2024 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR) | 979-8-3503-7202-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/AIxVR59861.2024.00035

Toward Realistic Human Crowd Simulations with Data-Driven Parameter Space Exploration

Kaidong Hu
Rutgers University
kaidong.hu@rutgers.edu

Sejong Yoon
The College of New Jersey
yoons@tcnj.edu

Vladimir Pavlovic
Rutgers University
vladimir@rutgers.edu

Mubbasir Kapadia
Rutgers University
mk1353@rutgers.edu

***Abstract*—Understanding human crowd motion is crucial for realistic crowd simulations and content creation. Over the multiple decades, several computational algorithms have been developed to devise a more realistic simulation that will vastly improve the user experiences. However, the gap between real and simulated human motion and behavior is still large. One of the promising thrusts of the crowd simulation community to reduce the gap is to utilize a data-driven trajectory prediction model, using the access to a large amount of data. However, building a crowd simulation model based on the learned microscopic trajectory model is still a challenging task. In addition, unlike individual or group-level human trajectory data, large-scale real human crowd motion data is not readily available. To overcome these, we investigate the utility of synthetic simulation data. We propose a novel human crowd motion estimation framework that can predict simulator parameters from trajectory data. Our initial findings show promising results that our method can robustly estimate simulation parameters.**

***Index Terms*—Parameter space exploration, Crowd simulation, Data-driven model**

I. Introduction

Crowd behavior modeling is applied in real-world scenarios, such as content creations for video games and animations, egress analysis, autonomous driving, and crowd control at events [1]–[3]. Since understanding real human crowd motion is essential for building realistic crowd simulations, researchers have investigated the problem in various directions. A lot of work has been done to predict human trajectory from real-world videos, since the ability to accurately predict future locations shows that computational models can effectively capture human motion, even in crowded places.

Most human trajectory prediction models try to learn patterns from microscopic real human trajectories. There are two main lines of work, namely knowledge-based and data-driven approaches [4]. Traditional approaches see an agent as a particle moving in space, described using human motion dynamics. This includes the popular force-based models like Social Force (SF) [5], velocity-based models like reciprocal collision avoidance (RVO) [6] and optimal reciprocal collision avoidance (ORCA) [7], and rule-based models [8]. There have been many other notable models and the interested readers may refer to the review [4] that also provides a good summary of trends in the past decade. These simulation models are useful foundations to understand large-scale dynamics, but obtainable insights are indirect.

The other line of work is the data-driven approach to model and predict human trajectories. Early methods used recurrent neural network-based sequence models, like Social LSTM [9]. Others focus on prediction aspect of the problem with a generative model. The idea is to sample potential future positions from a learned model and hope the sample is close to the ground truth [10], [11]. Recent advances like transformers have also been applied, like AgentFormer [12] and TUTR [13]. However, these data-driven trajectory prediction models focus on trajectory prediction itself, and they are not for human crowd motion generation and simulation.

At the intersection of the trajectory prediction models and the human crowd simulation, researchers investigated ways to learn a steering model from the identified trajectory dynamic pattern, e.g., by utilizing hand-crafted policies [7], [14]–[17]. However, they are not invertible, i.e., the parameter configuration cannot be readily computed from an agent's observed steering. Some [18] have tackled the parameter configuration search as an optimization problem, but no known method is shown to be effective for more realistic heterogeneous crowds. Others tried to learn policies in a non-parameterized way [19]–[23] but the problem remains challenging. In addition, limited access to real human crowd trajectory is another critical roadblock in pursuing this research.

In this paper, we propose a new framework that aims to overcome aforementioned challenges by making two contributions: (a) our framework can find configuration parameters for a crowd simulation model using trajectory patterns learned from a data-driven trajectory prediction model, (b) we suggest a novel algorithm to find long term agent-to-agent interaction that trajectory prediction models often miss. The proposed framework consists of two autoencoders and a crowd simulator. First, we learn a trajectory prediction network with an autoencoder called Trajectory Prediction Network (TPN). Then, we learn another autoencoder, called Configuration Prediction Network (CPN), that regresses the latent trajectory representation of the first network to the simulator's configuration parameters. In the experiments, we show that the proposed framework can effectively estimates an environment parameter (corridor width) from a given trajectory pattern, that can be used for SF and ORCA simulators. We also report initial findings that our method can potentially predict other agent parameters and robust to initialization.

