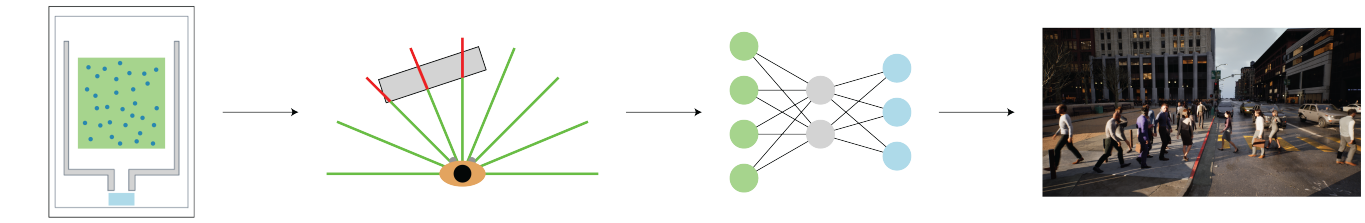
Dynamic Crowd Routing: RL-Driven Crowd Dynamics

Daniele Della Pietra*, Nicola Garau*†, Nicola Conci*†, Fabrizio Granelli*†
*University of Trento, †CNIT - Consorzio Nazionale Interuniversitario per le Telecomunicazioni
Via Sommarive 14, Trento, 38123 (Italy)
{daniele.dellapietra}@studenti.unitn.it, {nicola.garau, nicola.conci, fabrizio.granelli}@unitn.it



a) Scenario definition b) Sensing and Rewards assignment c) Policy Training d) Simulation and Rendering

Fig. 1: we present Dynamic Crowd Routing (DCR), a RL-based method for crowd dynamics. Our full pipeline starts with (a) the definition of the map of the initial configuration of an arbitrarily large scenario. Each agent (b) is equipped with 180° sensing rays that capture observations from the nearby environment, including dynamic obstacles and pedestrians. *No goal information is given at any time to any of the agents*, promoting the exploration of the environment. We train (c) a very shallow neural network using the PPO algorithm to optimize a combination of carefully crafted rewards. Finally, (d) we render a photorealistic simulation of the crowd during inference, in real-time.

Abstract*—The simulation of crowds is complex and challenging. Every individual in a crowd exhibits a different behaviour, targets a different goal, and undergoes different types of interactions. Within crowds, groups can be identified in both static and dynamic configurations, with varying levels of responsiveness, leading to the emergence of complex avoidance mechanisms. In the past, rule-based models have been proposed to simulate crowds, unlocking the potential for large-scale simulations. Over the years, learning-based solutions have been presented, achieving acceptable results despite the lack of high-quality ground truth data for training. While both rule-based and learning-based methods have recently been integrated into 3D simulation engines, they usually rely on navigation meshes or B-spline functions, hindering their generalization to open-world scenarios. In this work, we propose a reinforcement learning-based solution to learn meaningful crowd dynamics inside the Unreal Engine 3D engine, enabling massive and highly dynamic crowd simulations. We show how our proposed method makes it possible to simulate crowd setups that require complex dynamic routing mechanisms, which are otherwise hard to achieve using rule-based approaches or even deep learning-based methods. Our approach also allows us to easily collect large synthetic datasets that are both photorealistic and provide accurate ground truth data without the need for any manual annotation. *Some demonstration videos are available at mmlab-cv.github.io/DynamicCrowdRouting; Code, complete experiments and analysis will be made available upon acceptance.

***Index Terms*—Digital Twins, Simulation, Rendering, Reinforcement Learning, 3D Animation**

I. Introduction and Motivation

Pedestrian dynamics is a critical area of study, focusing on the movement and interactions of individuals inside a crowd. The diverse behaviours, goals, and social rules of each individual are guided by complex mechanisms that give rise to intricate and intrinsically dynamic group formations and avoidance mechanisms.

Historically, rule-based models have been the most reliable way of simulating crowd behaviour. Perhaps the most influential one is the *Social Force Model* (SFM) [1], which simulates pedestrian movement through simple attractive and repulsive forces. Similarly, *Reciprocal Velocity Obstacles* (RVO) [2] tries to prevent pedestrian collisions by adjusting their velocity reciprocally.

