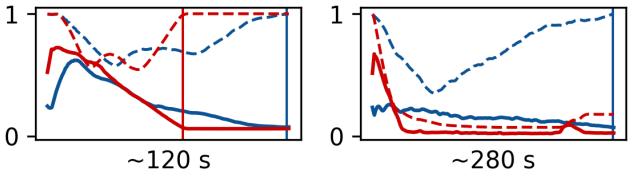


Figure 2: *Average speed (solid) and inter-agent spacing (dashed), normalized to* $[0,1]$ *(speed scaled by max, spacing by preferred distance ρ), for no waiting (red) and waiting (blue) over time in crowds passing scenarios without (left) and with a bottleneck (right). Vertical lines mark last agent arrivals.*

In the open scenario (no bottleneck, Fig. 2, left), the waiting behaviors (blue) can slow agents and delay goal arrival, while producing smoother changes in speed and spacing, resulting in a more consistent and organized crowd. In the bottleneck scenario (Fig. 2, right), non-waiting agents (red) form dense clusters and get stuck (Fig. 1a, no vertical line). Waiting agents (blue) maintain greater spacing and achieve steady throughput (Fig. 1b). Without waiting, speed rises sharply and both speed and spacing drop abruptly at the bottleneck; with waiting, speed and spacing evolve steadily, preventing congestion and allowing all agents to reach their goals.

Discussion and Limitations. Strategic waiting prevents congestion and improves spacing, though it can slow goal achievement in open spaces. It produces smoother, more consistent crowd behavior as speed and distances evolve steadily. The policy generalizes to larger crowds and novel scenarios, but its performance in fully complex virtual worlds remains untested, and the policy does not model human-like motion or realistic crowd behaviors.

4. Conclusion and Future Work

We show that agents can prevent congestion using only fast-to-compute local sector-based observations and a waiting-based reward. This enables reliable goal attainment and steadier crowd behavior compared to simple goal-spacing rewards. Future work will extend this to other social behaviors (yielding, overtaking and flocking) and integrate the crowd into a complex immersive world.

References

[CPV*23] C. P., Pettre J., Vassiliades V., Chrysanthou Y., Pelechano N.: Greil-crowds: Crowd simulation with deep reinforcement learning and examples. *ACM Trans. Graph. 42*, 4 (July 2023). 1

[HM95] Helbing D., Molnár P.: *Social force model for pedestrian dynamics*. 05 1995. 1

[KKPC23] Kwiatkowski A., Kalogeiton V., Pettré J., Cani M.: Reward function design for crowd simulation via reinforcement learning. In *SIGGRAPH MIG* (2023), ACM. 1

[LWL18] Lee J., Won J., Lee J.: Crowd simulation by deep reinforcement learning. In *SIGGRAPH MIG* (2018), ACM. 1

[RMEP24] Ruprecht I., Michelic F., Eggeling E., Preiner R.: Adaptive movement behavior for real-time crowd simulation. *The Visual Computer 40* (06 2024). 1, 2

[ZLH*22] Zhong J., Li D., Huang Z., Lu C., Cai W.: Data-driven crowd modeling techniques: A survey. *ACM Trans. Model. Comput. Simul. 32*, 1 (Jan. 2022). 1