2771-7453/24/$31.00 ©2024 IEEE
DOI 10.1109/AIxVR59861.2024.00035

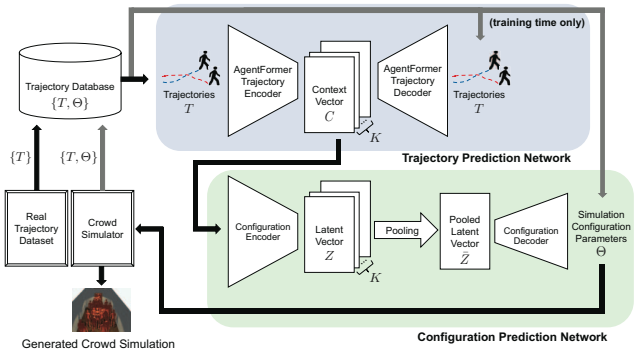

Fig. 1: Proposed learning framework. K indicates the number of trajectory segments used in the AgentFormer model. At test time (dark black), simulator configuration parameters may be estimated from an unseen, real trajectories, and the estimates can be used in the simulator to generate new trajectories.

II. Proposed Framework

Our proposed framework is depicted in Figure 1. At *training* phase, we first prepare simulated trajectories dataset with known simulator configuration parameters. Any crowd simulator can be used and our goal is to find parameters for the steering model used in the simulator. Next, we fed the trajectories into the TPN so that the network can learn the trajectory pattern from the input data. The TPN should have an autoencoder-like structure, since our goal is to find the underlying trajectory pattern representation, rather than future trajectory prediction. In this paper, we adopted AgentFormer [12], but any model that is capable of reconstructing the input trajectory can be used. Next step is to extract the latent pattern representation of the TPN and fed it to the CPN as input. The CPN then regresses the latent representation to the configuration parameter using another autoencoder. This second autoencoder aims to fill the gap between TPN model's latent space and simulator's configuration parameter space. At *testing* phase, we can fed the new trajectory set into the framework and the framework will predict the simulator's configuration parameter that likely enables the simulator generate new trajectories similar to the TPN input.

A. Agent groups for neighbor interaction

For human pedestrian trajectory prediction models, it is common to include a method to capture interactions between people. AgentFormer also does this by grouping neighboring agents in trajectories. However, this approach is problematic for crowd simulations. Suppose a crowd evacuation scenario, where agents tries to exit from a room with a narrow corridor exist in front of the exit. Naturally, agents enter and exit the corridor at different timing. Due to congestion, one agent typically interacts with a small group of others before it evacuates from the room. In addition, over the long duration of crowd simulation, the agent may not interact with only nearby agents and may interact with some other agents who were far away at the beginning but later get close together, vice versa. In other words, the agent-to-agent interaction learned by AgentFormer TPN may not be effective in capturing long term interactions. A naïve approach to overcome this limitation is to increase the agent group size, but increasing the agent size will easily get stuck by the limit of the hardware (GPU) memory for applications like crowd simulations.

To resolve this issue, we propose an agent grouping algorithm that aims to cluster all agents within a simulation into small groups. Note that the meaning of the term "agent group" in this paper differs from the social group used in the transportation literature [24]. We used the term "group" to describe the algorithm only. Each agent group should have the same number of agents, N. We utilize Average Displacement Error (ADE), which is the Mean Square Error (MSE) between the predicted trajectory locations and the ground truth trajectory locations, as the clustering criterion. The overall agent grouping algorithm is summarized in algorithm 1.