More recently, learning-based approaches have emerged, leveraging large datasets to infer behavioural patterns. These methods, however, face challenges such as the scarcity of high-quality ground truth and limited generalization abilities, when applied outside of the training environment. In fact, despite their potential, they often struggle to adapt to novel, open-world scenarios commonly encountered in real-life simulations.

In this work, we propose a reinforcement learning-based approach to simulate crowd dynamics. This method aims to enhance the scalability and adaptability of crowd simulations, enabling complex dynamic routing mechanisms that are

difficult to achieve with traditional rule-based or even deep learning-based solutions. Furthermore, our proposal facilitates the generation of large, photorealistic synthetic datasets, providing valuable ground truth data without manual annotation. The method has been integrated within Unreal Engine using the Learning Agents[1] framework.

II. Related Work

Pedestrian dynamics approaches can be roughly subdivided into two main categories: model-based and model-free. In the next subsection, we will give an overview of the related work concerning the two, as well as a dedicated section detailing existing pedestrian simulators and frameworks. For a more comprehensive overview of crowd and pedestrian dynamics, please refer to [3].

A. Model-based crowd modelling

Model-based approaches for crowd modelling have been fundamental for the development of meaningful pedestrian dynamics simulators, taking inspiration from complex motion typical of the natural world [4], and are still widely adopted today in both research and commercial applications. They typically use physical and social forces to represent the interactions between pedestrians and their environment, relying on predefined rules and equations to simulate pedestrian behaviour. The Social Force Model (SFM) [1] simulates pedestrian movements by applying attractive and repulsive forces, which drive individuals toward their goals while avoiding collisions. The Reciprocal Velocity Obstacles (RVO) method [2] enhances this approach by incorporating velocity adjustments, effectively preventing collisions in real-time multi-agent systems.

While both the SFM and the RVO focus on modelling attractive and repulsive forces, other approaches try to ensure a collision-free and guaranteed-to-terminate simulation, which is not always the case for force-based methods. For example, the Collision-free Speed Model (CFSM) [5] adjusts the pedestrians' speed to ensure intrinsically collision-free trajectories. However, it can't be used to model the interactions between pedestrians and obstacles explicitly. The authors in [6] solve this issue by building upon the Collision-free Speed Model, incorporating wall influences and ellipse occupancy regions for calculating distances.

B. Model-free crowd modelling

Model-free crowd modelling methods usually rely on huge annotated datasets to infer a possible underlying behavioural model, without explicitly defined rules. To learn the underlying distribution, they often rely on deep learning approaches such as neural networks. Once a good representation of the behavioural model has been learned, it is possible to predict or simulate new trajectories based on short historical data.

[1]https://dev.epicgames.com/community/learning/courses/kRm/unreal-engine-learning-age

Recent works in trajectory prediction have demonstrated the effectiveness of model-free approaches. Notable examples include Social-LSTM [7], which uses Long Short-Term Memory (LSTM) networks to predict human trajectories in crowded spaces by considering social interactions among pedestrians. GroupLSTM [8] focuses on predicting group trajectories in crowded scenarios, addressing the collective behaviour of groups within the crowd. Social-GAN [9] explores a similar idea by incorporating Generative Adversarial Networks (GANs) to produce socially acceptable trajectories that respect personal space and group movement patterns. More recently, methods like Trace and Pace [10] employ guided trajectory diffusion for controllable pedestrian animation, further pushing the boundaries of realistic trajectory prediction, at the cost of increased computing.

Although they achieve similar or better results when compared to model-based methods, model-free methods present some disadvantages. Perhaps the biggest one is their lack of generalization outside the scope of the simulation environment. In other words, being neural networks universal function approximators, it is unlikely that a neural network generalizes to out-of-distribution data. For these reasons, most of the works in trajectory prediction fail to generate *socially-acceptable* trajectories beyond the maximum number of steps considered by the dataset.

Moreover, model-free approaches often face challenges in real-time performance and encounter difficulties when incorporating time-varying conditioning signals. This can limit their applicability in dynamic and unpredictable environments where real-time adaptation is crucial.

By combining insights from both model-based and model-free approaches, it is possible to create more robust and adaptable crowd simulation models that leverage the strengths of each methodology.

