Heterogeneous Crowd Simulation using Parametric Reinforcement Learning

Kaidong Hu*, Brandon Haworth*, Glen Berseth, Vladimir Pavlovic, Petros Faloutsos, and Mubbasir Kapadia

Abstract—Agent-based synthetic crowd simulation affords the cost-effective large-scale simulation and animation of interacting digital humans. Model-based approaches have successfully generated a plethora of simulators with a variety of foundations. However, prior approaches have been based on statically defined models predicated on simplifying assumptions, limited video-based datasets, or homogeneous policies. Recent works have applied reinforcement learning to learn policies for navigation. However, these approaches may learn static homogeneous rules, are typically limited in their generalization to trained scenarios, and limited in their usability in synthetic crowd domains. In this paper, we present a multi-agent reinforcement learning-based approach that learns a parametric predictive collision avoidance and steering policy. We show that training over a parameter space produces a flexible model across crowd configurations. That is, our goal-conditioned approach learns a parametric policy that affords heterogeneous synthetic crowds. We propose a model-free approach without centralization of internal agent information, control signals, or agent communication. The model is extensively evaluated. The results show policy generalization across unseen scenarios, agent parameters, and out-of-distribution parameterizations. The learned model has comparable computational performance to traditional methods. Qualitatively the model produces both expected (laminar flow, shuffling, bottleneck) and unexpected (side-stepping) emergent qualitative behaviours, and quantitatively the approach is performant across measures of movement quality.

Index Terms—Multi-agent Navigation, Reinforcement Learning, Parametric Policy Learning

1 Introduction

Synthetic crowd simulation is an enabling technology for several fields. In particular, agent-based crowd simulators have found success in several applications, such as entertainment (film, TV, and games), data-driven environment design (procedural architecture, many worlds design), and safety evaluation (fire egress, disaster scenarios). An agent-based crowd simulator is usually composed of a hierarchy of methods resolved by various models. Often, crowd simulation is defined by its lowest level of control – steering. At this level, the agent perceives the environment and other agents and produces some action or action plan. These actions are designed to resolve at least two primary concerns: collision avoidance and goal-seeking movement. Decisions at the steering level are generally responsible for emergent behaviours in the simulated crowd.

However, defining a robust steering model is a complex trade-off between accuracy and performance. There are many ways to solve this problem. Most approaches consider one of data-driven [1], vision-based [2], space-time planning [3], [4], [5], rule-based [6], velocity-based [7], [8], force-based [9], [10], [11], and even composite [12] solutions to the problem. Because it is difficult to model the myriad factors which impact steering decisions, in real-time, and produce predictable and expected qualitative and quantitative results, often models choose one or more of these factors over the others. Therefore, a plethora of models with diverse capabilities has been developed suited for varying scenarios.

The problem space of steering in crowd simulation is well suited to Reinforcement Learning (RL) in that RL produces control policies, or a state-action mappings, for an agent in an environment given some reward signal. However, RL may require a broad sampling of the state space, which has led to many models that only work under restrictive conditions and typically with homogeneous agent definitions. Multi-Agent Reinforcement Learning (MARL) extends this by having multiple agents within the same shared environment where their actions may impact each other. The synthetic crowds problem is typically formulated in a multi-agent problem space. Recently, MARL approaches have been applied to multi-agent navigation, what we will refer to as crowds [13], [14], [15], [16], [17], [18], [19]. These methods have yet to support or evaluate fundamental features of synthetic crowds such as *generality*, *heterogeneity*, and *reciprocal predictive collision avoidance*. While models often perform well in particular scenarios, the ability to resolve scenarios with varying numbers of agents and environment complexity is essential. Synthetic crowd heterogeneity at the steering level often comes in the form of diversifying action updates. Perceptually, movement profiles have the largest impact on heterogeneity, crowd variety, and

- *Kaidong Hu, Vladimir Pavlovic, and Mubbasir Kapadia are at the Computer Science Department, Rutgers University.*
- *Brandon Haworth is at the Department of Computer Science, University of Victoria.*
- *Glen Berseth is at the Berkeley Artificial Intelligence Research (BAIR) Lab, University of California, Berkeley.*
- *Petros Faloutsos is at the Department of Electrical Engineering & Computer Science, York University, and is also with the University Health Network: Toronto Rehabilitation Institute.*

Manuscript received XXXXX; revised XXXXXX.

∗. These authors contributed equally to this work.

Fig. 1: Multiple interacting heterogeneous *HOP-RL* agents in a randomized scenario. The *HOP-RL* model has learned to continuously predictively avoid collisions with other heterogeneous agents and the environment.

naturalness [20], [21]. However, RL-based crowds methods do not afford heterogeneity in this manner. This is mainly due to the non-stationary problem of multiple learning agents in the same environment (an essential problem in MARL), which limits how *different* agents can be and still learn useful policies. More generally, this limitation also stems from the fact that machine learning models are fixed once learned (unless online learning or domain adaptation is used), or scenario-specific models are learned by training on only variations of that scenario. Finally, predictive, or anticipatory, avoidance has become commonplace in emerging model-based crowd simulators while in the early rule-based steering literature, models handled mostly reactive changes in velocity at every simulation time-step.