Algorithm 1: Agent grouping strategy

input : Number of agents in each group: N
All trajectories: $\{\mathcal{T}_1, \cdots, \mathcal{T}_M\}$ where $\mathcal{T}_i$ is the trajectory of agent $i \in [1..M]$ where M denotes the total number of agents
output: A set of agent groups (clusters): $\mathcal{L}$
begin
  $\mathcal{L} \leftarrow \{\}$; $\mathcal{P} \leftarrow \{1, \cdots, M\}$
  while $\mathcal{P} \neq \varnothing$ **do**
    $i \leftarrow$ Sample from $\mathcal{P}$
    $\ell \leftarrow \{i\}$; $n \leftarrow N$; $\mathcal{Q} \leftarrow \{1, \cdots, M\} \setminus i$
    while $n > 0$ **do**
      $\ell \leftarrow \ell \cup \{j\}$
      where $j \in \mathcal{Q}$ s.t. $\arg\min_j \text{ADE}(\mathcal{T}_j, \mathcal{T}_i)$
      $\mathcal{Q} \leftarrow \mathcal{Q} \setminus j$; $n \leftarrow n - 1$
    end
    $\mathcal{P} \leftarrow \mathcal{P} \setminus \ell$; $\mathcal{L} \leftarrow \mathcal{L} \cup \{\ell\}$
  end
end

B. Extracting latent variables from TPN and training CPN

In AgentFormer, trajectories are split into several small segments before the prediction is made. We also apply this strategy in our trajectory representation. Each trajectory segment is 27 frames long. AgentFormer considers first 13 frames as a past window, i.e., history, and the rest 14 frames as a future window. Since not all agents exist in all frames and trajectories may not be aligned, the beginning of a trajectory segment is determined by the first frame of the first agent in the record. An agent in the same group will only be included in a certain temporal segment if it has at least one frame in past window and in future window in that segment. Thus, each temporal segment may cover different number of agents.

After the trajectory pattern representation is learned by AgentFormer, we can retrieve per-agent, per-trajectory segment. To represent the whole trajectory set from an agent group, we need to combine all of these into one latent representation, a fixed-length vector. To do this, we first take

the AgentFormer's context vertor C and pad it to have the size of maximum number of agents. Then, we apply a linear transformation to compress the vectors. We find a normal distribution that is represented by the context vectors with $\mathcal{N}(\text{mean}(C), \text{var}(C))$. We sample once from the distribution that has the desired length of $256 \times K$ where K is the number of segments. Next, an average pooling is applied over all trajectory segments for the agent group in consideration, and the result is the latent trajectory pattern-to-configuration mapping representation for that specific agent group. There, we impose a normal distribution constraints to this latent vector during training time to form a desired shape in the latent space following [25]. For training loss, we used root mean squared error (RMSE) of the CPN output against the ground truth parameter Θ.

III. Experiments

To evaluate the potential of the proposed method, we designed a evacuation scenario using crowd simulators. First, we introduce the environment and steering model parameters. Next, we explain the dataset and performance measures. Then, we present the experiment results.

A. Simulation Configuration Parameters

We consider an environment, considering a evacuation scenario, built in a 20 meters $\times$ 7 meters region as shown in Figure 2. Agents are initially located in the top half of the space lined up in 5 rows and 15 columns. The bottom half of the space contains a gradually narrowing bottleneck to a corridor. The bottleneck opens the same width as the hallway near the end. The width of corridor is configurable between 1 meter (narrowest) and 7 meters (fully open). A one meter exit door is placed directly after the hallway. An agent is considered successfully evacuated when it moves below the exit and will be removed from the environment. Each agent's trajectory is formatted as a time series data of its global position (x, y). The recording of the trajectory starts immediately after the agent enters the hallway and stops when agent evacuated. The configuration of agents' properties are defined differently upon each steering model. In the SF model, there are 14 real-valued parameters and for ORCA, there are 5 real-valued parameters. Table I and Table II summarize what constitute these parameters and the range.

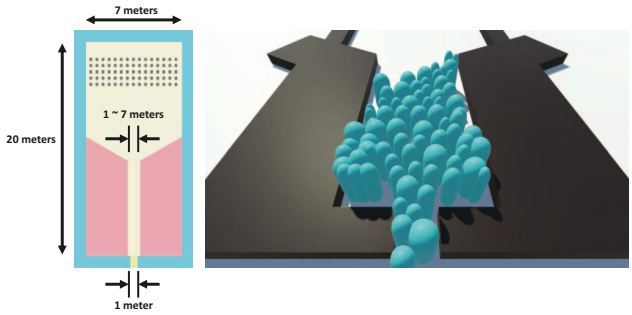

Fig. 2: Evacuation scenario blueprint. An example simulation using ORCA with heterogeneous agents.