We argue that reinforcement and imitation learning are well suited for this particular scenario since the rewards foreseen in the models can be explicitly modelled upon mathematically defined social rules. At the same time, this allows us to learn edge cases by directly learning through simulation.

C. Pedestrian Simulators and Frameworks

Simulators enable researchers to model and study the behaviour of pedestrians in various scenarios, providing insights into pedestrian movement, interaction, and safety.

One significant contribution to the field is the Continuum Crowds model developed by Treuille et al. [11]. This model simulates large crowds using a continuum dynamics approach, treating the crowd as a fluid. This method allows for the efficient simulation of large numbers of pedestrians by solving partial differential equations that govern the flow of the crowd. The Continuum Crowds model is particularly effective in scenarios where high-density crowds are present, such as evacuations or public gatherings, as it can capture complex flow patterns and interactions.

UniCrowd [12] is a human crowd simulator designed for modelling human-related dynamics. It is accompanied by a

dataset collected within its synthetic environment mimicking both the behavioural and visual aspects of crowds and includes a comprehensive validation pipeline.

PedPy [13] is another important tool in pedestrian dynamics research. PedPy is an open-source pedestrian trajectory analyzer that facilitates the analysis and visualization of pedestrian trajectories. It provides a comprehensive suite of tools for preprocessing, analyzing, and visualizing pedestrian movement data, allowing to identify patterns and evaluate the effectiveness and accuracy of different pedestrian models.

JuPedSim [14] is a well-established pedestrian simulation framework that focuses on the microscopic modelling of pedestrian dynamics. JuPedSim offers a variety of modules for simulating pedestrian movement, including models for route choice, walking behaviour, and interaction with the environment. One of the key features of JuPedSim is its ability to simulate pedestrian flow in complex environments, such as buildings with multiple exits or public transportation hubs. This flexibility makes JuPedSim a valuable tool for designing and evaluating infrastructure and safety measures in public spaces.

Recently, Mass[2] has been introduced as a set of plugins for Unreal Engine for creating AI agents that can roam inside an environment, simultaneously avoiding other pedestrians using either the Detour Crowd Manager or RVO [2].

In this paper, we base our comparison on different models implemented in JuPedSim, while using PedPy to extract meaningful metrics to allow fair comparison. All the training and inference-time rendering is performed in real-time inside Unreal Engine.

III. Method

A. Environment and Curriculum Training

We define a series of real-world-sized maps that exactly match the ones implemented inside JuPedSim, for fair comparisons. In addition, we also consider some additional and more complex scenes, as shown in Fig. 2. We randomly apply variations to the size, shape, position and rotation of both obstacles and goals in a curriculum-like fashion, from easier to more difficult scenarios. This is proven to be a good strategy in reinforcement learning [15] to learn better policies and adapt to variations in the environment configuration. Each agent is equipped with 18 uniformly distributed sensing rays, as depicted in Fig. 1, that serve as the main observations, together with its facing direction, and its actual and desired speed. No additional observation about the goal and obstacles' position, scale and orientation is given at any time to the agent, which most of the other methods require, as shown in Table I. Without the goal information, our agents need to find their way through it, promoting an explorative behaviour. The number of training agents can be changed for different types of maps. All the agents share the same weights and the learning rates for the Policy and Critic networks are 1e-3 and 1e-4 respectively.

[2]https://dev.epicgames.com/community/learning/tutorials/JXMl/unreal-engine-your-first-60-minutes-with-mass

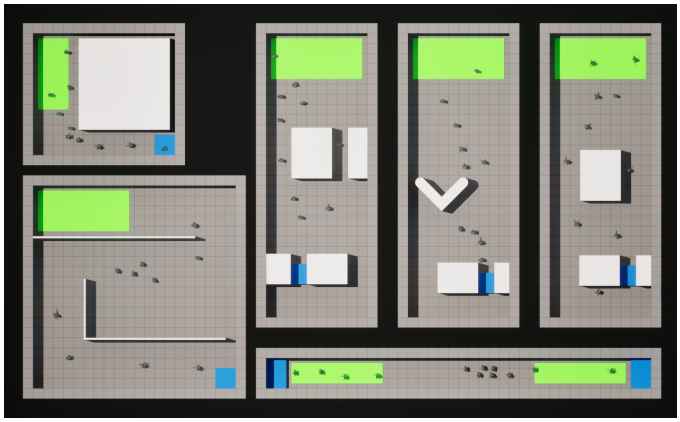


Fig. 2: A top-view example of a possible configuration of our training environment. Each of the five maps comprises: (i) a light green spawn area, (ii) a light blue goal area, (iii) multiple obstacles and walls with arbitrary size and location (in white). Agents are instructed to navigate each procedurally generated environment according to the rewards detailed in Sec. III-C.