We propose the **H**eter**o**geneous crowds using **P**arametric **R**einforcement **L**earning (HOP-RL) approach to steering in synthetic crowd simulation, as seen in Figure 1. That is, a learned policy that does not rely on any single concrete definition of the rules translating states into actions, or, more specifically, how agents are supposed to move or interact with their environment. In our proposed approach, we attempt to address all prior shortcomings regarding *generality*, *heterogeneity*, and *anticipatory reciprocal collision avoidance* in RL-based crowds. We propose a model-free multi-agent reinforcement learning approach with goal conditioning and parameter sharing that does not rely on shared agent information, centralized control, or agent-agent communication. The end result is a model that learns generalized crowd steering with parametric agents from a simulated procedural environment using an explicit trial-and-error algorithm. Our goal in these contributions was to learn a MARL-based approach that performs on par with or better than traditional crowds steering methods both quantitatively and qualitatively, while also allowing a practitioner using our method (e.g. a game developer) to deploy diverse crowds simply by setting diverse input parameters. We specifically propose a learning-based method for crowds that affords performant directability, and we show that it is capable of learning more complex behaviours than traditional methods and even succeeding at scenarios where past methods can fail.

1) *Heterogeneity:* We propose that a parametric policy space can be learned by observing parameter(s) sampled during training. To make the MARL problem tractable, i.e. handle the non-stationary learning of multiple agents, we use parameter sharing. This method learns the desired policy using a single architecture that can be duplicated amongst all agents and evaluated independently. Ultimately this means the agents are homogeneous in their definition (they all use the same policy). However, we propose also using goal conditioning (passing goal information into the state observations), rewarding the desired goal behaviour, and affording some form of control related to the goal in the action space. We show that through this goal-conditioned learning, we can produce parametric heterogeneous crowds post-training. Agents learning using the proposed method have parameters that can be changed as needed and without retraining. We resolve both the heterogeneous and parametric requirements of crowds agents without producing multiple models. This supports the authoring of crowds for practitioners without need to engage in the learning process.
2) *Generality:* We propose using an expanded procedural scenario definition, from the crowd evaluation literature, which has a high likelihood of agent-agent and agent-environment interactions [22]. Our hypothesis is that the distribution of environments this procedural definition produces replicates the distribution of local interactions found in the broader scope of scenarios. While the generality of crowd steering models can be difficult to prove, we show that using our environment produces learned policies that generalize well to a battery of crowds scenarios from the comparative crowds analysis literature [23], [24], [25].
3) *Reciprocal predictive collision avoidance:* We propose that by making use of widening the neural network input to facilitate multiple state observations agents may learn reciprocal collision avoidance. We show that observing multiple prior state instances allows agents to learn predictive policies and avoid future collisions with other agents in a heterogeneous environment. This mitigates the partial observability

problem in our multi-agent reinforcement learning approach where we decentralize information sharing, control, and communication. We learn this behaviour through model-free, trial-and-error, learning in our simulated training environment. We show that our agents can learn to make expected anticipatory behaviours, as well as, avoid densely packed areas without additional information.

2 Related Work

Synthetic crowds, in particular, steering models, have a rich history in the literature, beginning with the originative work on flocking behaviours [6]. This history has since been covered in comprehensive surveys of the field [26], [27], [28], [29]. In this section, we address works related to crowd steering models and then explore more recent works in machine learning-based solutions. In particular, we focus on decentralized agent-based approaches to the crowd steering problem.

2.1 Learning Physical Character Control

As a precursor to this discussion we discuss a separate (though fast converging) related field of physical character control. The physical character control problem is typically a closed-loop approach to physically driving virtual characters. The approaches have a long history in the literature particularly the single character/agent solution–we focus on those which incorporate neural networks in their design. Early biped character control models recreated the neural oscillators found in mammals to produce walking patterns [30]. Numerous works have gone the route of using neural networks to learn control policies of humanoid bipeds. The approaches are similar to ours in that a neural network learns to map egocentric and proprioceptive sensory inputs appropriate actions (in the form of joint level control) [31], [32], [33]. More recent works have sought to improve the robustness of models. For example, using phase conditioned neural networks for producing cyclic humanoid control over varied environments [34]. This approach can be separated into two levels with phase conditioned lower-level controller for locomotion skills and higher-level control for producing more complex behaviours [35]. Recent has extended this approach into the crowds problem domain by using parameter sharing amongst a modified low-level controller to mitigate the non-stationary problem in MARL [36]. While these approaches learn a high dimensional problem, typically for one or few characters, they are often limited in the skill and user parameterization of the character. Work has been done to specifically address this by composing individual controllers which learn specific skills [37]. Similarly, modular policies can be learned and recomposed to control characters of differing morphologies [38]. Another approach uses RL to learn to mimic skills and blend between them using a parameter which can be directly user controlled [39]. Related to our work, it has been shown that a policy can be conditioned on body shape variation to learn a single parametric controller for a class of physically enabled characters of different body shapes [40]. This allows the user to control a single physically enabled character, within a range of body shapes for that morphology, using a single policy.

An additional problem space, related to physical character control, is human-robot interaction where we wish to model a physical robot navigating and performing tasks among groups of humans. It has been shown that robot navigation among humans can be learned by conditioning the policy on pairwise state observations of humans relative to the robot and coarse estimations of human-human interactions that allows the robot to further estimate future human movements [41]. A similar approach using relational graphs encodes the local relative interactions of the crowd in a way that graph learning techniques (like Graph Convolutional Networks) may then be used to estimate the movement updates of the humans in the scene [42].