TABLE I: Simulation configurations parameters (SF)

ID	Parameter name	Value Range
1	Environment corridor width	[1.0, 7.0]
2	Agent radius	[0.15, 0.3]
3	Acceleration	[0.25, 0.75]
4	Personal space threshold	[0.15, 0.45]
5	Repulsion importance	[0.15, 0.45]
6	Query radius	[1.5, 4.5]
7	Body force	[750, 2250]
8	Agent body force	[750, 2250]
9	Sliding friction force	[1500, 3000]
10	Agent inverse proximity force importance	[0.04, 0.12]
11	Agent proximity force importance	[12.5, 37.5]
12	Wall inverse proximity force importance	[0.04, 0.12]
13	Wall proximity force importance	[12.5, 37.5]
14	Max speed limit	[1.3, 3.9]

TABLE II: Simulation configurations parameters (ORCA)

ID	Parameter name	Value Range
1	Environment corridor width	[1.0, 7.0]
2	Agent radius	[0.15, 0.3]
3	Neighbor distance threshold	[5.0, 15.0]
4	Time horizon for agents	[1.0, 3.0]
5	Max speed	[0.7, 0.8]
6	Time horizon for obstacles	[1.0, 2.0]

B. Dataset and Performance Measures

For each parameter setting, simulations are performed using all variable parameters. We used SteerSuite [26] for simulations. When generating random parameters, we sample one parameter at a time while fixing all the others. Each parameter is sampled uniformly from the value range summarized in Table I and Table II. The other parameters took the default value which is the center of the value range. For each parameter, we generated 550 random values and each value was simulated. 400 of them were included in the training split. The remainder is reserved for the test split. It should be noted that a small number of trials were discarded due to failed simulation. For the hyperparameters needed for architectural design, we used cross validation. For most experiments, we used SF [5] and ORCA [7] as base steering model. We considered both homogeneous and heterogeneous crowds, where the agents in the former shares the same parameter configuration while those in the latter do not as shown in Figure 2.

All the environment and agent steering model parameters are arranged to form the ground truth configuration parameter: $\Theta = [\theta_1, \cdots, \theta_u]$, where θ_u denotes the parameter with ID u as listed in Table I and Table II. We used mean absolute error (MAE) to measure the performance of parameter prediction: $\text{MAE}\,(y, y^\star) = \text{mean}\,(|\, y - y^\star \,|)$. Configurations are also scaled to the range $[-1, 1]$. Since the configurations are sampled uniformly from defined range, a random predictor will produce an MAE value 0.66, and a trivial zero-predictor which always predict zeros will have an MAE value of 0.50.

C. Determining agent group size

Our algorithm 1 requires a hyperparameter N. To determine this, we ran preliminary experiments using TPN. Figure 3

shows the trend of trajectory prediction error as a function of group size. Intuitively, for homogeneous agents, larger group size means more agents share the same parameters, hence the prediction becomes easier. For heterogeneous agents, larger group means each group has more sophisticated mixture of heterogeneous behaviors, thus prediction becomes harder. Hence, we chose 10 as a balance between the two.

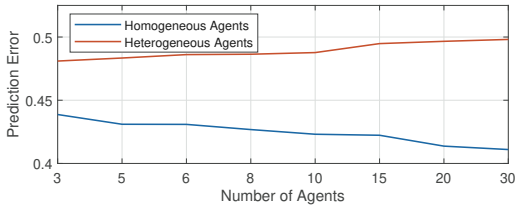

Fig. 3: Prediction error with varying agent group size

D. Results

Our experiments focused on verifying two aspects of the proposed framework: feasibility of the idea and robustness of the trained model. For the first part, we investigated which simulation configuration parameters are reasonably predictable. For the second part, we investigated the prediction performance change with various randomization settings. Namely, we tested three types of randomizations: (a) network weight initialization (TPN, CPN), (b) dataset, and (c) initial locations of the agents. For each randomization setting, we ran three trials and report the mean and the variance of the parameter prediction MAE. For the dataset, we generated the entire dataset from scratch for each trial.