B. Policy/Critic architectures and Actions

The objective of our proposed solution is to provide a simple yet effective tool for the generation of socially-acceptable human trajectories, with better realism compared to classic frameworks such as SFM and CFSM. Thus, the architecture is simple and shallow for both the policy and critic networks, as reported below. Though the tasks could appear to be similar, we do not compare our work against the multitude of frameworks for trajectory prediction: in fact, they target a different scope, validating a displacement error with respect to a given ground truth, which, in our case, does not exist.

Policy Network

Input → Concatenation with Memory (512 units)
→ Hidden Layer (256 units, ReLU) → Output

Critic Network

Input → Hidden Layer (256 units, ReLU) → Output

Actions

Output → Clamping (Tanh)
→ Actions (2 floats: right, forward vectors)

C. Rewards

The overall reward for each agent at time t is given by:

$$r^t = \lambda_g r_g^t + \lambda_v r_v^t + \lambda_c r_c^t + \lambda_d r_d^t \tag{1}$$

with $\lambda_g = 1$, $\lambda_v = 0.001$, $\lambda_c = \lambda_d = 0.1$, where

- $r_g^t = 1$ (**goal**) is a fixed reward for reaching the goal.
- r_v^t (**desired velocity**, Fig. 3) penalizes an agent with velocity $v \in [v_{min}, v_{max}]$ whenever it fails to match a certain desired velocity v_d. It is defined by a spline function $V(\cdot)$ taking values in $[V_{min}, V_{max}]$ defined by two polynomials ensuring parametric continuity:

$$r_v^t = \begin{cases} \alpha_l \left(v - v_d\right)^2 + V_{max} & \text{if } v_{min} \leq v \leq v_d \\ \alpha_r \left(v - v_d\right)^2 + V_{max} & \text{if } v_d < v \leq v_{max} \\ -1 & \text{otherwise} \end{cases} \quad (2)$$

with coefficients:

$$\alpha_l = V_{min} - \frac{V_{max}}{\left(v_{min} - v_d\right)^2},$$

$$\alpha_r = V_{min} - \frac{V_{max}}{\left(v_{max} - v_d\right)^2}$$

It forms a skewed quadratic function with vertex P_v and left/right control points P_l and P_r and clipping at V_{min}.

$$P_v = \begin{pmatrix} v_d \\ V_{min} \end{pmatrix} ; P_l = \begin{pmatrix} v_{min} \\ V_{max} \end{pmatrix} ; P_r = \begin{pmatrix} v_{max} \\ V_{max} \end{pmatrix} \quad (3)$$

- r_c^t (**ray collision**, Fig. 4) is a penalty for a ray colliding with an obstacle or a pedestrian, defined by the repulsion functions $R_o(\cdot)$ and $R_p(\cdot)$ respectively. If we define r_o and r_p as the shortest rays hitting every nearby obstacle or pedestrian, we have:

$$R_o(\cdot) = A_o * e^{-||r_o||/B_o}$$
$$R_p(\cdot) = A_p * e^{-||r_p||/B_p} \quad (4)$$

For example, if an agent has in its sensing range two walls and three pedestrians, it will receive a penalty given by the negative norm of the vector sum of five different vectors with length R and direction $-r$.

- r_d^t (**desired direction**) is a penalty that encourages the agents to explore the environment by penalizing the action of turning back towards their starting position. It is expressed as a gated dot product of the agent's forward vector $\vec{v}$ and the vector connecting the agent to its spawn position $\vec{v_s}$.

$$r_d^t = \begin{cases} -(\vec{v} \cdot \vec{v_s}) & \text{if } \vec{v} \cdot \vec{v_s} > 0.9 \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

IV. Results

We show multiple qualitative (Figures 5, 6, 8) and quantitative (Fig. 7) results on both real and synthetic scenes, following the style of JuPedSim and PedPy. What can be seen in the plots is that our method produces more human-like trajectories in every scenario, which look more human-like and not unrealistically straight, like the ones produced by the SFM and CFSM. This is particularly evident when plotting the time series of the angle between the forward vector and the x-axis, as originally proposed by [6] (Fig. 7).

