CEDRL: Simulating Diverse Crowds with Example-Driven Deep Reinforcement Learning

ANDREAS PANAYIOTOU, CYENS - Centre of Excellence, Cyprus
ANDREAS ARISTIDOU, University of Cyprus & CYENS - Centre of Excellence, Cyprus
PANAYIOTIS CHARALAMBOUS, CYENS - Centre of Excellence, Cyprus

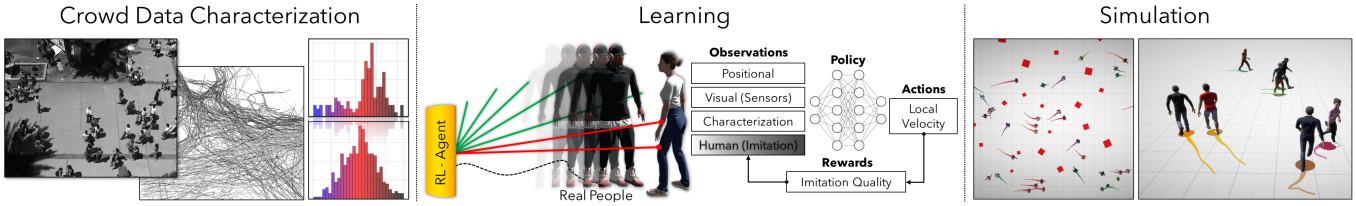

Fig. 1. We use a complexity metric to characterize real-world crowd data and train virtual agents to exhibit controllable diverse crowd behaviors from data-driven guidance to independent decision-making. Our model populates new environments with plausible simulations.

The level of realism in virtual crowds is strongly affected by the presence of diverse crowd behaviors. In real life, we can observe various scenarios, ranging from pedestrians moving on a shopping street, people talking in static groups, or wandering around in a public park. Most of the existing systems optimize for specific behaviors such as goal-seeking and collision avoidance, neglecting to consider other complex behaviors that are usually challenging to capture or define. Departing from the conventional use of Supervised Learning, which requires vast amounts of labeled data and often lacks controllability, we introduce **C**rowds using **E**xample-driven **D**eep **R**einforcement **L**earning (CEDRL), a framework that simultaneously leverages multiple crowd datasets to model a broad spectrum of human behaviors. This approach enables agents to adaptively learn and exhibit diverse behaviors, enhancing their ability to generalize decisions across unseen states. The model can be applied to populate novel virtual environments while providing **real-time controllability** over the agents' behaviors. We achieve this through the design of a reward function aligned with real-world observations and by employing curriculum learning that gradually diminishes the agents' observation space. A **complexity characterization metric** defines each agent's high-level crowd behavior, linking it to the agent's state and serving as an input to the policy network. Additionally, a parametric reward function, influenced by the type of crowd task, facilitates the learning of a diverse and abstract **behavior "skill" set**. We evaluate our model on both training and unseen real-world data, comparing against other simulators, showing its ability to generalize across scenarios and accurately reflect the observed complexity of behaviors. We also examine our system's controllability by adjusting the complexity weight, discovering that higher values lead to more complex behaviors such as wandering, static interactions, and group dynamics like joining or leaving. Finally, we demonstrate our model's capabilities in novel synthetic scenarios.

CCS Concepts: • **Computing methodologies** → **Sequential decision making**; **Real-time simulation**; **Reinforcement learning**; **Learning from demonstrations**; **Animation**.

Additional Key Words and Phrases: crowd simulation, data-driven methods, user control, crowd authoring, reinforcement learning, imitation learning

Authors' Contact Information: Andreas Panayiotou, CYENS - Centre of Excellence, Nicosia, Cyprus, a.panayiotou@cyens.org.cy; Andreas Aristidou, University of Cyprus & CYENS - Centre of Excellence, Nicosia, Cyprus, a.m.aristidou@gmail.com; Panayiotis Charalambous, CYENS - Centre of Excellence, Nicosia, Cyprus, p.charalambous@cyens.org.cy.

1 INTRODUCTION

Simulating real human crowd behavior is a challenging task due to the diverse and intricate nature of human interactions. Human navigation is not just about moving from one point to another; it includes a wide spectrum of additional and complex actions. For instance, in urban settings, individuals often navigate in small groups towards specific destinations or pause outside storefronts. In contrast, in dynamic environments like outdoor markets or university campuses, people are more likely to wander and engage in conversations within static groups. Furthermore, real-world observations reveal that each individual exhibits unique behaviors, contributing to a rich variety of actions. Accurately simulating this array of distinct behaviors proves to be a challenging task.

Although there exist various techniques for simulating crowds, many of them require intricate user or designer adjustments to achieve the desired visual outcome. Addressing this challenge involves exploring methods, such as implicitly learning behaviors [Charalambous et al. 2023] or parameters [Pettré et al. 2009; Wolinski et al. 2014] from reference data, representing the desired crowd behaviors. Traditionally, Supervised Learning (SL) has been the dominant approach to modeling agent behavior, often incorporating real-world data to improve navigation realism [Charalambous and Chrysanthou 2014; Lerner et al. 2007]. While these methods have advanced the field, they are often hard to implement, face limitations related to the quantity and quality of input data, struggle with generalizability, and lack controllability, which is an important feature in crowd simulators [Lemonari et al. 2022]. Recent works have introduced Deep Reinforcement Learning (RL) as a dynamic and effective alternative [Hu et al. 2023; Long et al. 2017; Panayiotou et al. 2022]. RL adopts a trial-and-error learning methodology, optimizing agent behaviors through scalar reward signals generated during the simulation. These rewards are based on the actions executed by agents after receiving a list of observations, which partially describe the current state of the environment. However, the challenge lies in defining and balancing reward signals to accurately capture the diverse spectrum of human crowd behaviors,

which is extremely difficult due to the complexity and variability. For example, Panayiotou et al. [Panayiotou et al. 2022] handles this by concurrently learning a distribution of reward functions, while Charalambous et al. [Charalambous et al. 2023] defines a reward function using novelty detection with respect to some reference crowd data.

In this work, we introduce CEDRL, an innovative methodology for learning human crowd behaviors. Our approach distinguishes itself from most existing RL-based methods by combining imitation learning features, deep RL, and real-world data, enabling virtual agents to gain comprehensive crowd behavior "knowledge". Unlike traditional imitation learning techniques such as Behavioral Cloning [Bi et al. 2020], where agents directly mimic expert demonstrations and struggle with adapting to unfamiliar states, our model develops a more abstract set of behavioral decisions that generalize better across unseen states. To achieve this, we design a reward function to align the agents' behaviors with real-world observations and incorporate a curriculum learning mechanism. Initially, the agents learn to imitate diverse crowd behaviors with partial data guidance. Gradually, we exclude information about movement states, presented in the reference data, from the agents' observation space, and use a parametric reward function to evaluate behavior over complexity-characterized real-world samples. This transition allows agents to switch to pure RL through individual decision-making. The acquired knowledge can then be applied and controlled in real-time to populate novel environments, enabling agents to exhibit a diverse spectrum of crowd behaviors without explicitly defining task-specific reward terms. For instance, manually fine-tuning reward parameters and balancing conflicting behaviors, such as wandering or engaging in stationary conversations, becomes increasingly time consuming and complex as the range of behaviors increases. Although recent methods like Guided-RL [Charalambous et al. 2023] advance beyond the manual definition of reward functions by inferring them from expert behavior, CEDRL employs a simple RL function to match assigned behaviors to those of experts, eliminating the need to manually define behavior-dependent rewards and address conflicting objectives in diverse datasets. This approach offers controllability and reduces the implementation complexity, providing a promising avenue for training virtual agents to exhibit generalized and adaptive behaviors in complex environments.

In summary, our primary contributions include a novel crowd simulation training framework that blends imitation learning with deep RL, utilizing a parameterized reward function to enable adaptive, complexity-sensitive agent learning. We also introduce a multifaceted trajectory characterization metric that integrates various aspects of movement and crowd dynamics to define diverse individual behaviors. Additionally, our model can simultaneously leverage multiple datasets during the learning phase, allowing virtual agents to learn a wide range of behaviors observed in real human data.

We conduct a series of experiments, both quantitative and qualitative, to assess our model's performance on both training and unseen data. The results reveal that our method excels in providing flexibility, adaptability, and generalizability, as it is able not only to reproduce the behaviors observed in each dataset but also to successfully replicate them in novel synthetic and dynamic environments. We compare our work with two baseline models [Lee et al. 2018; Panayiotou et al. 2022] that implement a similar state representation, demonstrating that it performs better and more accurately matches the tracked statistics. Additionally, we further evaluate our model by comparing against the leading state-of-the-art method (GREIL-Crowds [Charalambous et al. 2023]). Finally, we execute a sensitivity analysis for the characterization metric, to assess how different values alter the agents' behavior, showing that varied values consistently influence the overall behavior of the agents.

2 RELATED WORK

Over the years, extensive research has been conducted in simulating crowds and understanding their behaviors, leading to the development of various techniques covering a wide array of scenarios and goals [van Toll and Pettré 2021]. Consequently, two primary approaches have emerged: *microscopic*, which concentrate on local navigation, interactions, and behavioral diversity, and *macroscopic*, which regards the crowd as a unified continuous entity, with lesser emphasis on diversity. More recently, the popularity of data-driven and RL methods has grown due to their promising results, and thus, our literature review focuses more on these methods, as they are more closely aligned with the core concepts of our research.

2.1 Data-Driven Crowds

In the context of microscopic simulation, data-driven and learning-based approaches have been employed to improve the manual crafting of rules or functions, as these approaches excel at capturing detailed behaviors. Early approaches involve searching databases with trajectories from crowd videos stored alongside representations of local states and actions [Kwon et al. 2008; Lai et al. 2005; Lee et al. 2007; Lerner et al. 2007; Zhao et al. 2017]. Thus, during run-time, agents compare their current states with those stored in the database and modify their actions accordingly. In that manner, the PAG Crowd proposed by Charalambous and Chrysanthou [Charalambous and Chrysanthou 2014] utilizes interconnected clustered databases encoded as graphs, enhancing querying efficiency. Similarly, Zhao et al. [Zhao et al. 2013] trained an artificial neural network to select suitable clusters based on input states, while Boatright et al. [Boatright et al. 2015] used context descriptors for grouping and learning a behavioral policy; this enables more efficient querying as agents access the trained model instead of a database. In addition, Ju et al. [Ju et al. 2010] presented a method blending existing crowd data to generate new motions applicable to various agents. However, these methods heavily rely on the quantity and quality of recorded data, influencing computational cost and complexity, while controllability is minimal. Generalizability is another challenge, as encountering new and unseen states often leads to accumulation of errors.