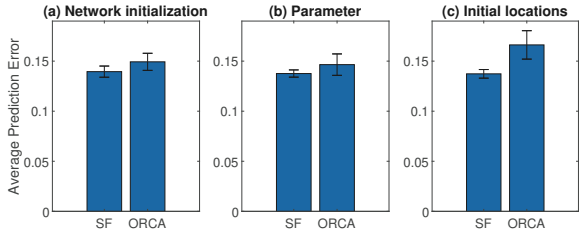

Fig. 4: Environment configuration parameter (ID 1) prediction errors with various randomization settings. It is clear that the proposed framework can predict the corridor width.

Figure 4 shows that our framework can robustly estimate the corridor width (varying from 1 to 7 meters) regardless of randomization type and the steering model choice. For agent steering model parameters, we found a clear difference in prediction performance comparing homogeneous and heterogeneous setups. Not surprisingly, homogeneous setup is much easier to predict than heterogeneous setup, which is virtually impossible to do much better than random (MAE 0.5). However, it was interesting to see that some parameters (Agent inverse proximity force importance in SF and Max speed in ORCA) are relatively easy to predict in the heterogeneous setting.

We also compared the trajectories generated by a simulator using the steering model configuration parameters predicted by our method to the trajectories generated by the simulator using the original ground truth configuration parameters. We computed ADE between the trajectories to evaluate the performance. The results show that trajectories generated in the homogeneous setup are closer to the trajectories of simulations using ground truth parameters than those generated in the heterogeneous setup. This result coincides with the observation of the parameter prediction experiments discussed above.

What is different in the proposed framework vs. learning steering model parameters directly from trajectories? At high level, it may look like there is no difference in the sense that we learn model parameters from trajectories. However, unlike existing approaches, our framework enables utilizing knowledge learned from various TPN choices. By separating the trajectory pattern knowledge and the mapping from the pattern to the configuration parameter space, we can further expand the framework in various ways to learn from multiple trajectory distributions. For example, one can combine two TPN latent representations, one for simulated trajectories and the other for the real trajectories. This way we can alleviate the issue of imbalanced data nature to the data-driven crowd models. To see the potential of this idea, we conducted an ablation study to compare the proposed framework to directly regressing TPN latent variable to configuration parameter, hence removing CPN. As shown in Figure 5, it is clear that CPN improves the prediction performance over the TPN-only model, suggesting the importance of pursuit for latent mapping between TPN-learned trajectory pattern knowledge and the simulator configuration parameter space.

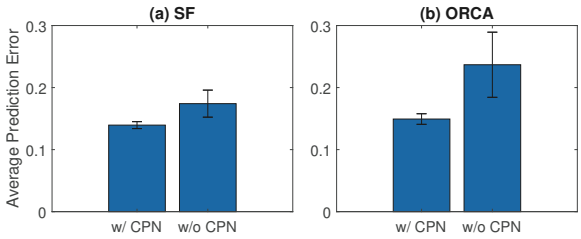

Fig. 5: Ablation study for environment configuration parameter (ID 1, corridor width) with random weight initialization.

IV. Conclusion

In this paper, we proposed an autoencoder framework that can predict simulator configuration parameter. We also proposed an agent grouping algorithm that can capture long term agent-to-agent interaction and reduce computation. Experiments suggest a promising potential of the proposed method with rooms of improvement in challenging situations like heterogeneous crowd simulations. In the future, we aim to expand the framework to learn the CPN that can map multiple trajectory distributions into a configuration parameter space. It would be interesting to see how the proposed model applies to other steering algorithms such as centrifugal force-based [27], [28] or reinforcement-based [29] models.

ACKNOWLEDGEMENT

This work was supported in part by the NSF grant awards IIS #1955365 and #1955404. The authors acknowledge use of the ELSA high performance computing cluster at The College of New Jersey for conducting the research reported in this paper. This cluster is funded in part by the NSF under grant numbers OAC #1826915 and #2320244.