In all the cases, it can be seen that our method produces a distribution of angle variations that resembles a normal Gaussian distribution, while the other methods present large sudden variations in the forward angle, leading to spiky distributions. Please refer to Table I and all the plots' captions for further interpretation of the results.

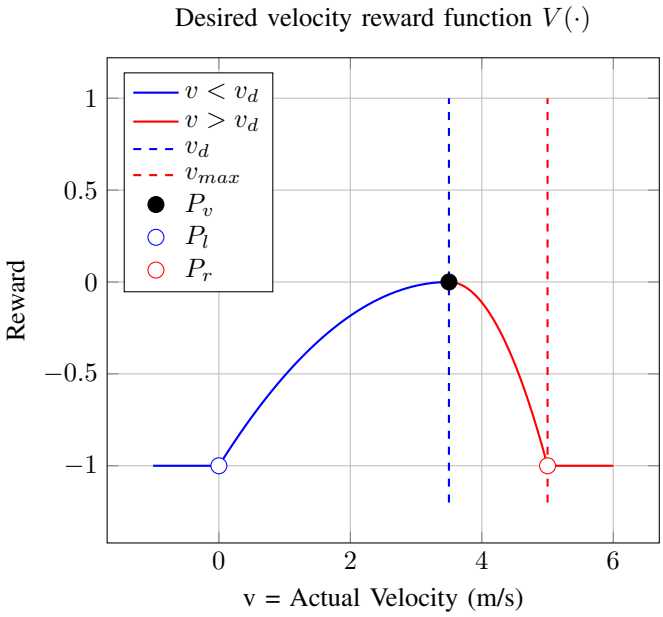


Fig. 3: Desired Velocity Function reward. At v_d each agent will receive the lowest penalty (0) for maintaining its desired speed. The left and right parts of the curve will be dynamically reconfigured during the simulation to fit the agents' desired minimum and maximum speed. v_d dynamically changes during the simulation based on the social forces given by pedestrians and obstacles in the agent's neighbourhood.

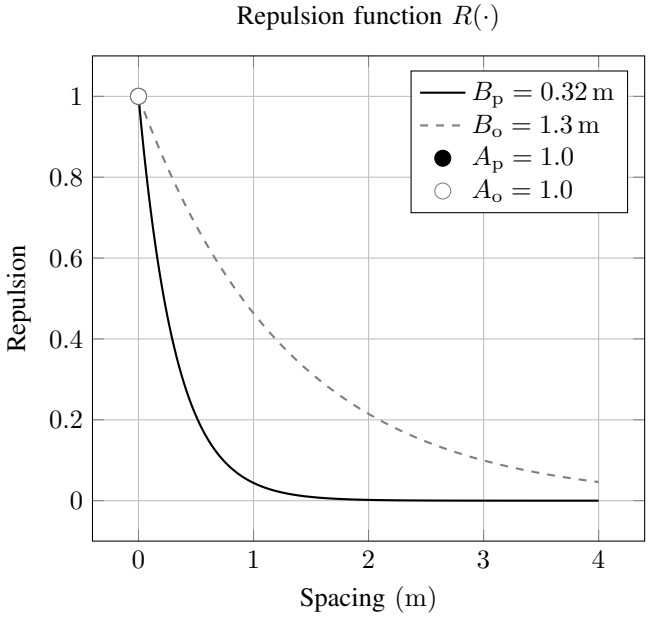


Fig. 4: Repulsion Function. While an agent gets closer to a pedestrian (full line) or an obstacle (dashed line), it will receive a bigger penalty modelled after the repulsion function $R(\cdot)$. The desired repulsion is a property of each individual agent and thus can be modified at runtime as well.