An alternative method for utilizing reference data was selected by [Guy et al. 2011; Paris et al. 2007; Pettré et al. 2009; van Basten et al. 2009] which leverages this data to tune crowd simulators parameters, mainly focusing on collision avoidance. Recent efforts incorporate real-world data with Deep Learning (DL) for crowd navigation. Alahi et al. [Alahi et al. 2016] employed a Recurrent Neural Network to predict agents' future trajectories, while Gupta et al. [Gupta et al. 2018] and Amirian et al. [Amirian et al. 2019a,b] used Generative Adversarial Networks to generate diverse trajectories.

Despite the potential of DL techniques, learned models rely on the diversity and quantity of the reference data. A study by Qiao et al. [Qiao et al. 2019] explored how training data and methods impact the ability of imitation models to replicate expert agent movements when applied to novel scenarios.

2.2 Reinforcement Learning Crowds

RL-based methods have proven to be effective for learning optimal solutions in sequential decision-making problems [Sutton and Barto 2018]. Recently, Kwiatkowski et al. [Kwiatkowski et al. 2022, 2023] explored the various deep RL methods for character control, while also studied the impact of different RL design decisions in crowd simulations, revealing that certain choices significantly enhance efficiency and performance. The potential of RL in crowd simulation was initially demonstrated by Treuille et al. [Treuille et al. 2007], who trained controllers for real-time character animation and collision avoidance. RL approaches have thereafter been widely used to train policies in crowd simulation [Godoy et al. 2015; Lee et al. 2018; Martinez-Gil et al. 2011]. For instance, Chen et al. [Chen et al. 2016] trained a model for robot navigation, selecting optimal velocities predicted by ORCA [Van den Berg et al. 2011]. Xu and Karamouzas [Xu and Karamouzas 2021] learned human-like collision avoidance behavior using knowledge distillation and RL to construct the reward function, based on human trajectory demonstrations. Some works achieved notable results by defining simple reward functions focusing on goal-seeking and collision avoidance [Casadiego and Pelechano 2015; Lee et al. 2018; Long et al. 2017; Martinez-Gil et al. 2017; Sun et al. 2019]. Haworth et al. [Haworth et al. 2020] combined multi-agent and hierarchical RL achieving emergent behaviors for physically-based character simulations. However, crafting these manual reward signals is complex, covering a wide spectrum of real behaviors is challenging, and policies remain constant throughout the simulation without enabling diverse behaviors among agents and the environment.

To address this, various works introduced parameterized reward functions affecting the agents' behavior during simulation. For instance, Lee et al. [Lee et al. 2021] presented an algorithm learning a parameterized family of motor skills from a single motion clip, Won and Lee [Won and Lee 2019] trained parametric controllers for body shape variation, while Hu et al. [Hu et al. 2023] proposed a multi-agent RL approach learning a parametric predictive collision avoidance and steering policy. More recently, Panayiotou et al. [Panayiotou et al. 2022] developed an RL-based framework with a single parameterized policy enabling a mixture of core behaviors (goal-seeking, collision avoidance, grouping, and interaction with environmental elements). However, even though it can generate heterogeneous agent behaviors efficiently, balancing the weights of the reward function requires manual work, while the lack of real-world data impact its efficiency and simulation's plausibility. Finally, Charalambous et al. [Charalambous et al. 2023], proposed GREIL-Crowds which utilize Deep Q-Learning to learn a model for pedestrian behaviors guided by reference crowd data, effectively encapsulating behaviors like goal-seeking, group formation, and wandering, with generalizability to unseen scenarios. Even though the latter work has similar properties to our work, it lacks behavioral control through the policy network, ignores the environment's structure, does not have a continues action space, and is trained once per dataset.

3 FRAMEWORK

This section describes the high-level pipeline of our framework aiming on learning controllable crowd behaviors that can be distributed to populate novel virtual environments. We train an RL agent-based model over a variety of behaviors observed in different real-world datasets simultaneously, able to produce realistic-looking crowd behaviors. The central concept of our work involves integrating pure RL with imitation learning. This fusion effectively eliminates the requirement for manually crafted reward functions, offering a significant advantage by mitigating the constraints associated with direct imitation learning discussed in Section 2. By combining these methodologies, we harness the strengths of both approaches, allowing for more robust and adaptable agent behaviors, while offering behavioral control during run-time.

The proposed system consists of three phases: *Crowd data characterization*, *Learning* and *Simulation* (see Figure 1). First, we acquire real-world trajectories extracted from real videos and we "characterize" them to be used in the next phase (Section 4.1). Then, during training, we build an exact copy of the environment from each dataset and spawn real agents and RL-agents concurrently. We aim on training RL-agents to learn how to behave (under specific conditions) based on the behaviors acquired by observing the movements of real agents. We employ a curriculum learning approach that gradually reduces the size of the observation space, and we incorporate a complexity metric both as observation and reward signal (Section 4.2). Finally, during inference, we can utilize our learned policy to populate novel virtual environments with agents exhibiting varied, plausible, and controllable behaviors.

Designed as a decentralized agent-based crowd simulator, this framework allows each agent to function independently. RL-agents make decisions based on acquired observations within a designated radius, capturing only partial information of the environment's current state. We model our agents as cylinders, navigating on a 2D plane, with their movements directed by assigning a preferred velocity at each simulation step.

4 LEARNING CONTROLLABLE CROWD BEHAVIORS

4.1 Crowd Data Characterization

We characterize real-world crowd data by calculating a *Complexity Score* (w_{comp}) for each trajectory. This score defines the complexity of behaviors observed in a specific trajectory and is a multifaceted measure that combines various aspects of movement and crowd dynamics. This particular step operates independently from the agents' training phase and is not a prerequisite for learning crowd behaviors. Nevertheless, the primary objective of introducing this characterization metric is to facilitate intuitive and real-time controllability over the agents' behaviors; a critical component that is frequently neglected by previous works.

The complexity score is a combination of three metrics. First, we consider *Movement Diversity Score (MDS)*, aiming on evaluating

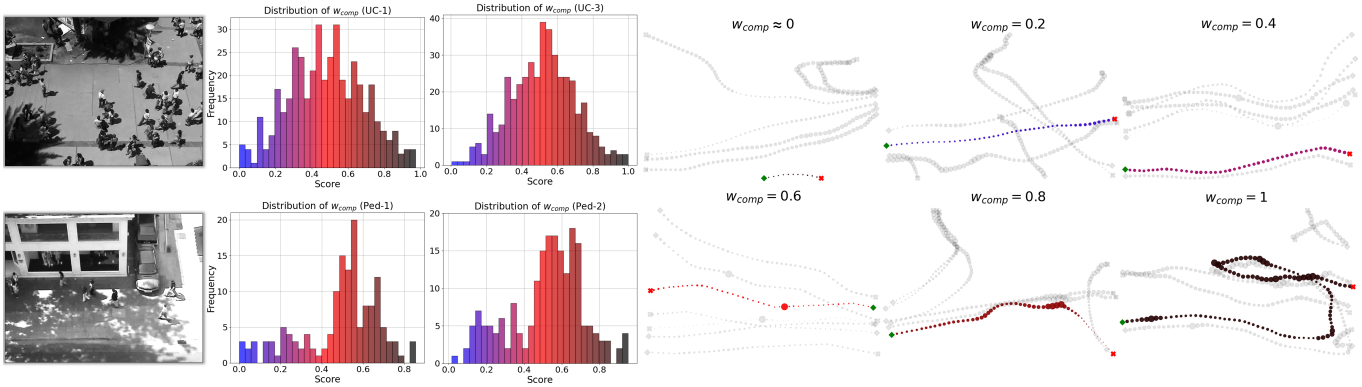

Fig. 2. Training datasets along with their w_{comp} distributions. We use four datasets UC (1,3) and Pedestrians (1,2) [Lerner et al. 2007] **(left)**. Examples of real-world trajectories, using different values of w_{comp}. We show the referenced trajectory (color-coded by its complexity value) along with five additional trajectories for the current time-window (grey); larger trajectory points indicate lower speed. **(right)**

the diversity of local movement. A higher overall score indicates more frequent changes in both speed and direction, revealing more diverse, or "complex" local navigation. We divide each trajectory $T_i = \{\mathbf{p_i^1}, \ldots, \mathbf{p_i^n}\}$, containing n points, in m segments $\{s_i^1, \ldots, s_i^m\}$ using a window of $4sec$, and get the average speed u_i^j and displacement vector $\hat{\mathbf{v}}_\mathbf{i}^\mathbf{j}$ for each; we chose this window size after experimenting with various durations, as it allows sufficient time for an individual to change direction or speed. We additionally calculate the change in direction between consecutive segment vectors, quantified as the dot product: $dir_i^j = \hat{\mathbf{v}}_\mathbf{i}^\mathbf{j} \cdot \hat{\mathbf{v}}_\mathbf{i}^\mathbf{j+1}$. An illustration of this process in shown in Figure 3. The standard deviations of speeds $\boldsymbol{\sigma}_{u_i}$ and direction changes $\boldsymbol{\sigma}_{dir_i}$ are calculated for every trajectory. These values are normalized against the maximum observed speed and direction standard deviations in the current dataset. We note that we sum the two standard deviations and then clamp the result in the range [0,1]. This approach was adopted after observing that averaging the standard deviations often leads to scenarios where a significantly low value in one of the two statistics highly impacts the overall score. Thus, for every T_i, we compute MDS_i as:

$$MDS_i = \boldsymbol{\sigma}_{u_i} + \boldsymbol{\sigma}_{dir_i}. \tag{1}$$

Second, we include a *Path Deviation Score (PDS)*, as the paths that deviate from the direct route to the goal position typically indicate a higher level of diverse behaviors. In real life, people generally tend to walk directly towards their destination, unless they engage in intermediate tasks that may cause them to deviate from their current path. We calculate PDS_i, for each T_i, using the Equation 2, setting the balance term $\alpha = 2$.

$$PDS_i = 1 - \frac{1}{1 + \alpha \left(\sum_{t=1}^{n-1} \|\mathbf{p_i^t} - \mathbf{p_i^{t+1}}\|_2 - \|\mathbf{p_i^1} - \mathbf{p_i^n}\|_2 \right)}. \tag{2}$$