2.2 Agent-based Crowd Simulation

There are several de-facto standard models in agent-based crowd steering with very different underlying approaches. Crowd simulation and steering algorithms have a long and rich history in the literature [29], [43], [44]. Physical approaches (force-based) model interactions with agents and the environment as forces that repel or attract the agent [9], [10], [11]. Velocity obstacle-based approaches decompose velocity-space such that collision-free movement updates can be guaranteed among agents [7], [8], [45], [46]. Vision-based models have been added to these models to enhance their performance [2]. Multi-phase approaches integrate multiple steering models to generate a final steering decision [12]. Space-time planning approaches have been formulated to address the complexities of biomechanical steering [4], [5]. Probabilistic approaches have successfully modelled time-variant behaviours over fields [3].

2.3 Learning and Crowd Simulation

Defining static crowd steering models has led to a plethora of simulators that typically focus on one element of steering to the detriment of others. It is difficult to define, a unique all-encompassing method that captures all the emergent effects of crowds and reproduces dynamics accurately under diverse conditions. Often, what works for one scenario does not work for many others.

The steering problem in synthetic crowds is a high dimensionality semi-chaotic open-loop system – and thus it is a fundamentally difficult problem to solve. To address this high dimensionality problem, methods have been proposed to help tune, optimize, and explore the emergent properties of a given model by learning optimal steering parameters for crowd-centric outcomes measures [47].

Several machine learning methods have been proposed to address the chaotic open-loop nature of the problem. Unsupervised approaches have been used to learn trajectory models rather than simulation time steering actions. For example, data-driven learning of trajectory models may be done via unsupervised clustering [48]. Deep learning has been used to learn previously existing models through a form of behaviour cloning, such as with *RVO* [49]. Similarly, it is possible to decompose the problem into (1) a pairwise single agent RL collision avoidance that is then (2) constrained using a known crowd model (ORCA) to produce

a multi-agent collision free decision [50]. (Recent work has used a pool of LSTM networks to simulate trajectories end-to-end for crowd simulation [51]. While these methods may learn important behaviours such as reciprocal collision avoidance, or they may reproduce density distribution similar to their real-world counterparts, they are largely limited to replicating their source data or model. This leads to new models that have the same issues as the source model, or models which only work in limited scenarios (i.e., that reflect the distribution of the data source).

Recently, a different perspective on similar problem space has emerged in the form of local trajectory prediction methods. These methods formulate the agent movement update as a local trajectory prediction step that extrapolates from the agent's current location, goals, and state into a full trajectory. This work emerged from the idea of social pooling using long short-term memory (LSTM) networks used to predict trajectories of humans in dense environments [52]. This work was improved toward producing numerous possible predictions using generative adversarial networks (GANs) to generate socially plausible trajectories [53]. This area has been further improved by using Info-GANs (information maximizing GANs), to preserve the multi-modality of predicted trajectories, and conditioning the attention pooling on features from neuroscience and biomechanics [54]. The convergence rate and real-time deployment of these approaches have been improved upon by using variational recurrent neural networks [55]. Highly effective results have been achieved using a graph-structured recurrent model based on conditional variational autoencoders (VAE) [56]. Other work has used VAEs to model the distribution of predicted future goals and then predict the many paths taken to solve them in a two-phase approach [57].

Our focus in this paper is learning methods related to Reinforcement Learning (RL), and Multi-agent Reinforcement Learning (MARL) in particular. Reinforcement learning maps particularly well to the crowd simulation problem [17], [19]. This approach has been used to learn the above trajectory problem as well [58]. In RL, the task is to sample the state space of a problem and learn policies for producing corresponding actions. That is, RL's outcome is a Q-table or function which maximizes the value of state-action pairs in the form of policies. Similarly, agent-based synthetic crowd steering models sample the local state at every model update and produce some form of action. Often this is a rule-based approach that integrates some form of a mathematical model of the agent and its interactions with other agents and the environment. The task in RL-based crowd simulation then is to learn this model. A Vector Quantization approach has been used to successfully model multi-agent pedestrian navigation [59]. Some related preliminary work has been shown to produce small groups of navigating agents successfully [60], [61]. A particularly well-evaluated set of models was used to show that this problem is tractable and scalable [16], [18]. However, this approach requires scenario-specific training to converge on expected emergent behaviours. Most related to our work is the recent agent-based RL model learning continuous actions in a curriculum manner [15]. Recently, another LSTM-based method using Deep Reinforcement Learning proposed first learning leader-based control in one phase and using *RVO* to handle collisions in a second phase [14]. Similarly, RL has been used to learn simple swarming behaviour from leader-based control [13]. While these methods have taken many approaches to crowd simulations they are all limited in generalization and evaluation. Additionally, it is not clear whether trained models can be reused or reparameterized in scenarios not present in the training simulations.

2.4 Evaluation and Performance

The evaluation of crowds and steering models is a generally ill-posed problem. Data-driven evaluation suffers from the need to recreate real-world data or to find a means of measuring the difference between real and simulated scenarios. Metric-based solutions reflect the intricacies of their chosen metrics and may not have meaning over different scenarios or may require normalization. Normalization introduces the issues of what reference to normalize with respect to. Early comparative crowd evaluation methods noted the importance of a dense hierarchical approach to evaluation. That is, by formulating a set of benchmarks from the lowest level of steering behaviour performance (reaching waypoints) to the highest (collision avoidance and emergent behaviours in sophisticated crowded scenarios) [25]. We use several similar environments and build out an evaluation of crowd dynamics in a similar manner. Similarly, the scenario test set can be constrained to one definition, such as room evacuation to compare across algorithms in a controlled environment [10]. This approach has been extended to sample over densities to evaluate the changing performance found at different service levels in egress or pedestrian hallway scenarios [24], [62]. We define similar environments and procedures in our evaluation set to test for expected outcomes related to dense environments.