	Requires training	Real-time	Dynamic goals	Dynamic obstacles	Without map knowledge	Without goal knowledge	Run-time behaviour customizability	Collision avoidance
SFM [1]	✗	✓	✓	✓	✓	✗	Low	Explicit
R-SFM [13] *	✗	✗	✓	✓	✗	✗	Low	Explicit
CFSM [5] *	✗	✗	✓	✓	✗	✗	Low	Explicit
GCFSM [6] *	✗	✗	✓	✓	✗	✗	Low	Explicit
Continuum Crowds [11]	✗	✓	✓	✓	✗	✗	Medium	Implicit
Unicrowd [12]	✗	✓	✓	✗	✗	✗	Medium	Explicit
Mass AI (RVO) [2]	✗	✓	✗	✗	✗	✓	Medium	Explicit
TRACE [10]	✓	✗	✓	✓	✗	✗	High	Implicit
Ours	✓	✓	✓	✓	✓	✓	High	Implicit

TABLE I: Comparison with deterministic and learning-based methods for crowd dynamics. Methods with (*) use pathfinding (including JuPedSim's SFM implementation, where R stands for routing), so they cannot be considered real-time. Our method is the only one that does not require any map or goal knowledge.

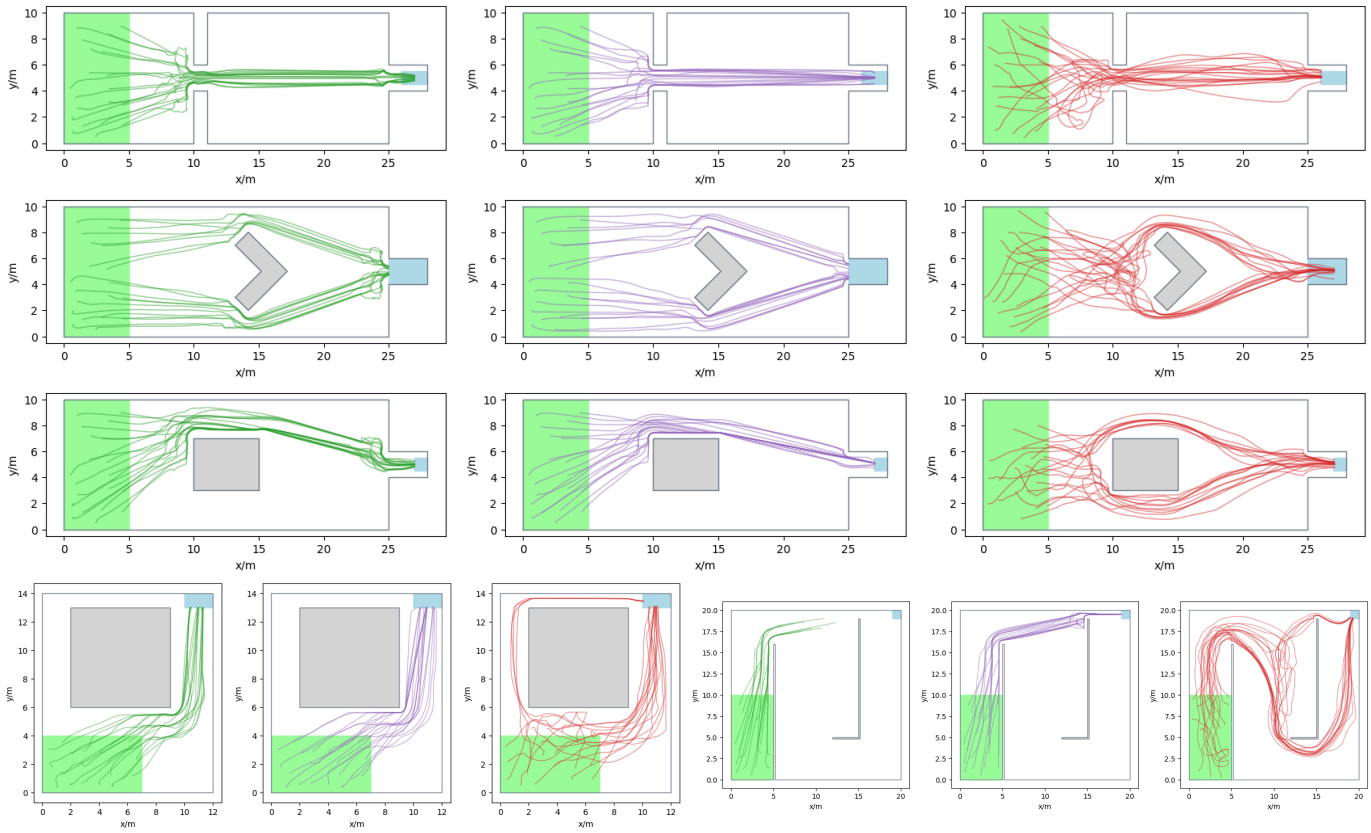


Fig. 5: **Green**: R-SFM, **Purple**: CFSM, **Red**: Ours. **Bottlenecks and obstacles avoidance**. The green rectangle is the spawn area. Agents in the classical models (SFM and CFSM) walk in unrealistic, straight lines and tend to prefer a unique direction even in cases where an identically expensive free pathway is available. Our model can produce more organic and distinct pathways, allowing agents to sometimes "get lost" and explore the map for alternative paths.