Finally, we incorporate a *Grouping Dynamics Score (GDS)*, measuring the tendency of a person to perform any kind of behavior, shared with other people. We assume that when people engage in specific behaviors, while interacting with others at the same time, this should lead to an increase in the overall complexity score. Such situations typically require cooperation and coordination, which add layers of complexity. For every T_i, we calculate GDS_i by following the steps outlined below. First, we examine every other trajectory T_j that meets explicit criteria, both in terms of space and time. Specifically, the distance between the first position of T_i and T_j (spawn points) must be less than 2 meters and the difference between their initial timestep (spawn timesteps) less than 2 seconds. Then, for each qualifying trajectory pair (T_i, T_j), where $i \neq j$, we compute the distances between corresponding points at each timestep, denoted as $D_{i,j} = \{d_{i,j}^1, \ldots, d_{i,j}^n\}$. Subsequently, for each pair, we calculate the proportion, denoted by $Q_{i,j} \in [0, 1]$, of the trajectory length where each element in $D_{i,j}$ is less than a social distance threshold $d_{\text{social}} = 3.6m$; this value is defined as the "social distance" for interactions by Hall [Hall 1963]. We calculate GDS_i as:

$$GDS_i = \frac{1}{n} \sum_{j=1}^{n} \left(1 - d_{\text{close}}^{i,j} / d_{\text{social}}\right) \times Q_{\text{i,j}}, \tag{3}$$

where, $d_{\text{close}}^{i,j}$ is the average of all values in $D_{i,j}$. We characterize the behavior of each T_i and calculate its overall complexity score w_{comp}^i by combining the three metrics presented above using:

$$w_{comp}^i = .25 \times (MDS_i + PDS_i) + .5 \times GDS_i. \tag{4}$$

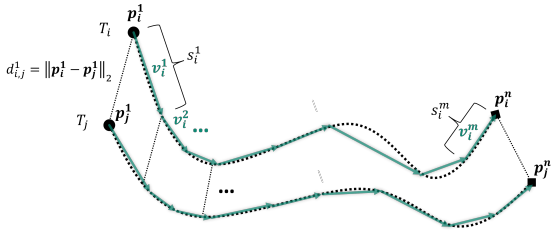

Fig. 3. Illustration of two trajectories T_i and T_j with selected notations introduced in Section 4.1. Every $T_i = \{\mathbf{p_i^1}, \ldots, \mathbf{p_i^n}\}$ is divided into m segments $\{s_i^1, \ldots, s_i^m\}$, and a displacement vector $\mathbf{v_i^j}$ is computed for each. The term $d_{i,j}^t$ defines the distance between corresponding points $(\mathbf{p_i^t}, \mathbf{p_j^t})$, at each timestep t.

We decide to equally combine MDS_i and PDS_i as they both capture the individual behavior of a person, while GDS_i defines the grouping dynamics in which individual behavior occurs. We normalize each trajectory complexity score using the minimum and maximum scores across all datasets to ensure consistency.

On the right part of Figure 2 we present sample trajectories, extracted from real-world data, for different values of w_{comp}. We observe that at lower values (0,.3), there is a dominant goal-seeking behavior where individuals follow the direct path towards their goal. For medium values (.4,.7), people may start walking in pairs, slightly deviate from their path, or perform sudden stops. Finally, at higher values (.8,1], we show that people exhibit a combination of actions leading to more complex crowd behaviors. An in-depth crowd behavior analysis is presented in Section 4.1.1.

To train our network, we select four available crowd datasets (see left part of Figure 2), a set of two datasets from a *University Campus* (1,3), and another set from a *Commercial Street - Pedestrians* (1,2); for simplicity we will refer to them as UC-1,3 and Ped-1,2 respectively. These datasets contain annotated trajectories of people captured at $25Hz$. The total duration of our training data is 19.5 minutes, with a total of 1194 individual tracked trajectories. We observe that these datasets contain both simple behaviors, like goal-seeking, and more complex behaviors, including wandering around, stationary and moving groups, interactions with the environment, sudden stops, walking and talking and more.

The characterized *training data* and *source code* can be accessed at https://github.com/veupnea/CEDRL.

4.1.1 Crowd Behavior Analysis. We perform a detailed analysis to examine the efficiency of our characterization process and the impact of different values of w_{comp} on the agents' behaviors. A list of frequent crowd behaviors, that can be observed in our training data, has been considered. We divide the selected behaviors in three types, **(a)** local movement behaviors, **(b)** whole path behaviors, and **(c)** grouping dynamics.

First, we consider *Stop* and *Accelerate*, where we split each trajectory in segments having a $4sec$ duration, and calculate the average speed for each. We determine whether a person is standing still, has started moving, or has suddenly stopped, by setting a speed threshold of $.1m/s$ and examining the state of the current and previous segment. Second, we identify if a person performs direct *Goal-Seeking* behavior or *Wander* around, by examining the ground truth path P_g. For each dataset, we use the environment map to construct a grid and apply a path-finding algorithm to determine the most efficient path, P_e, between the spawn and goal positions of each trajectory. We then compute the deviation ratio, $R_d = P_e/P_g$ where P_g is the actual path taken. Based on the dynamics in the training data, we set the threshold $R_d \geq .7$ to classify a trajectory as goal-seeking; otherwise, it is considered wandering. Finally, we evaluate grouping dynamics for each individual trajectory by focusing on how many other individuals the current person is interacting with. We define an interaction between two persons by considering their spatial and temporal similarities following the same procedure as in the calculation of *GDS* in Section 4.1.

In Figure 4 we present the fraction of each behavior across different w_{comp} intervals, by merging the data from all four training datasets. Then, we normalize the frequency of each interval using the total entries and show the comparative results. Regarding the local movement behaviors **(a)**, we observe that, for higher values, humans stop and accelerate with a higher frequency, leading to a logical increase in w_{comp}, as the overall speed deviation is larger. We note that stop occurrence is higher than accelerate as, especially in University Campus datasets, we notice a large number of stationary interactions between individuals, leading to a convergence in static behaviors. Second, in terms of the entire path behaviors **(b)**, goal-seeking, which is a simple and fundamental crowd behavior, has a high occurrence in low and medium complexity values. Wander behavior, which indicates the execution of intermediate tasks by people, appears in higher w_{comp} values. Third, overall grouping and interactions have high occurrence in medium and high complexity **(c)**. We show that groups of two persons (pairs) have the higher frequency in our training data. To conclude this analysis, it is evident that an increase in the complexity score is associated with a corresponding increase in the complexity of crowd behavior.

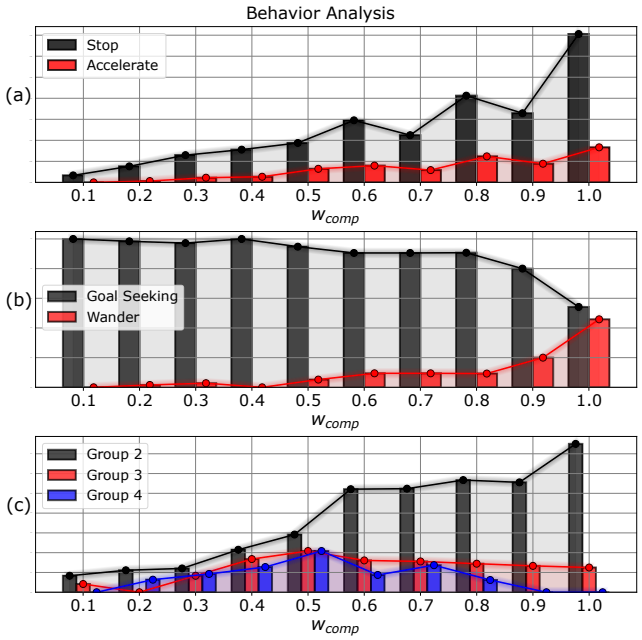

Fig. 4. Behavior occurrence analysis for various w_{comp} values across training datasets. We analyse **(a)** local movement behaviors, **(b)** whole path behaviors and **(c)** grouping dynamics.

4.2 Training Strategy

The first step of our learning pipeline is the characterization of the training data as described in Section 4.1. Then, we build the environment for each training dataset as shown on the left side of Figure 6. The training process starts by placing real agents in the scene based on the ground-truth timesteps and spawn/goal positions as appear in data. Each time a new real agent is spawned in the environment, we initialize an RL-agent at the same position applying random noise, both on position and look direction; at this

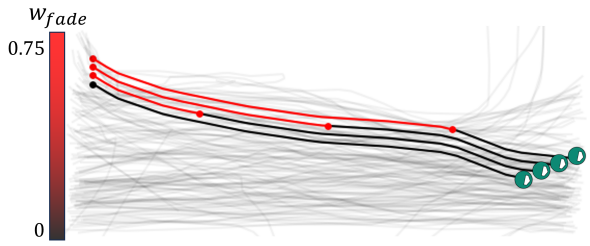

Fig. 5. During training we gradually decrease w_{fade}, controlling the percentage of the trajectory's duration that the RL-agent receives observations about the state of its assigned real agent.

point a new per-agent episode begins. Each episode ends when the real or RL-agent arrives at its assigned goal position or a collision with an obstacle occurs (refer to Table 2). We note that to increase the training efficiency, we spawn multiple RL-agents in the same environment (one for each real agent) concurrently, however, each RL-agent is not aware of the existence of the other RL-agents, nor do they affect its navigation. Each agent is decentralized and runs on its own episode, however still all agents train the same policy. We emphasize that our policy is trained simultaneously across all training datasets, instead of creating separate policies for each one, something that results in better scalability.

Our main objective is to train agents to mimic real behaviors and replicate them on novel environments. However, the end goal is to achieve this without any prior knowledge of the real agent's state, neither explicit guidance. Agents should solely rely on their own experience and the partial observations receiving for the current environmental state. Thus, we apply a *curriculum learning* approach during training, utilizing the global *Observation Fading Factor* w_{fade} that affects all agents in the scene. The training phase starts by providing RL-agent with information about the state of its assigned real agent for the $w_{fade} = 75\%$ of the trajectory's duration. An illustration of this concept is presented by Figure 5. We found that initiating training at that specific percentage consistently yields the best results, as agents get familiar from the beginning that at some point part of the observation space will be eliminated. Then, we gradually decrease w_{fade}, until reaching 0, which completely removes the observations related to real agent's state (Section 4.4). This process gradually enhances the agents' adaptability and further develops their behavior "knowledge". Finally, during the last training steps, our agents are able to navigate in the scene without guidance, exhibiting behaviors defined by different w_{comp} values.

4.3 Action Space

Our model generates actions that control the local movement of each agent. Thus, for every individual agent we set a preferred velocity $\mathbf{v_p^t}$ for each simulation step t by combining two continuous actions, a movement speed multiplier $m_d^t \in [-.5, 1]$ and a turning angle multiplier $m_a^t \in [-1, 1]$; we promote natural navigation by setting a higher forward movement probability. Then, for every simulation step t, the movement speed of an agent is calculated by $m_d^t \times 2.25m/s$ (maximum speed observed in the real-world data) and the turn angle by $m_a^t \times 90°/s$. Finally, we pass $\mathbf{v_p^t}$ to a Reciprocal Velocity Obstacles (RVO) [Van den Berg et al. 2008] simulator, to produce a collision-free future velocity, directing our agents' local movement. We note that our model makes a decision every $T_d = .2s$ and selects a suitable action, which repeats for every in-between simulation step $T = .04s$.