Metrics can be directly computed which give some sense of performance within a particular scenario, such as collisions [63]. This set of measures can be expanded to include metrics that are computed along the path integral of the agents throughout the simulation [64]. This provides an in-depth look at performance but outcomes can only be compared within scenarios.

When real- or ground-truth data does exist we can deploy data-driven methods to evaluate algorithms. One approach to this is to recreate the initial conditions of the real data and measure the divergence between the simulation and real data over time [65]. These approaches require highly accurate re-enactment and assumes that crowds evolve deterministically from initial conditions. One way to address this is to instead model the statistical error distribution between the real data and the simulation. Principal Component Analysis (PCA) can also be applied to the sources of variability in metrics and trajectories to closely examines the components from each data source that contribute to the variability [66]. This facilitates close evaluation of the differences between real and simulated data when the results may be too noisy and/or too high dimensional.

Normalization of evaluation is important for comparing models of different dimensionality but can be difficult to apply to very different types of models (difficult to apply to learning methods and rule-based at the same time). For

example, outcomes can be measured against their optimal (or best possible value) assuming one can model the optimal accurately (such as in crowds the shortest path can be considered perfectly optimal) [67], [68]. Assuming reference data exists, crowd model parameters can be optimized to best fit reference data prior to performing evaluations (including direct metric comparisons) [69], [70].

Finally, it is possible to also evaluate crowds simulators from a perceptual approach. For example, an experiment can be devised to elicit responses to different crowd simulators and their impact on presence [71]. Most recently, there are methods that blend approaches and devise perceptually validated metrics that capture the salient features impacting the perception of realism trajectories produced in simulation [72].

2.5 Comparison to prior work

Our method differs from these prior works (specifically the most related [15], [16], [18], [40]) in several ways. Our proposed method is a model-free multi-agent reinforcement learning using parameter sharing, with additional goal-conditioned control parameters and state observation stacking. The model-free component means our method can learn from trial-and-error with an underlying ground truth model of the environment. We specifically use parameter sharing to mitigate the non-stationary problem of multi-agent reinforcement learning and massively improve the gathering of experience during training [73]. This can be seen as a form of centralization, however, we do not centralize internal agent information, control signals, or facilitate communication between agents. By goal-conditioning control parameters and rewarding adherence to their expected outcomes we can learn a policy that is then parametric on deployment, i.e. practitioners can set the desired control parameters of individual agents despite the sharing of the same duplicated policy among them. This parametric learning approach is somewhat related to conditioning a control policy on a low dimensional latent variable to select a family of policies [74]. Finally, by using state observation stacking on a compact state observation vector, we sidestep the computational requirements and difficulties with vanishing or exploding gradients found in recurrent neural networks (RNN) and long short term memory (LSTM) approaches for temporal dynamic behaviour that requires observations of higher-order movement derivatives.

3 Heterogeneous Parametric-RL Model

In this section, we cover the components involved in defining and solving the proposed Multi-agent Reinforcement Learning problem. For this work, we represent agents with a standard particle-based model which consists of three components: $\{\mathbf{a}_p, a_r, \mathbf{a}_f\}$, where $\mathbf{a}_p$ is the position, a_r is the radius, and $\mathbf{a}_f$ is the present heading of the agent.

3.1 Reinforcement Learning

In reinforcement learning, we model the learning problem as finding a policy for a Markov Decision Process (MDP). The MDP is defined as a tuple $\{\mathcal{S}, \mathcal{A}, P, R, \gamma\}$, where $S \in \mathbb{R}^n, A \in \mathbb{R}^m$ are the state space and action space in the environment, the transition probability function $P : S \times S \times A \to [0, 1)$ is a probability density function, with $P(s_{t+1} \mid s_t, a_t)$ being the probability density of s_{t+1} given that action a_t is executed in state s_t. The reward function $R : S \times A \to \mathbb{R}$ gives a scalar reward for each transition in the environment. $\gamma \in [0, 1]$ is the discount factor that determines the planning look-ahead. The reinforcement learning objective is to find a policy $\pi(a|s, \theta)$, with parameters θ, that maximizes the following metric:

$$\max_{\theta} J_\pi(\theta) = E_{a_t \sim \pi(\cdot|s_t)} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right] \tag{1}$$

This essentially means we wish to find a mapping between input state and action pairs that maximizes the expected discounted return, or the discounted future rewards. The extension of the basic RL problem to multiple agents can be thought of as a Markov game, or multiple interacting MDPs. While our problem is multi-agent, we make use of parameter sharing to reduce the overall parameters we need to learn for our policy [73]. Policy gradient algorithms are a popular approach to solve this problem, where $\nabla_\theta J_\pi$ is estimated using on-policy samples, i.e., using data collected from the current stochastic policy [75]. Specifically, we use the Proximal Policy Optimization (PPO) algorithm to optimize the RL objective [76]. PPO is a model-free policy optimization algorithm that trains a stochastic policy in an on-policy fashion. That is, it samples actions based on the latest version of its stochastic policy. On-policy methods like these are more suited to our problem because it is generally less expensive to collect data and they generally require less tuning than off-policy methods. PPO aims to make the changes between policies during learning updates big enough to efficiently explore but small enough that the policy does not drastically change between updates. The implementation used in this paper uses a clipped objective to reduce the incentive for the policy to drastically change between update steps, i.e., a type of regularizer which can help in the learning process [77]. Because PPO is model-free, a policy can be learned from a trial-and-error approach without relying on an underlying model, i.e. we do not need a ground-truth model of the environment in order to learn a policy successfully.