V. Conclusions

In this work, we presented Dynamic Crowd Routing (DCR), a reinforcement-learning solution that allows to model complex pedestrian dynamics which are more human-like than classical deterministic methods (like SFM and CFSM). Each agent in our simulation is highly customizable in terms of actions, goals, desired velocity and social behaviours, while goals and obstacles are meant to be dynamic. No map or goal knowledge is needed to run the simulation, allowing for its usage in unbounded, dynamic urban digital twins.

VI. Acknowledgements

Project funded under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.4 - Call for tender No. 1031 of 17/06/2022 of Italian Ministry for University and Research funded by the European Union – NextGenerationEU (proj. nr. CN_00000013)

References

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

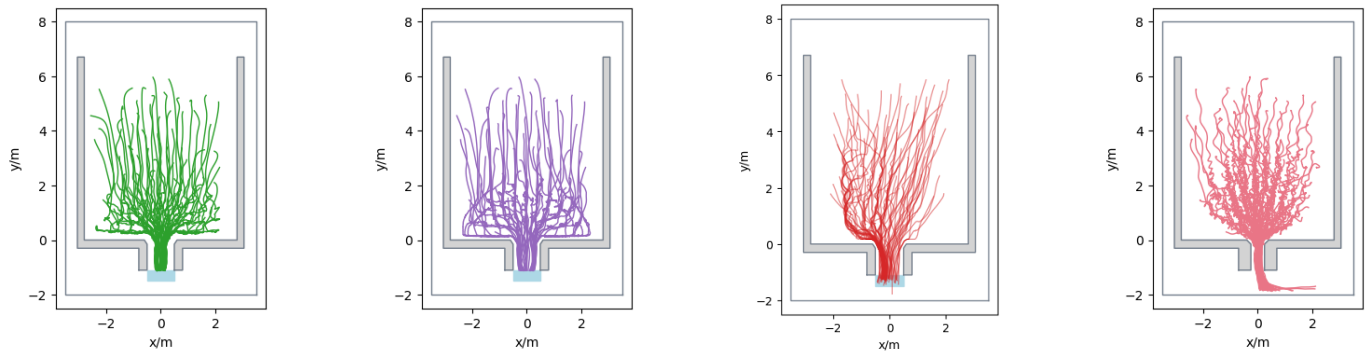


Fig. 6: **Green**: R-SFM, **Purple**: CFSM, **Red**: Ours, **Pink**: Ground truth. Comparison with real ground truth trajectories in a bottleneck scenario (agents move top-to-bottom). The agents in our simulation behave more like real humans, not walking towards walls to suddenly change direction, but instead slowing down and organizing in a more realistic queue formation.