4.4 Observation Space

Each individual agent makes decisions by collecting a number of observations that partially describe the environment's current state. We utilize both scalar and visual observations. The observations of our agents are divided into two sets: **(a)** information about agent's current state and **(b)** information about the current state of their assigned real agent that are trying to imitate. We note that the local coordinate system of each agent is aligned with its look direction.

(a) RL-agent state. At each decision step t, we collect the *Local velocity* $\mathbf{v^t}$ relative to the look direction and the *Relative Goal Position* described by the tuple (ρ^t, θ^t), where $\rho^t \in [0, 1]$ is the distance to the goal normalized by a maximum environment distance, and $\theta^t \in [0, 1]$ is the normalized angle between the agent's look direction and the vector towards the goal position. Likewise, we include the w_{comp} value as observation, which characterize the type of the behavior that the agent is currently facing. In addition to the scalar observations above, we utilize two *Distance Sensors*, detecting real agents and obstacles, by casting 2×15 rays uniformly in front of the agent, on an angle 240° and distance $5m$. The purpose of this sensor is to simulate the human visual perception, as close as possible. Additionally, to capture surrounding contexts and hidden states that humans often perceive through other senses, such as hearing, we employ two *Visual Sensors*. Each sensor is defined as a 20×20 grid, detecting real agents and obstacles respectively, encoding the top-down view around an agent ($3m$ distance) as a PNG image. We note that the distance sensors and the visual sensors detect all other real agents in the environment, *excluding* the real agent assigned to the RL-agent. An illustration of how the two types of sensors are used is provided by Figure 6.

(b) Assigned Real agent state. This observation set provides information about the state of the assigned real agent relative to the

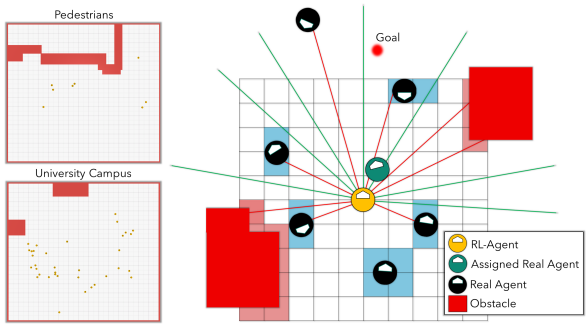

Fig. 6. Training environments with placed obstacles **(left)**. RL-agent state representation **(right)**. We spawn both RL-agents and real agents in the environment. For each RL-agent we assign a real agent that need to imitate. Our agents are equipped with distance sensors (laser lines) and visual sensors (grid) that detect all other real agents and obstacles.

local coordinate system of each RL-agent. We include the normalized *Relative Position*, *Relative Velocity*, *Relative Orientation* and *Distance*. The values of these observations are formed to always be in range [0,1], where lower values decreases the deviation, indicating better imitation performance and behavior replication by the RL-agent. This formulation is crucial, as when this part of observation space is eliminated (more in Section 4.2), these scalar observations are set to zero, indicating a perfect imitation quality. The complete observation space is summarized in Table 1.

Finally, we note that for all the observations described above, we employ a two-step stacking by integrating both current and previous observations into a single policy input. This approach equips the agent with a temporal component, enriching its understanding of the environment's state, including the direction and speed of movement of the observed entities.

Table 1. Observation Space.

Name	Description
Local Velocity ($\mathbf{v^t}$)	Relative to the agent's look direction.
Goal Position (ρ^t, θ^t)	Distance and angle towards goal position.
Complexity (w_{comp})	Characterization of current real trajectory.
Distance Sensors	Detecting Real agents and obstacles.
Visual Sensors	Detecting Real agents and obstacles.
Position	Position w.r.t. RL-agent's coord. system.
Velocity	Velocity w.r.t. RL-agent's coord. system.
Orientation	Orientation w.r.t. RL-agent's coord. system.
Proximity	Distance between RL-agent and Real agent.

4.5 Reward Function Design

The reward function is a crucial part in RL, as it sets the objective for algorithm optimization [Sutton and Barto 2018]. Defining reward terms for crowd simulation is a complex task, as people in real life often exhibit multiple behaviors while frequently displaying a blending of them. Thus, as crafting and balancing reward terms for capturing that large spectrum of behaviors is extremely challenging, we enable RL-agents to build a behavior "knowledge" by observing the movements of real agents. We employ a reward function that mainly evaluates the imitation performance of agents without the need to explicitly define behavior-specific reward terms. Thus, the following subsections describe the individual reward terms and present our collective reward function.

4.5.1 Evaluating Imitation Performance. We employ the *imitation quality* reward term R^t_{imi} to evaluate how well the RL-agent performs while trying to replicate its assigned real agent's behavior, at each simulation step t. This reward is mainly based on q^t_{imi}, which defines the imitation quality of the agent, and is calculated by considering the deviation in local movement between the RL-agent and its assigned real agent. We combine the following differences:

- *Speed*: $d^t_u = 1 - (\left|RL^t_u - real^t_u\right|)^{.5}$
- *Orientation*: $d^t_o = 1 - (\left|RL^t_o - real^t_o\right|)^{.5}$
- *Proximity*: $d^t_p = 1 - (\|\mathbf{RL^t_p} - \mathbf{real^t_p}\|_2)^{.5}$

For proximity, we set a maximum distance of $5m$ and normalize the corresponding distance in the range $[0, 1]$. We note that when calculating each reward, we apply the square root, rather than a linear approach, to enforce agents minimizing even further the deviations, in order to receive higher positive reward. Then, the overall imitation quality q_{imi} is calculated by:

$$q^t_{imi} = w_u * d^t_u + w_p * d^t_p + w_o * d^t_o, \tag{5}$$

where $w_u, w_p, w_o \in [0, 1]$ are individual constant weights for speed, orientation and proximity respectively, and also $w_u+w_p+w_o = 1$. We conducted a dedicated study to determine the optimal combination of $\{w_u, w_p, w_o\}$, available in the supplementary material; the results reveal that $\{w_u = .25, w_p = .5, w_o = .25\}$ are the optimal values. Finally, we incorporate w_{comp} into the imitation quality reward (see Table 2) to encourage agents adhere more closely to the behavior of the real agent, when faced with more complex behaviors. Moreover, this addition contributes to real-time controllability over the agents' behavior during inference.

4.5.2 Reward Function. We design a reward function encouraging RL-agents to perform similarly to how real agents behave, while avoiding collisions and moving smoothly. We utilize both *sparse* and *dense* reward signals (see Table 2) to construct our reward function $R(t)$, which calculates the reward at each simulation step t as:

$$R(t) = R^t_{goal} + R^t_{co} + R^t_{imi} + R^t_{smooth}. \tag{6}$$

We utilize sparse reward signals to highlight critical tasks for the agents, emphasizing those that yield significant positive or negative rewards. Agents receive a large positive reward R^t_{goal} if they reach their designated destination within a time frame that closely mirrors the actual data, specifically between 80% and 100% of the total trajectory's duration. This criterion encourages broader exploration, acts as a regularization technique, and deters agents from heading straight to their goal too soon. Moreover, we incorporate a large negative reward R^t_{co} for obstacle collisions. Although we use RVO for handling collision avoidance, as we have mentioned in Section 4.3, we still allow agents to contact with obstacles to understand their influence in the environment. Dense rewards are given at every training step and are responsible for rewarding or punishing the agents based on their immediate actions. Our main dense reward is R^t_{imi}, which has been introduced in Section 4.5.1. Additionally, we use a negative *smooth movement* reward R^t_{smooth} that penalizes

Table 2. Metrics and Reward Signals.

Weight	Range	Usage	Description
w_{comp}	$[0, 1.0]$	Training, Inference	Trajectory Complexity
w_{fade}	$[0, .75]$	Training	Observation Fading Factor
q^t_{imi}	$[0, 1.0]$	Training	Imitation Performance
d^t_{smooth}	$[0, 1.0]$	Training	Movement Difference

Event	Term	Reward	D	EE
Reached Goal	R^t_{goal}	$+.5 \times q^t_{imi}$	X	✓
Obstacle Collision	R^t_{co}	$-.5$	X	✓
Imitation Quality	R^t_{imi}	$+.005 \times w_{comp} \times q^t_{imi}$	✓	X
Smooth Movement	R^t_{smooth}	$-.001 \times d^t_{smooth}$	✓	X

D: Dense reward, **EE**: Episode ends

agents when their selected action leads to a high change in current velocity. We calculate the difference $d^t_{smooth} \in [0, 1]$ between current and future velocity, at each step t, and penalize the agent prepositionally if $d^t_{smooth} > .5$ (see Table 2).

5 EXPERIMENTS AND EVALUATION

We train our model over all four training datasets (as introduced in Section 4.1) *concurrently*, using the on-policy RL method Proximal Policy Optimization [Schulman et al. 2017] implemented by the Unity's ML-agents framework [Juliani et al. 2018], with the parameters presented in Table 3. We use a fully connected neural network with 3 hidden layers and 256 units, trained on a personal computer equipped with an Intel Core i9-14900K CPU, an Nvidia RTX 4090 GPU, and 64GB of RAM. The training was completed in approximately one day, using 32 simultaneous Unity application instances, each containing 12 environments.

We quantitatively and qualitatively evaluate our model by conducting a list of experiments. We evaluate our work against real-world data, two baseline models (Section 5.2), and the recent state-of-the-art work GREIL-Crowds [Charalambous et al. 2023] (more details in Section 5.5). For quantitative evaluations, we use relevant statistics and methods including the Fundamental Diagram (FD), Speed, Density, Distance to Nearest Neighbor (DNN), and Path Deviation (PD). For every timestep t, the local density d^t_a of an agent a is computed by Equation 7, as presented in [Helbing et al. 2007], where $\mathbf{p}^t_a$ and $\mathbf{p}^t_i$ are the position vectors of agent a and neighbor i at time t respectively, and $R_s = 3.6m$. DNN is calculated by considering the distance to the nearest neighbour at each timestep, where for PD we measure the distance from the current agent's position to the straight line connecting the spawn and goal positions.

$$d^t_a = \sum_{i \in N(a^t)} \frac{1}{\pi R_s^2} \exp(-\|\mathbf{p}^t_i - \mathbf{p}^t_a\|_2 / R_s^2). \tag{7}$$

In addition, we show qualitative results by providing generated trajectories and visual results for different scenarios, while also presenting various animated results in the provided supplementary video. We render simulations in the Unity Game Engine to demonstrate our model's capabilities, using a per-agent simple animation state machine; note that CEDRL and the animator operate independently and do not share any information about the agents' state.

Table 3. Simulation Parameters and Training Hyperparameters.