3.2 Network Architecture & Parametric Policy Learning

A key contribution of this paper is the learning of parametric policy space to support user control and agent heterogeneity within a single learned policy space. A ubiquitous parameter is required to accomplish this. In other words, a parameter we use must be a part of the state, reward, and either the action or representation of the agent in the environment. In this way, the parameter can be used to select subspaces within the learned policy space. We hypothesize that this approach may be used to learn any other fixed agent parameters, such as walking style or personal space, that a user may wish to parameterize. We make four assumptions about the parameters we wish to learn and how they may be included in the RL problem definition such that the parameters are "ubiquitous" within the defined architecture.

First, we include the parameter(s) ζ we wish to use for selecting policy subspaces within the state observations of each agent, such that $\pi(a|s,\theta)$ becomes $\pi(a|s;\zeta,\theta)$. That is, we use a goal-conditioned policy approach to select behaviours in the form of policy subspaces. In this paper, we use desired speed and later agent personal space to create agents that are heterogeneous with respect to speed and movement policies. For parametric policy learning, the value of this state is sampled uniformly random (each value has an equal chance of being sampled) from the parameter range, $\zeta \sim \mathcal{U}(\zeta_{low}, \zeta_{hi})$, where ζ_{low}, ζ_{hi} are the bounds of the components of ζ. In this way, we guide the learning process to explore the policy subspaces. Here we rely on the ability of deep neural networks to learn or embed the *in-between* values which are not sampled directly during training. In Section 3.3, we describe the agent state in detail.

Second, the parameter, in this form of learning, necessarily impacts the policies (i.e., we wish to get different behaviours from selected parameter values), so it must relate to the action space. However, this relation may be indirect. In this paper, the parameter has some bearing on the action space since velocity is a term in the anti-derivative (momentum) of the control signal (force applied to the particle) and a constraint (maximum speed of the particle).

Third, the exploration of the policy subspaces is only possible if we appropriately reward the value of the state-action pairs. That is, the learning process in RL is guided by reward, and further parametric policy learning is guided by rewarding how well we achieve the desired action given the current parameter setting in the state. This is simplified (i.e. we reduce variances in the learning) by fixing the parameter (fixed value in the state observations) over the lifetime of the agent and making components of the rewards with respect to the value.

Finally, to support parametric policy learning in heterogeneous environments, agents of particular parameter settings must be exposed to agents of other parameter settings within the same learning environment. In this way, the individual subspaces of the larger policy encapsulate the entire policy of a particular type of agent. In this paper, the agent learns not only to fulfill its speed constraints but also policies for interacting with other diverse agents.

The network model is 2 fully connected hidden layers of 512 hidden nodes each using the Swish activation function, which has been shown to improve deep neural network performance over a range of tasks [78], [79]. The network model can be seen in Figure 2 with all the components of the environment, state, and action. The following sections describe the details of the involved components individually.

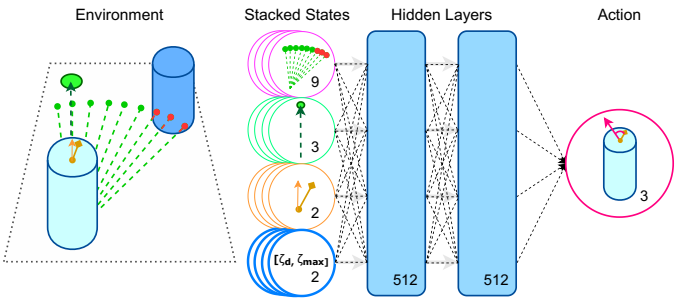


Fig. 2: The deep neural network is composed of 2 hidden layers of 512 hidden nodes each using the Swish activation function [79]. The environment observations are concatenated with a parameter ζ for learning parametric policies spaces using goal conditioning and stacked to mitigate the partial observability problem in multi-agent reinforcement learning. The action changes the movement and orientation of the agent. Each component is described in detail in State: Section 3.3 and Actions: Section 3.4

3.3 State Space

The state-space of the agent includes information required to take actions that resolve the two primary goals of steering – goal-seeking and collision avoidance. In this section, we outline the components of a single state observation and then extend this into several stacked snapshots of past observations motivated by a discussion of learning parametric, heterogeneous agents – a key contribution of this paper. The agent's current goal in the state is represented by a distance g_d normalized by the diagonal of the virtual maximum ground size, and by its direction $\mathbf{g}_p$ relative to the agent's forward $\mathbf{a}_f$. The agent also has a form of vision as a series of depth testing rays $\phi_{1..9} \in \mathbf{\Phi}$ evenly spaced within a $90°$ field, each normalized by its maximum vision distance ϕ_d. For this work, we found through testing, that using 9 rays provided adequate local information for steering. We postulate that using an odd number such that there is a central forward-facing ray is important for collision avoidance as it aligns with the agent forward vector $\mathbf{a}_f$.