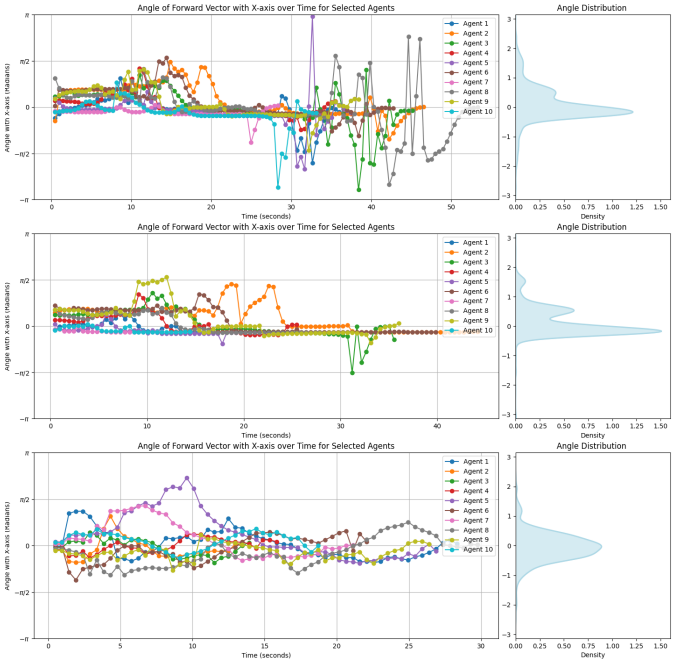


Fig. 7: Top to bottom: R-SFM, CFSM, Ours. Changing of the forward vector direction over time and corresponding distributions (right), highlighting the abrupt variations in the pedestrians' direction in rule-based models, which are notably mitigated using our solution. The plots correspond to the trajectories in the last line of Fig. 5.

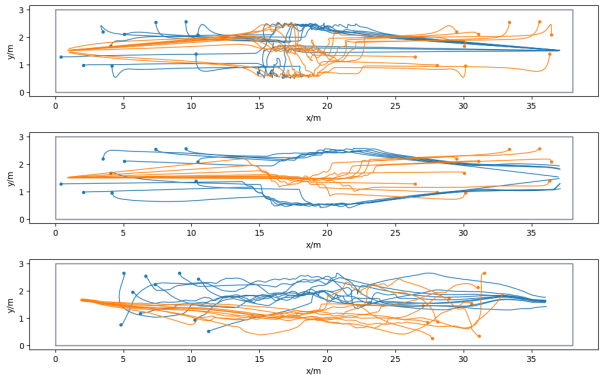


Fig. 8: Top-to-bottom: R-SFM, CFSM, Ours. Emergent lane formation in a narrow 3x40 corridor.

[2] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE international conference on robotics and automation*. Ieee, 2008, pp. 1928–1935.

[3] N. Conci, N. Bisagno, and A. Cavallaro, "On modeling and analyzing crowds from videos," in *Computer Vision for Assistive Healthcare*. Elsevier, 2018, pp. 319–336.

[4] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, pp. 25–34.

[5] A. Tordeux, M. Chraibi, and A. Seyfried, "Collision-free speed model for pedestrian dynamics," in *Traffic and Granular Flow'15*. Springer, 2016, pp. 225–232.

[6] Q. Xu, M. Chraibi, A. Tordeux, and J. Zhang, "Generalized collision-free velocity model for pedestrian dynamics," *Physica A: Statistical Mechanics and its Applications*, vol. 535, p. 122521, 2019.

[7] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[8] N. Bisagno, B. Zhang, and N. Conci, "Group lstm: Group trajectory prediction in crowded scenarios," in *Proceedings of the European conference on computer vision (ECCV) workshops*, 2018, pp. 0–0.

[9] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2255–2264.

[10] D. Rempe, Z. Luo, X. Bin Peng, Y. Yuan, K. Kitani, K. Kreis, S. Fidler, and O. Litany, "Trace and pace: Controllable pedestrian animation via guided trajectory diffusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 756–13 766.

[11] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," *ACM Transactions On Graphics (TOG)*, vol. 25, no. 3, pp. 1160–1168, 2006.

[12] N. Bisagno, A. L. Stefani, N. Garau, F. De Natale, and N. Conci, "Unicrowd simulator: Visual and behavioral fidelity for the generation of crowd datasets," in *2024 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2024.

[13] T. Schrödter and T. P. D. Team, "Pedpy - pedestrian trajectory analyzer (v1.1.0)," 2024. [Online]. Available: https://doi.org/10.5281/zenodo.10814490

[14] M. Chraibi, K. Kratz, T. Schrödter, and T. J. D. Team, "Jupedsim (v1.2.1)," 2024. [Online]. Available: https://doi.org/10.5281/zenodo.11093061

[15] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *Journal of Machine Learning Research*, vol. 21, no. 181, pp. 1–50, 2020.