Parameter	Value	Description
r	$.25m$	Agent Radius
R_s	$5m$	Maximum Search Distance
G_s	$3m$	Visual Encoding Distance
T	$.04s$	Simulation Step
T_d	$.2s$	Decision Step
Learning Rate	3e-4	For Gradient Descent Updates
γ	.995	Discount factor
Epochs	3	Training Epochs
Batch Size	2048	Batch Size
Buffer Size	20480	Buffer Size
β	7.5e-3	Entropy Regularization Strength
ϵ	.2	Divergence Threshold
Time Horizon	2048	Steps for Experience Buffer

5.1 Ablation Study

We first conduct an ablation study to evaluate the influence of the underlying collision avoidance algorithm (RVO) on agent behavior and navigation. Specifically we compare *CEDRL* against two modified versions, **(a)** *w/o RVO*: the already trained model by removing RVO and let our policy set the velocity of the agents directly, and **(b)** $R_{co-agent}$: retraining our policy by incorporating another reward term $R^t_{co-agent} = -.05$ that penalize agents when colliding with other agents, at each simulation step t; the episode does not end when an agent-agent collision occurs.

We analyze the impact of RVO on agent behavior using the whole Ped-1 training dataset. First, we compute the w_{comp} distribution for Real data, CEDRL, and the two variations of our model. Next, we calculate the Kullback–Leibler divergence (KLD) [Kullback and Leibler 1951] between the three simulators and the real data (lower KLD indicates better distribution alignment), and present the results in the last column of Table 4. The results indicate that incorporating RVO during training preserves agent behavior, unlike removing it from the pre-trained model which have a negative influence.

Table 4. Ablation Study.

Metrics	ϵ	ΔV_i	ΔA_i	E_i	$steerE_i$	$w_{comp} \downarrow$
Real	.001	2.323	57.2	2705.4	1785.7	-
CEDRL	.000	2.253	222.7	2178.8	1580.1	**.135**
w/o RVO	.064	1.811	220.2	1765.3	1465.3	.165
$R_{co-agent}$	.015	3.157	651.3	3914.8	2994.5	.137

Additionally, we assess navigation dynamics using a list of metrics, as presented in [Zhao et al. 2017]. In particular, for each agent i, we use the collision rate ϵ, change in speed $\Delta V_i(ms^{-1})$, change of angle $\Delta A_i(^\circ)$, biomechanical energy $E_i(J)$, and biomechanical energy to avoid collisions $steerE_i(J)$. Table 4 summarizes the results for each metric. We note that CEDRL experiences almost no collisions due to RVO, w/o RVO agents collide more frequently due to the absence of a collision handling mechanism, whereas $R_{co-agent}$ agents generate a minimal number of collisions. Next, we show that replacing RVO with a dedicated collision reward leads to less smooth navigation as the collision avoidance is not performed efficiently. Agents experience higher speeds and angle changes, expending more energy on navigation and collision avoidance. Likewise, as shown in Figure 7, the $R_{co-agent}$ agents sometimes show rough, unnatural movements when they avoid collisions. While a more refined collision avoidance reward could improve results, would increase the complexity of the reward function, making it challenging to balance behavior-specific reward terms with those for collision avoidance. Therefore, we argue that allowing an underlying algorithm to handle collisions balances the complexity of the reward function with smooth navigation, without impacting the agent behaviors.

5.2 Setup

We conduct our evaluations on various environments to fully examine the capabilities of our model. Specifically, we design the following four scenarios: **(a)** the replicated environment from the real corresponding dataset, **(b)** an infinite environment with size

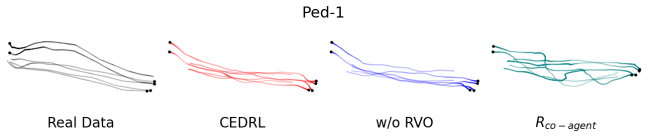

Fig. 7. Simulated trajectories for a snapshot of Ped-1 dataset with 6 agents, from real data, CEDRL, and two modified versions; lower color intensity indicates higher speed.

$20x20m$ enabling a continuous flow of agents, **(c)** benchmark setups (Hallway, Crossing, Circular), and **(d)** an infinite dynamically changing environment. The evaluation related to the latter, accompanied by a w_{comp} distribution reproduction study, is presented in the supplementary material provided.

We compare our model against real-world seen and unseen data, while also we utilize two baseline models, one offering behavior diversity (CCP [Panayiotou et al. 2022]), and an RL-based method focusing only on goal-seeking and collision avoidance (Lee et al. [Lee et al. 2018]). First, we train the *CCP-Density* simulator, which is a modified version of CCP. We state that a direct comparison against the original CCP simulator would not enable a fair assessment as it has not been trained on real-world data. Likewise, selecting the dynamic optimal parameters (agents' profiles) for every scenario is a very challenging task, that requires individual further investigation. Thus, we select to train an RL controller responsible for selecting agents' profiles overtime, trying to match the desired speed of a given current density. Specifically, we construct the FDs for Ped-1 and UC-3 datasets, encoding speed over density. For each dataset we train a separate per agent controller that runs on-top of CCP and selects an agent's profile every 5 seconds. At each decision step, CCP-Density agent gets the following observations: current velocity, current density, preferred speed, and a partial state of the environment using a distance sensor. Then, the agent selects three continuous values defining the current CCP profile (goal-seeking, grouping, interaction); we keep collision avoidance weight constant and set its value equal to .5 for simplicity. Moving to the reward function, agents aim to minimize the difference between the current and preferred speed (for the current density), thus the reward at each decision step is given by: $-d_v * d_v$, where d_v is the different between the current and preferred agent's speed. As a second baseline model,

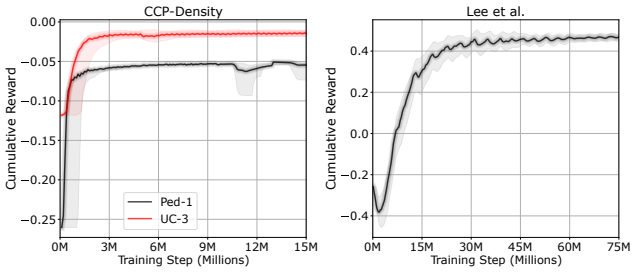

Fig. 8. Baseline models. We train a simple controller that selects CCP [Panayiotou et al. 2022] profiles defining the agents behavior **(left)**, and we reproduce the work from Lee et al. [Lee et al. 2018] **(right)**.

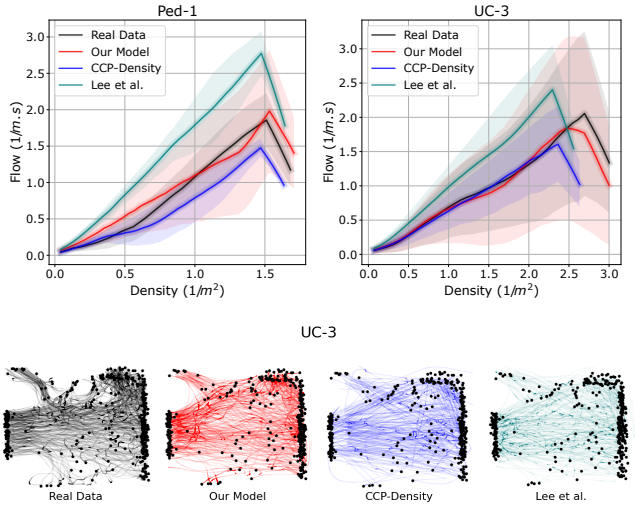

Fig. 9. Fundamental diagrams for two training datasets **(top)**. Simulated trajectories for the UC-3 dataset, generated by each model; lower color intensity indicates higher speed **(bottom)**.

we replicate the work from [Lee et al. 2018] setting the maximum speed similar to that of our model, and training it over both Ped-1 and UC-3 datasets concurrently. We note that while this work does not focus on behavior diversity, we include it as implements a similar state representation and serves as a baseline deep RL-based method for crowd navigation. Figure 8 demonstrates the training curves for the two baseline models.

5.3 Performance on Training Real-World Data

In this series of experiments, we assess the performance of our model using real-world training data and compare it with the two baseline models introduced in Section 5.2. First, we employ the FD by plotting *Flow* over *Density* trends for two real-world datasets, showing simulations generated by our and the baselines models; note that we use the whole duration the datasets and not part/s of them. Thus, for each simulation, we populate the scene using the ground-truth spawn timings, and spawn and goal positions. For the simulations generated by our model, we set each agent's w_{comp} equal to this of the corresponding real agent, and for CCP-Density runs we choose the model trained on the corresponding dataset.

The upper part of Figure 9 presents the FD for Ped-1 and UC-3 datasets (generated by each model), while the bottom part illustrates sample trajectories for the latter. The trends from the FD indicate that our model behaves similarly to the real-world data, and outperforms the two baseline models. CCP-Density is close to the real-world trend, however it struggles with higher densities due to the lack of real-world data during training, and the way that approaches local navigation and collision avoidance. On the other hand, the second baseline model is expected to have higher flow as neglects any other behaviors and guides agents directly towards their goal. Generally, initially we show that, as density increases, the flow increases too, where at some point higher densities slow down the agents. Likewise, the trajectories from our model show a

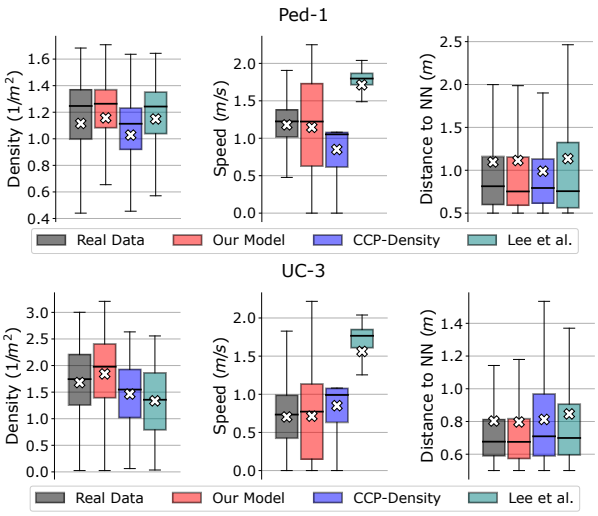

Fig. 10. Comparative analysis using two training crowd datasets. We present *Density*, *Speed* and *DNN*.