To support the parametric and heterogeneous requirements of the agents, two factors regarding the state are required. The first is that the parameter to be used is a part of the state, such that the agent has information about the parameter's value. That is, the state includes, in this particular instantiation, the desired speed v_d as well as the maximum allowed speed v_{max} which is randomly selected during training from $[v_{min}, v_{max}]$ at initialization and remains fixed over the agent's lifetime. In this way, the approach is a goal-conditioned policy that selects a subspace for the desired behaviour. The agent's forward relative velocity state, $\mathbf{a}_\Delta$, is the normalized relative vector between the agent's current movement direction and its forward vector.

Additionally, in this particular instantiation, we note that a single state can not capture velocity information of neighbouring agents and thus will not be capable of learning policies for heterogeneous configurations. Prior work has encapsulated this information as part of the state [80]. However, this requires that agents are aware of other agent's internal parameters. We want to avoid the overhead and limitations associated with centralizing information, control, or communication (i.e. agents should remain individual and autonomous) and for agents to *learn* to "perceive" this information. This is made partially possible by the depth testing portion of the state, but this requires some form of additional memory to observe the change in state over time, e.g. changes in depth are a perception of relative velocity.

One approach to this problem is to expand the input space with multiple sequential observations of the state,

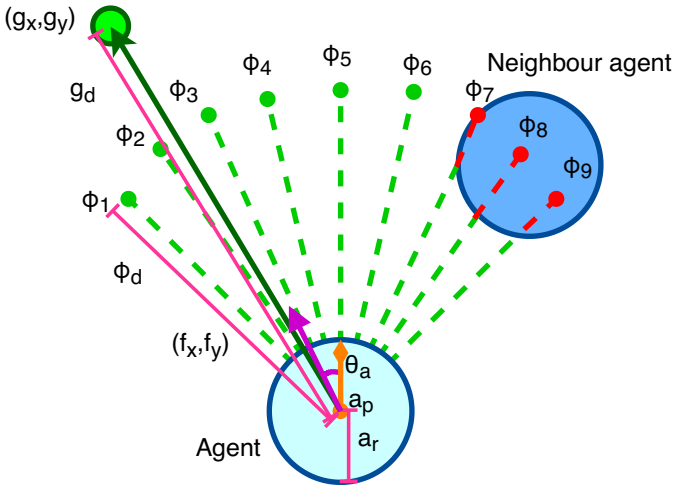


Fig. 3: The agent is a particle model represented by a (blue) circle defined by its current position $\mathbf{a}_p$, forward vector (orange), and radius a_r. The state includes depth test rays, representing vision, shown in green ($\mathbf{\Phi}$). These are defined by their count, start/stop angles (field sweep), and maximum test distance ϕ_d. In this definition, we use 9 rays $\phi_{1..9}$ within a 90° field–using an odd number ensures there is always a central ray emitting from the forward vector of the agent. Additionally, the state includes the current distance to the goal g_d and the relative goal position $\mathbf{g}_p = (g_x, g_y)$. Details for the state are covered in Section 3.3. The action space is represented in fuchsia and includes the angle to rotate θ_a and the vector representing the force to apply $\mathbf{f}_a = (f_x, f_y)$. Details for the action space are covered in Section 3.4.

or *observation stacking*. In this context, observation stacking means expanding the number of input nodes of the network to accommodate each stacked observation. That is, if a single state observation is $\mathcal{S} = \{g_d, \mathbf{g}_p, \mathbf{\Phi}, \mathbf{a}_\Delta, a_s, v_d, v_{max}\}$ where $|\mathcal{S}| = 16$ when the vector components are expanded, then N stacked observations $\mathcal{S}_N$ amounts to $|\mathcal{S}_N| = 16 \cdot N$ input nodes in the network. In this way the state includes the current observation and the previous $N-1$ observations. At initialization, prior observations are zeroes. This method is evaluated in Section 4.1. A visual representation of a single state observation can be seen in Figure 3.

3.4 Action Space

We define a three-dimensional action space. The first dimension represents the normalized rotation angle θ_a to apply to the forward vector $\mathbf{a}_f$ of the agent particle by the maximum allowed angular speed (desired angular speed) θ_d. The second and third dimensions of the action space are the two components of the driving force (f_x, f_y) of the force vector $\mathbf{f}_a$ to apply to the agent particle. The resultant velocity is clamped by the desired speed. In this way, the model is essentially learning to turn and to move using the final net force vector which is to be applied to common force-based approaches in steering [9], [10].

3.5 Reward Function

Our proposed reward function values movement towards the goal and anticipatory collision avoidance while penalizing collisions. Both location-based rewards and collision-based penalties have predictive and instantaneous counterparts to also encourage predictive, or anticipatory, behaviours.