REFERENCES

[1] D. Thalmann and S. R. Musse, *Crowd Simulation*. Springer-Verlag London, 2013.

[2] M. Kapadia, N. Pelechano, and J. Allbeck, *Virtual Crowds: Steps Toward Behavioral Realism*. Morgan & Claypool Publishers, 2015.

[3] N. Pelechano, J. M. Allbeck, M. Kapadia, and N. I. Badler, *Simulating Heterogeneous Crowds with Interactive Behaviors*. A K Peters/CRC Press, 2016.

[4] R. Korbmacher and A. Tordeux, "Review of pedestrian trajectory prediction methods: Comparing deep learning and knowledge-based approaches," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 24 126–24 144, 2022.

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

[6] 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*, 2008, pp. 1928–1935.

[7] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[8] A. Kirchner and A. Schadschneider, "Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics," *Physica A: Statistical Mechanics and its Applications*, vol. 312, no. 1, pp. 260–276, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437102008579

[9] 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.

[10] 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.

[11] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "Sophie: An attentive gan for predicting paths compliant to social and physical constraints," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1349–1358.

[12] Y. Yuan, X. Weng, Y. Ou, and K. Kitani, "Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Los Alamitos, CA, USA: IEEE Computer Society, oct 2021, pp. 9793–9803. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICCV48922.2021.00967

[13] L. Shi, L. Wang, S. Zhou, and G. Hua, "Trajectory unified transformer for pedestrian trajectory prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 9675–9684.

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

[15] S. Curtis, B. Zafar, A. Gutub, and D. Manocha, "Right of way," *The Visual Computer*, vol. 29, no. 12, pp. 1277–1292, 2013.

[16] P. Knob, V. F. de Andrade Araujo, R. M. Favaretto, and S. R. Musse, "Visualization of interactions in crowd simulation and video sequences," in *2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 2018, pp. 250–25 009.

[17] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, "Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 424–14 432.

[18] 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.

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

[20] J. Lee, J. Won, and J. Lee, "Crowd simulation by deep reinforcement learning," in *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*, 2018, pp. 1–7.

[21] N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. Ortega, D. Strouse, J. Z. Leibo, and N. De Freitas, "Social influence as intrinsic motivation for multi-agent deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3040–3049.

[22] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in neural information processing systems*, 2016, pp. 4565–4573.

[23] P. Long, W. Liu, and J. Pan, "Deep-learned collision avoidance policy for distributed multiagent navigation," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 656–663, 2017.

[24] C. Feliciani, X. Jia, H. Murakami, K. Ohtsuka, G. Vizzari, and K. Nishinari, "Social groups in pedestrian crowds as physical and cognitive entities: Extent of modeling and motion prediction," *Transportation Research Part A: Policy and Practice*, vol. 176, p. 103820, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965856423002409

[25] K. Kato, J. Zhou, T. Sasaki, and A. Nakagawa, "Rate-distortion optimization guided autoencoder for isometric embedding in Euclidean latent space," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 5166–5176.

[26] S. Singh, M. Kapadia, P. Faloutsos, and G. Reinman, "An open framework for developing, evaluating, and sharing steering algorithms," in *Proceedings of the 2nd International Workshop on Motion in Games*, ser. MIG '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 158–169. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-10347-6_15

[27] M. Chraibi, A. Seyfried, and A. Schadschneider, "Generalized centrifugal-force model for pedestrian dynamics," *Phys. Rev. E*, vol. 82, p. 046111, Oct 2010. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.82.046111

[28] M. Moussaïd, D. Helbing, and G. Theraulaz, "How simple rules determine pedestrian behavior and crowd disasters," *Proceedings of the National Academy of Sciences*, vol. 108, no. 17, pp. 6884–6888, 2011. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1016507108

[29] F. Martinez-Gil, M. Lozano, and F. Fernández, "Emergent behaviors and scalability for multi-agent reinforcement learning-based pedestrian models," *Simulation Modelling Practice and Theory*, vol. 74, pp. 117–133, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X17300503