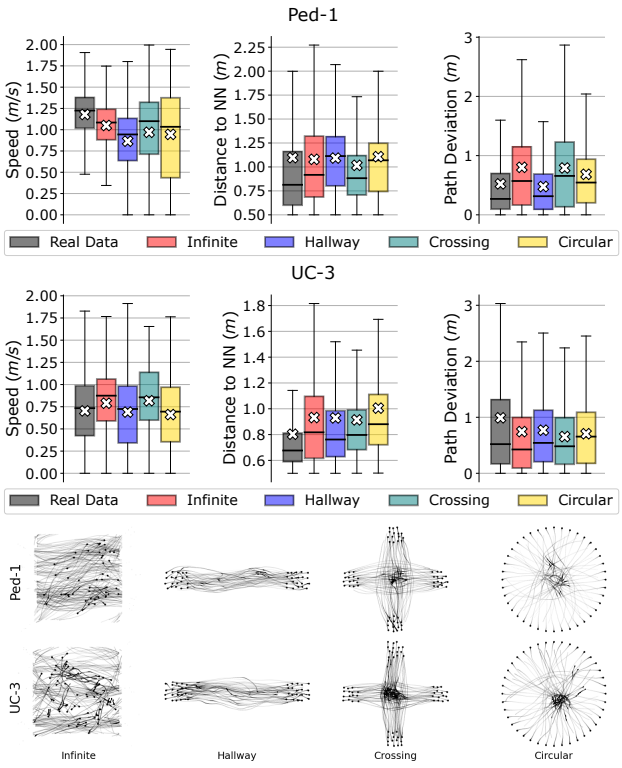

Fig. 11. Comparative analysis on various benchmark environments. We present *Speed*, *DNN*, and *Path Deviation*, accompanied by simulated trajectories.

higher behavioral similarity, successfully replicating both moving and static behaviors found in UC-3 dataset.

In addition to the FD, we calculate individual statistics including *Speed*, *Density*, and *DNN*, presenting the results in Figure 10. We show that our model can better capture the range of values for each statistics, compared to the two baseline models. While CCP-Density performs better than Lee et al., fails to the meet dataset's requirements related to speed. As expected, the second baseline model completely overlooks behaviors occurring at lower speeds.

5.3.1 Benchmark Environments. We conduct environment-specific experiments to assess our model's ability to replicate behaviors observed in real-world data across a range of synthetic benchmark scenarios. To thoroughly evaluate replication performance, we collect and analyze statistics from various setups, detailed as follows:

- *Infinite*: spawn 50 agents in a 20×20m environment, setting random spawn/goal positions on the edges.
- *Hallway*: spawn two groups of 20 agents each in a 30×15m environment, setting their goal position to the opposite side.
- *Crossing*: spawn four groups of 15 agents each in a 30×30m environment, setting their goal position to the opposite side.
- *Circular*: spawn 40 agents on the perimeter of a circle with radius 10m, setting their goal position to the opposite side.

For each simulation, we sample per-agent w_{comp} values from the distributions of different real-world datasets and assign them to agents accordingly. In Figure 11 we show results for *Speed*, *DNN*, and *PD*, accompanied by the generated trajectories; we mention that *Density* statistic is not utilized for this experiment, as is strongly influenced by the number of agents and area size, which need to be balanced across various setups. The results indicate that in the Ped-1 dataset, the model is able to reproduce the observed behaviors, which are mainly goal-seeking and moving in pairs/groups. Particularly, the consistency of speed with real-world data is maintained in most environments, except in the Hallway scenario, where the available space for navigation is limited, and in the Circular scenario, where numerous agents gather towards the center. In terms of PD, we observe a greater deviation in the Infinite and Crossing scenarios, where more space is provided for navigation. Moving to UC-3 dataset analysis, we observe slightly better coverage in Speed and PD statistics, however we notice a higher DNN across all scenarios. Finally, we can clearly support that, by comparing the statistics between the two datasets, the average speed and DNN are lower in the UC-3 dataset, compared to Ped-1. This makes sense as the former dataset exhibits a higher occurrence of stationary and interactive behaviors.

5.4 Generalization to Unseen Real-World Data

In this subsection, we evaluate the generalization performance of our model on unseen real-world data, using similar statistics with those in Section 5.3. We employ two additional crowd datasets, Ped-3 [Lerner et al. 2007] and Flock [Charalambous et al. 2014]. Ped-3 is another Pedestrians dataset, containing 180 tracked trajectories with a total duration of 5 minutes, where Flock showcases a group of 24 individuals walking across a churchyard for 30 seconds, simulating a human flock-like behavior. Figure 12 presents the fundamental diagram for each dataset, accompanied by sample trajectories; we note that these datasets are also unseen by the two baseline models. We observe that our model can better match the trend of the

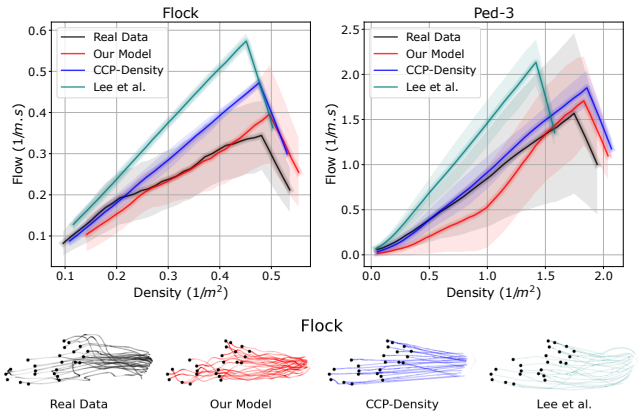

Fig. 12. Fundamental diagrams for two unseen datasets **(top)**. Simulated trajectories for Flock dataset, generated by each model; lower color intensity indicates higher speed **(bottom)**.

Flock dataset, compared to the baseline models. Regarding the Ped-3 dataset, we initially notice a lower flow at low to medium densities, yet it exhibits superior performance at higher densities and more comprehensively covers the entire flow spectrum compared to CCP-Density. The results demonstrate that our model can generalize to unseen real-world datasets, both those with similar environmental structures and behavior distributions (Ped-3) and those that are completely novel (Flock).

Continuing the analysis on generalization, we calculate additional individual statistics. In Figure 13 we plot *Density*, *Speed*, and *DNN* for the two unseen datasets. First, our model performs well on the Flock dataset across all statistics, notably outperforming the two baseline models in speed. Unlike the other models, which neglect

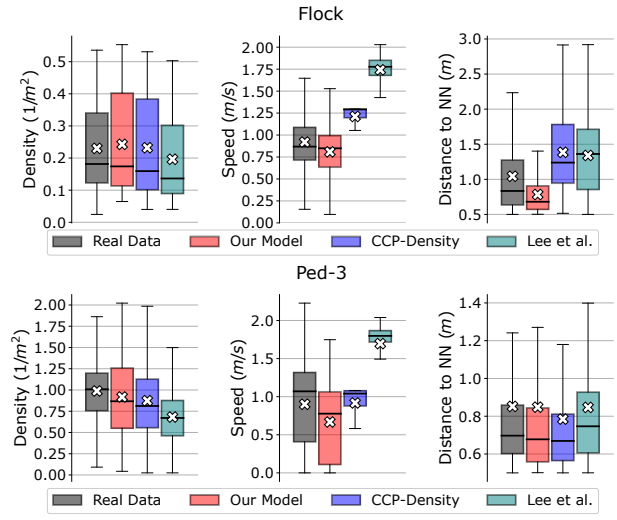

Fig. 13. Comparative analysis using two unseen crowd datasets. We present *Density*, *Speed*, and *DNN*.

more complex behaviors and direct agents straight to the goal, CEDRL captures a broader range of behaviors; this can be noticed in the trajectories presented at the bottom section of Figure 12. Moving to the Ped-3 dataset, our model is able to accurately match Density and DNN. As previously showed in the FD of this dataset, our model produces a lower average speed but more accurately captures the ground-truth speed range, compared to the baseline models. Animated results can be viewed in the supplementary video provided.

5.5 Comparison with a Guided-RL Approach

In this section, we evaluate our model by comparing it to the recent guided RL approach, GREIL-Crowds [Charalambous et al. 2023]. This method was chosen for a separate comparison because it diverges from conventional deep RL techniques and uses a distinct policy state space that omits environmental elements. To conduct this evaluation, we derive a set of scenarios in which GREIL was trained and perform both quantitative and qualitative assessments. We apply our model (which was trained simultaneously on multiple datasets), and highlight that the Ped-3 and Flock datasets were excluded during training. It is important to note that GREIL-Crowds uses policies specifically trained for each dataset. To ensure a fair comparison, we also exclude any environmental elements from the state of our model.

Beginning with the *quantitative* evaluation, we construct the FDs for the three available scenarios (Ped-2, Ped-3, and Flock), presenting the results in Figure 14. First, with respect to Ped-2, we observe that our model can more closely match the dynamics of this dataset, with GREIL producing higher flows as it struggles with the stationary behaviors occurring in the current data. Second, regarding the Ped-3 dataset, in addition to the default dataset's characteristics, we include the trends of a crossing scenario, where we increase the agent density by adding a rotated version of the data in the environment. In this scenario, while both models match the dynamics of the default data, we interestingly see that they both perform better in the increased density setup. Lastly, in the Flock scenario, although GREIL shows marginally better performance, our model displays a trend comparable to the real-world data, despite this dataset not being included in the training.

Although the FD offers a general overview of crowd performance, the *qualitative* evaluation offers a more precise understanding of

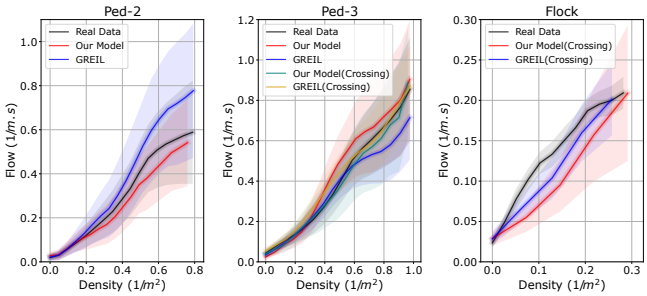

Fig. 14. Fundamental diagrams presenting trends generated from our model and GREIL-Crowds, for three distinct datasets and a total of four scenarios.

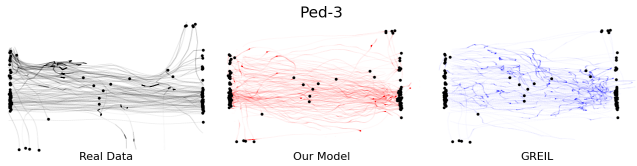

Fig. 15. Simulated trajectories for Ped-3 dataset, generated by our model and GREIL-Crowds; lower color intensity indicates higher speed.

crowd simulation quality, especially when focusing on diverse behaviors. Figure 15 presents the tracked trajectories for the Ped-3 dataset, as simulated by the two models. Although both models succeed in guiding agents to their assigned destination and matching observed behaviors, our model tends to produce smoother navigation. In particular, we show that the discrete action space in GREIL detracts from the plausibility of the simulation, while the movement of the agents is sometimes negatively impacted by nearby passing agents, leading to tangled trajectories due to unexpected starts, stops, or direction changes. Although these insights can be gathered from the provided figure, we strongly recommend that the readers refer to the supplementary video for a more comprehensive understanding of the results.