We define a predictive and cumulative location reward r_l designed to be conservative over all paths. That is, we aim to value reduced effort paths of different modalities in our reward. In this way, the location reward forms an accumulation of predictive location-based rewards as a potential field $r_\Sigma = |\mathbf{g}|^\beta$, where $\mathbf{g}$ represents the agent-to-destination vector. In each step, r_l is calculated by taking the gradient of r_Σ projected onto a vector of small step Δt shown as:

$$r_l = -\nabla\left(r_\Sigma\right) \cdot \Delta\mathbf{d} = -\beta\left|\mathbf{g}\right|^{\beta-1} \hat{\mathbf{g}} \cdot \mathbf{v}\Delta t \tag{2}$$

where $\Delta\mathbf{d}$ represents the distance travelled in the current time frame, $\mathbf{v}$ represents current velocity, and Δt the time interval associated with the current frame. We found $\beta = -0.5$ produces desired results but can be tuned for more or less direct goal-reaching behaviour.

We define an instantaneous location-based reward r_d which only triggers when an agent reaches their target. We found $r_d = 2$ sufficiently incentivizes goal-reaching.

A predictive collision penalty scales the final reward with respect to incoming collisions, defined as:

$$p_t = \prod_{\forall a' \in \mathcal{N}(a)} \tanh T\left(a, a'\right) \tag{3}$$

where $(\mathcal{N}(a))$ is the set of neighbouring agents within the radius n_r. $\mathrm{T}(a, a')$ is a function that predicts the time left before two agents collide under their current velocities:

$$\mathrm{T}(a, a') = \max(\arg\min_t(\|a_p - a_{p'} + t(v_a - v_{a'})\| = a_r + a_{r'}), 0) \tag{4}$$

where a_p is the agent position, a_r is the agent radius, and the prime values ($\prime$) are for the other agent. In this formulation, t may have more than one solution, so we solve this equation for the minimum t–the closest ray-disk intersection test [9]. Dependent on Equation 4, if the time left to collide is 3 seconds or more, then $p_t \approx 1$, i.e., there is no predictive collision penalty because p_t is used as a scaling factor for the continuous target reward in Equation 2. That is, if $\tanh T\left(a, a'\right) < 1$ (from Equation 3) then the agent is going to be penalized for a potential incoming collision. This predictive collision check is derived from [9] and converted into a penalty ([0,1]) scaling the contribution of the relevant goal reaching reward.

An instantaneous collision penalty p_c is applied on simulation steps when an agent either collides with another agent or with a wall. We found $p_c = -0.01$ sufficiently disincentivizes collisions.

Together the total reward function becomes the summation of all rewards and penalties with a weight factor ω_l for r_l to adjust its contribution to the finalized reward:

$$r_{all} = \omega_l r_l p_t + r_d + p_c \tag{5}$$

We found $\omega_l = 3$ balanced desired goal-seeking behaviours with collision avoidance learning. This final reward signal values ideal agent movement which is anticipatory, reduces effort in steering (takes short paths), and avoids collisions. In our multiplicative reward, anticipatory collision avoidance (p_t) and goal-seeking (r_l) is weighted (ω_l is for behavioural control and can be tuned by someone retraining the model) and scaled by their severity and likelihood ($\omega_l r_l p_t$) while discrete target rewards only occur when and an agent completes their global task(r_d) and collisions are heavily and consistently penalized at each step. In practice, p_t reduces the reward gained by r_l to penalize potential collisions more as they become more likely.

3.6 Training

We summarize our agents as the state space $\mathcal{S} = \{g_d, \mathbf{g}_p, \mathbf{\Phi}, \mathbf{a}_\Delta, a_s, v_d, v_{max}\}$ and action space $\mathcal{A} = \{\theta_a, \mathbf{f}_a\}$. All agents learn and share the same network parameters but have different randomized goals in randomized environments as described in the following sections.

3.6.1 Environment and Scenarios

We utilize domain randomization for our training environments. The training environment is defined by a grid in a square region (20m x 20m) in which obstacles of size (1m x 1m) may be randomly placed at the center of each grid cell with a probability of 10%. There are 100 agents spawned in each training iteration. However, purely random agent, goal, and obstacle spawning may lead to enclosed, i.e. unreachable, agents or goals. To deal with this during training we ensure that there is a free path between agent and goal when spawning, if not we reattempt at new locations until there is. The environment definition is a high-density procedural scenario and is intended to force agent-centric encounters between various environment and agent configurations, aimed to find more general policies [22]. This environment can be seen in Figure 4. That is, the training environment is intended to have coverage over the space of possible agent interactions and be sufficiently noisy to produce generalizations in the learned policies. Our evaluations in Section 4 show that this produces a model that generalizes to various environment configurations, agent configurations, and the number of agents.

3.6.2 Simulation

After obstacles are placed on the ground, stationary agents are randomly spawned, with random initial headings. Each agent is assigned a single randomly positioned static world target and there are no overlaps among obstacles, agent spawn locations, or agent targets. This formulates the initial conditions of our domain randomization approach during training. Each agent computes their shortest path to their randomized world target using the A* pathfinding algorithm on a NavMesh representation of the environment [81]. An Agents long-term path is updated every 2 seconds or when they lose sight of the current waypoint (the long-term path is composed of a series of waypoints on the NavMesh). Agents will initially receive the topmost waypoint in the queue as their current local target for steering and will iteratively switch to the next waypoint in the queue until the next waypoint can not be seen (obstructed by obstacles in the environment), i.e. simplified string-pulling. Once the agent reaches its final world target, it will be removed from the environment. We make use of long-term path planning because we performed several tests without the addition of explicit long-term path planning and agents were not able to handle complex planning in environments, particularly those with concavities. This is a known effect of steering/planning separation in the literature, and this paper focuses on the steering model layer. We hypothesize that it is possible to learn this planning layer policy and recent works have succeeded in doing so already [82], [83], [84]. However, this is outside the scope of this paper.