In conclusion, the results demonstrate that our method achieves state-of-the-art performance, delivering smoother simulations even in unseen scenarios. Moreover, our approach follows a simpler learning process compared to GREIL's methodology, enhances significant control over agents' behaviors, integrates environmental factors into the agent state, and utilizes a continuous action space.

5.6 Complexity Sensitivity Analysis

A key aspect of our framework is its ability to enable real-time controllability over the agents' behaviors. Thus, in this section we conduct experiments in various environments to examine how different w_{comp} values influence the generated actions. We perform our assessments in various scenarios: **(a)** real-world training (Ped-1) and unseen (Flock) environments, and **(b)** the four benchmark environments, as previously presented in Section 5.3.1. Specifically, we assign the same w_{comp} value to all agents and run individual simulations increasing the complexity metric by an interval of .1. Then, we compute *Speed*, *DNN* and *Path Deviation*, illustrating the results in Figure 16. Moreover, we show simulated trajectories for selected environments with varying complexity values in Figure 17. The statistics indicate that Speed and Path Deviation are significantly influenced by increased complexity values, a trend that is consistent across all scenarios. Furthermore, we show that DNN remains relatively stable in the benchmark scenarios, with occasional fluctuations, but without any clear upward or downward trend. This could be attributed to the challenges posed by the benchmark environments and their unnatural configurations, where multiple agents are spawned simultaneously with similar destinations and tasks, coupled with the presence of high-density areas; agents encounter a similar situation on the Flock scenario too. In contrast, in the Ped-1 scenario, where the agent spawning and setup is more natural, we observe a downward trend in DDN as the complexity metric increases, indicating higher level of interacting behaviors.

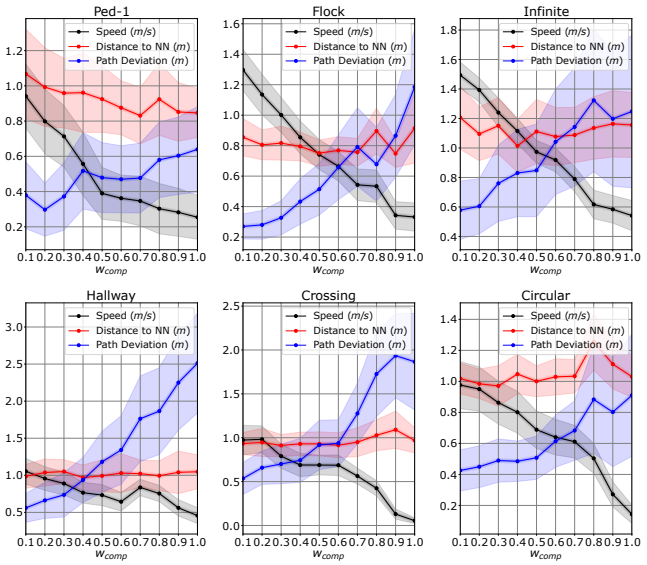

Fig. 16. Complexity Sensitivity (Quantitative). We present *Speed*, *DNN* and *Path Deviation*, over different w_{comp} intervals, for training data (Ped-1), unseen data (Flock), and benchmark setups (Infinite, Hallway, Crossing, and Circular).

Subsequently, moving to the qualitative results, presented in Figure 17, we can also justify the change over the agents' behaviors under different complexity values. First, in relation to the Ped-1 dataset, for low complexities ([.1, .3]) agents perform mainly goal-seeking behavior and walk directly to their goal with high speed. For medium values ([.4, .7]), we observe a mixture of behaviors; some agents exhibit goal-seeking, while others wander around and participate in static interactions. Finally, at higher values, the overall speed of the agents decreases and their trajectories become more disorganized due to the emergence of complex intermediate behaviors. Focusing on the presented benchmark environments, the key observation is the increased Path Deviation for higher complexity values. Interestingly, in the Circular setup with $w_{comp} \geq .7$, agents appear to move more efficiently as intermediate behaviors drive them to avoid the center of the circle, where bottlenecks occur at lower values, due to the nature of this setup. It is observed that in the circular environment, agents initially tend to move towards the left side. This behavior arises due to the structure and symmetry of the environment, coupled with the unnatural spawning of agents, where the RVO simulator consistently favors the left side to avoid any initial collisions.

Concluding this study, we contend that varying complexity values influence the agents' behaviors and these changes are observable and consistent across different environments, both real and synthetic. We strongly encourage readers to watch the supplementary video for this study's animated results, accompanied by a demonstration of real-time w_{comp} value manipulation.

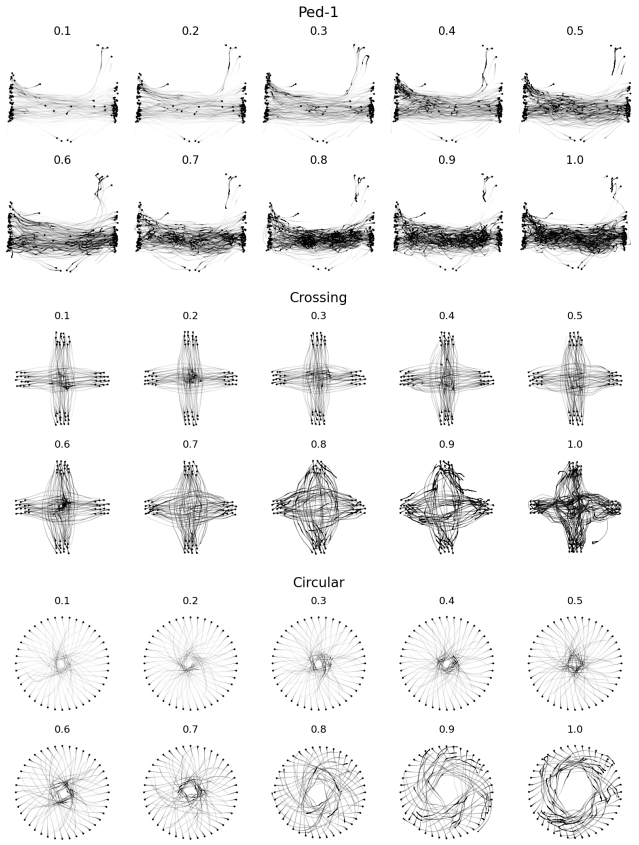

Fig. 17. Complexity Sensitivity (Qualitative). We show simulated trajectories, over various w_{comp} intervals, for real-world data and benchmark (Crossing and Circular) setups; lower color intensity indicates higher speed.

6 DISCUSSION AND FUTURE WORK

In this work, we present CEDRL, a simple RL-based framework allowing virtual agents to perform various behaviors observed in real-world data. These behaviors spanning from simple ones, such as goal-seeking, to more complex ones like wandering around, group-formations, static interactions, moving pairs, and more. In Figure 18 we display rendered simulations, showcasing a variety of agent behaviors in action. Our system allows users to populate virtual environments with diverse agents, adjusting their behavior accordingly, even in runtime. Unlike traditional RL-based approaches, our methodology combines imitation learning characteristics with deep RL, and gradually transitions agents from guided to pure RL during training, eliminating the need for defining individual behavior-specific reward signals. This enables agents to acquire a behavior knowledge by observing the movements of humans. Moreover, to enable controllability, a feature often lacking in current works, we employ a trajectory characterization metric. This metric is a multi-faceted measure that combines various aspects of movement and crowd dynamics, which is incorporated as input to our policy, influencing our parametric reward function.

We conduct both quantitative and qualitative evaluations and assess the performance of our model over training and unseen real-world data. The results demonstrate that our model not only reproduces the behaviors observed in each dataset, but also transfers them on novel synthetic environments. It performs better compared to the baseline models, achieves state-of-the-art performance and offers a broader range of capabilities and features. Our analysis of the characterization metric reveals that various values consistently influence the overall behavior of the agents, in different environmental settings, resulting in a diverse crowd simulator.

Autonomy of agents. Despite the fact that our method can produce a wide range of behaviors, the fact that each agent acts independently shows its impact. Although our training data cover a wide spectrum of behaviors, they may not contain more intrigue and rare interacting ones. We would like to explore methods for refining our learning strategy to include cooperation between agents, improving the simulation at a microscopic level.

Reward function design. We show that defining a simple reward function, considering only local movement characteristics (position, velocity, and orientation) enables agents to learn and replicate most of the behaviors presented. However, in certain cases where a specific behavior is influenced by the state of neighboring agents, our agents may struggle to perform as expected. Thus, additional research is needed to incorporate terms into the reward function that considers not just the imitation quality of individual real agents, but also their interactions with others and the environment.

Group dynamics. Our framework allows the presence of group dynamics, as agents perform both moving and static interacting behaviors. However, group formation is implicitly influenced by spawn/goal positions and timing. For instance, a group of agents initialized together, having similar goals, will stay within their group. Thus, it will be interesting to explore ways of enhancing the group dynamics of our simulator. A possible approach would be to incorporate explicit group constraints, as presented by Ren et al. [Ren et al. 2016], during our training phase.

Complexity weight as authoring tool. We use w_{comp} to define the high-level behavior characteristics of an agent, as a way of providing simulation authoring capabilities. While this is an easy and effective way for controlling the agents' behaviors, it may neglect finer differences between individual behaviors due to the nature of the metric. A possible approach to partially overcome this is to use individual statistics as varying weights (similar to [Panayiotou et al. 2022]); however, this approach significantly increases complexity, affecting both the learning process and the end-user experience, as different weights must be precisely balanced for the agent to perform the desired behavior. A more promising direction is to explore ways of extracting and mapping behaviors to high-level human characteristics that will define, even dynamically, the actions of the agents. Furthermore, given that our framework distinguishes between data characterization and learning as separate elements, an enhanced characterization that includes learned components could also be applied and passed as input to our policy.

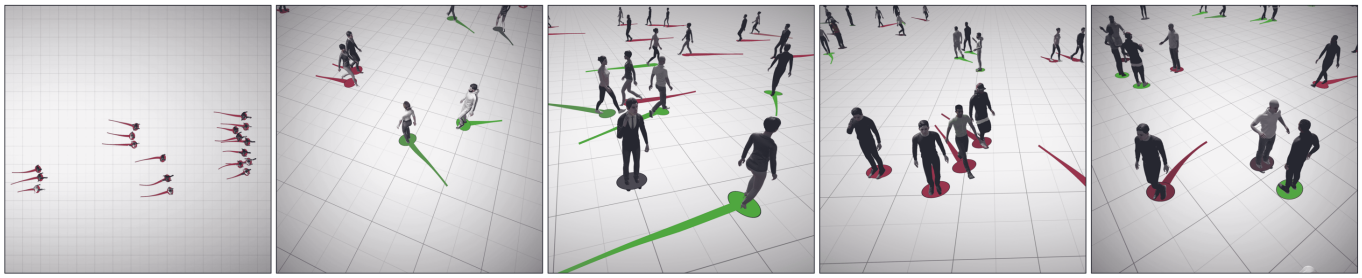

Fig. 18. Rendered simulations generated by our framework. We demonstrate the diverse behaviors our system can capture, including human flocks and both individual and collective goal-seeking activities, where people move either solo or in pairs/groups. Additionally, the system handles stationary behaviors and can manage a mixture of different behaviors concurrently.

ACKNOWLEDGMENTS

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 739578 and the Government of the Republic of Cyprus through the Deputy Ministry of Research, Innovation and Digital Policy.

REFERENCES

Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 961–971.

Javad Amirian, Jean-Bernard Hayet, and Julien Pettre. 2019a. Social Ways: Learning Multi-Modal Distributions of Pedestrian Trajectories With GANs. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2964–2972.

Javad Amirian, Wouter van Toll, Jean-Bernard Hayet, and Julien Pettré. 2019b. Data-Driven Crowd Simulation with Generative Adversarial Networks. In *Proceedings of the 32nd International Conference on Computer Animation and Social Agents* (Paris, France) *(CASA '19)*. Association for Computing Machinery, New York, NY, USA, 7–10.

Huikun Bi, Tianlu Mao, Zhaoqi Wang, and Zhigang Deng. 2020. A Deep Learning-Based Framework for Intersectional Traffic Simulation and Editing. *IEEE Transactions on Visualization and Computer Graphics* 26, 7 (2020).

Cory D. Boatright, Mubbasir Kapadia, Jennie M. Shapira, and Norman I. Badler. 2015. Generating a multiplicity of policies for agent steering in crowd simulation. *Computer Animation and Virtual Worlds* 26, 5 (2015).

Luiselena Casadiego and Nuria Pelechano. 2015. From One to Many: Simulating Groups of Agents with Reinforcement Learning Controllers. In *Intelligent Virtual Agents*. 119–123.

Panayiotis Charalambous and Yiorgos Chrysanthou. 2014. The PAG Crowd: A Graph Based Approach for Efficient Data-Driven Crowd Simulation. *Computer Graphics Forum* 33, 8 (2014), 95–108.

Panayiotis Charalambous, Ioannis Karamouzas, Stephen J. Guy, and Yiorgos Chrysanthou. 2014. A Data-Driven Framework for Visual Crowd Analysis. *Computer Graphics Forum* 33, 7 (2014), 41–50.

Panayiotis Charalambous, Julien Pettre, Vassilis Vassiliades, Yiorgos Chrysanthou, and Nuria Pelechano. 2023. GREIL-Crowds: Crowd Simulation with Deep Reinforcement Learning and Examples. *ACM Trans. Graph.* 42, 4, Article 137 (2023), 15 pages.

Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P. How. 2016. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2016), 285–292.

Julio E. Godoy, Ioannis Karamouzas, Stephen J. Guy, and Maria Gini. 2015. Adaptive learning for multi-agent navigation. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1577–1585.

Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. 2018. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2255–2264.

Stephen J. Guy, Sujeong Kim, Ming C. Lin, and Dinesh Manocha. 2011. Simulating Heterogeneous Crowd Behaviors Using Personality Trait Theory. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Vancouver, British Columbia, Canada) *(SCA '11)*. ACM, NY, USA, 43–52.

Edward T. Hall. 1963. A System for the Notation of Proxemic Behavior. *American Anthropologist* 65, 5 (1963), 1003–1026.

Brandon Haworth, Glen Berseth, Seonghyeon Moon, Petros Faloutsos, and Mubbasir Kapadia. 2020. Deep Integration of Physical Humanoid Control and Crowd Navigation. In *Proceedings of the ACM/SIGGRAPH Conference on Motion, Interaction and Games* (Virtual Event, SC, USA) *(MIG '20)*. ACM, NY, USA, Article 15, 10 pages.

Dirk Helbing, Anders Johansson, and Habib Zein Al-Abideen. 2007. Dynamics of crowd disasters: An empirical study. *Physical review E* 75, 4 (2007), 046109.

Kaidong Hu, Michael Brandon Haworth, Glen Berseth, Vladimir Pavlovic, Petros Faloutsos, and Mubbasir Kapadia. 2023. Heterogeneous Crowd Simulation using Parametric Reinforcement Learning. *IEEE Transactions on Visualization and Computer Graphics* 29, 4 (2023), 2036–2052.

Eunjung Ju, Myung Geol Choi, Minji Park, Jehee Lee, Kang Hoon Lee, and Shigeo Takahashi. 2010. Morphable Crowds. *ACM Trans. Graph.* 29, 6, Article 140 (2010), 10 pages.

Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, et al. 2018. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627* (2018).

Solomon Kullback and Richard A. Leibler. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics* 22, 1 (1951), 79–86.

Ariel Kwiatkowski, Eduardo Alvarado, Vicky Kalogeiton, C. Karen Liu, Julien Pettré, Michiel van de Panne, and Marie-Paule Cani. 2022. A Survey on Reinforcement Learning Methods in Character Animation. *Computer Graphics Forum* (2022), 1–27.

Ariel Kwiatkowski, Vicky Kalogeiton, Julien Pettré, and Marie-Paule Cani. 2023. Understanding reinforcement learned crowds. *Computers & Graphics* 110 (2023), 28–37.

T. Kwon, K. H. Lee, J. Lee, and S. Takahashi. 2008. Group motion editing. In *ACM Transactions on Graphics (TOG)*. ACM, New York, NY, United States, 80.

Yu-Chi Lai, Stephen Chenney, and ShaoHua Fan. 2005. Group motion graphs. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, Los Angeles, California, 281–290.

Jaedong Lee, Jungdam Won, and Jehee Lee. 2018. Crowd simulation by deep reinforcement learning. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*. ACM, Limassol, Cyprus, 1–7.

Kang Hoon Lee, Myung Geol Choi, Qyoun Hong, and Jehee Lee. 2007. A Data-driven Approach to Crowd Simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '07)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 109–118.

Seyoung Lee, Sunmin Lee, Yongwoo Lee, and Jehee Lee. 2021. Learning a Family of Motor Skills from a Single Motion Clip. *ACM Trans. Graph.* 40, 4, Article 93 (2021), 13 pages.

Marilena Lemonari, Rafael Blanco, Panayiotis Charalambous, Nuria Pelechano, Marios Avraamides, Julien Pettré, and Yiorgos Chrysanthou. 2022. Authoring Virtual Crowds: A Survey. *Computer Graphics Forum* (2022).

Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. 2007. Crowds by Example. *Computer Graphics Forum* 26, 3 (2007), 655–664.

Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. 2017. Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning.

Francisco Martinez-Gil, Miguel Lozano, and Fernando Fernández. 2011. Multi-Agent Reinforcement Learning for Simulating Pedestrian Navigation. In *International Workshop on Adaptive and Learning Agents*. Springer, Berlin, Heidelberg, Berlin, Heidelberg, 53.

Francisco Martinez-Gil, Miguel Lozano, and Fernando Fernández. 2017. Emergent behaviors and scalability for multi-agent reinforcement learning-based pedestrian models. *Simulation Modelling Practice and Theory* 74 (2017), 117–133.

Andreas Panayiotou, Theodoros Kyriakou, Marilena Lemonari, Yiorgos Chrysanthou, and Panayiotis Charalambous. 2022. CCP: Configurable Crowd Profiles. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques, Siggraph 2022*. ACM, New York, NY, USA, 10 pages.

Sebastien Paris, Julien Pettre, and Stephane Donikian. 2007. Pedestrian Reactive Navigation for Crowd Simulation: a Predictive Approach. *Computer Graphics Forum* 26, 3 (2007), 665–674.

Julien Pettré, Jan Ondrej, Anne-Hélène Olivier, Armel Crétual, and Stéphane Donikian. 2009. Experiment-based Modeling, Simulation and Validation of Interactions between Virtual Walkers. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, Louisiana, New Orleans, 189–198.

Gang Qiao, Honglu Zhou, Mubbasir Kapadia, Sejong Yoon, and Vladimir Pavlovic. 2019. Scenario Generalization of Data-driven Imitation Models in Crowd Simulation. In *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games* (Newcastle upon Tyne, United Kingdom) *(MIG '19)*. Association for Computing Machinery, New York, NY, USA, Article 36, 11 pages.

Z. Ren, Panayiotis Charalambous, Julien Bruneau, Qunsheng Peng, and Julien Pettré. 2016. Group Modeling: A Unified Velocity-Based Approach. *Computer Graphics Forum* 36, 8 (2016), 45–56.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR* abs/1707.06347 (2017).

Libo Sun, Jinfeng Zhai, and Wenhu Qin. 2019. Crowd Navigation in an Unknown and Dynamic Environment Based on Deep Reinforcement Learning. *IEEE Access* 7 (2019), 109544–109554.

Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Adrien Treuille, Yongjoon Lee, and Zoran Popović. 2007. Near-optimal Character Animation with Continuous Control. *ACM Trans. Graph.* 26 (2007).

B. van Basten, S. Jansen, and I. Karamouzas. 2009. Exploiting motion capture to enhance avoidance behaviour in games. *Motion in Games* 5884 (2009), 29–40.

Jur Van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. 2011. Reciprocal n-Body Collision Avoidance. In *Robotics Research*, Cédric Pradalier, Roland Siegwart, and Gerhard Hirzinger (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 3–19.

Jur Van den Berg, Ming Lin, and Dinesh Manocha. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *International Conference on Robotics & Automation*. IEEE, 1928–1935.

W. van Toll and J. Pettré. 2021. Algorithms for Microscopic Crowd Simulation: Advancements in the 2010s. *Computer Graphics Forum* 40, 2 (2021), 731–754.

D. Wolinski, S. J. Guy, A.-H. Olivier, M. Lin, D. Manocha, and J. Pettré. 2014. Parameter estimation and comparative evaluation of crowd simulations. *Computer Graphics Forum* 33, 2 (2014), 303–312.

Jungdam Won and Jehee Lee. 2019. Learning Body Shape Variation in Physics-Based Characters. *ACM Trans. Graph.* 38, 6, Article 207 (Nov. 2019), 12 pages.

Pei Xu and Ioannis Karamouzas. 2021. Human-Inspired Multi-Agent Navigation using Knowledge Distillation. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2021), 8105–8112.

M. Zhao, W. Cai, and S. J. Turner. 2017. CLUST: Simulating Realistic Crowd Behaviour by Mining Pattern from Crowd Videos. *Computer Graphics Forum* 37 (2017), 184–201.

M. Zhao, Stephen Turner, and Wentong Cai. 2013. A Data-Driven Crowd Simulation Model Based on Clustering and Classification. In *IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*. 125–134.