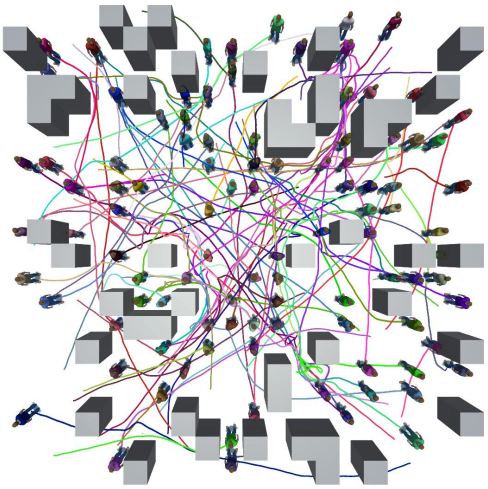


Fig. 4: Agents learn to continuously avoid each other as well as obstacles in arbitrarily complex environments. This figure is a single example of the procedural training environment.

3.6.3 Learning

Our deep neural network is composed of 2 hidden layers of 512 hidden nodes, or artificial neurons, each using the Swish activation function [79]. We train this network using Proximal Policy Optimization (PPO) with the Adam optimizer to update network weights [85]. Relevant hyperparameters include, $\gamma = 0.99$ (the reward decay factor in the expected discounted return seen in Equation 1), $\beta = 0.005$ (the weight of entropy regularization dictating policy randomness), $\epsilon = 0.2$ (the threshold of divergence in policy updates, used in the clipped objective for PPO [77]), $\lambda = 0.95$ (the reliance on current value estimates in the Generalized Advantage Estimate calculation [86]) with a learning rate of 0.0003 (the weight of gradient descent update steps), a batch size of 512 (the number of experiences to use in a gradient update step), and a buffer size of 20480 (the number of experiences to use in a model update). Each episode of simulation in the training is triggered either when all agents complete their goals or $10,000$ training steps have been reached. The reward update period is 0.02s, or 50Hz. Given our γ value and update frequency, our agents are learning to plan approximately 2s ahead.

4 Evaluation

We constructively evaluate our contributions in a series of experiments. We note first, as in Section 2.4, that the analysis of steering algorithms, and comparative crowds analysis in general, is an ill-posed problem. We take the approach of incrementally evaluating portions of the proposed approach, from several perspectives, with respect to metrics and scenarios from the literature.

First, we cover learning performance and features found in training. We focus on performance during training (from the perspective of reward) and the quantitative performance of our proposed state observation stacking methodology. Second, we cover several common qualitative benchmarks to show our model produces desired steering behaviours. Third, we cover the quantitative performance of general multi-agent navigation. We use metrics from the comparative crowds analysis literature to understand where the method is performant with respect to traditional methods. Fourth, we verify our behaviour conditioned model on heterogeneous scenarios and show that heterogeneous crowds diverge from homogeneous crowds in our outcomes. Fifth, we cover density-dependent artifacts to verify the model produces expected outcomes in a common multi-agent scenario from the literature. Sixth, we cover the computational performance of the method with respect to agent count. We focus on giving a general idea of how performant our model is with respect to real-time applications. Finally, we repeat training on an additional parameter to show that the method is extensible to other parameters a practitioner may wish to use.

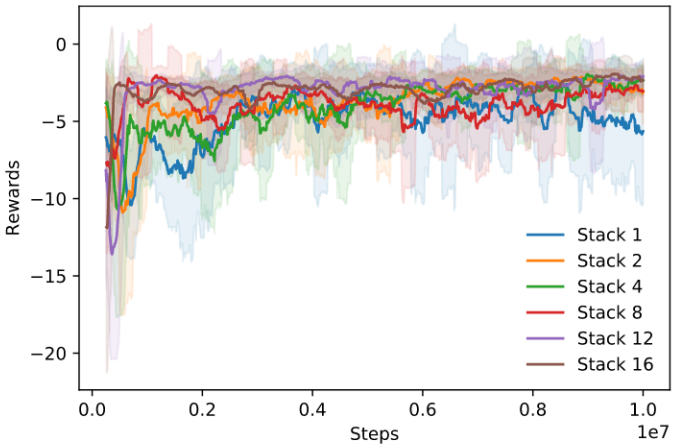


Fig. 5: The training portion of the observation stacking experiment investigates the improvement in memory-dependent behaviour conditioned learning. We increase the number of observation vectors in the stack and explore the mean reward overtraining. The stacking provides a form of crude memory, for handling the parameterization of velocity, a time-dependent factor between agents which the model is not given directly. The 4 level observation stacking model is selected based on these training results, quantitative (see Section 4.1 & Figure 6), and qualitative results (see Section 4.2). Each model is trained for 10 million iterations. Because training curves are similar for stacks and can only provide limited information, we perform additional follow-up studies on the performance of the learned policies, see Section 4.1.1 and Figure 6.

4.1 Learning Reciprocal Collision Avoidance in Heterogeneous Environments

In this section, we evaluate the learning performance and the contribution of our proposed stacking methodology for state observations. We hypothesize that observation stacking affords the learning of velocity-aware policies that resolve reciprocal collision avoidance in heterogeneous environments as described in Section 3.3. We further hypothesize that there is a critical value after which the depth of the stack has diminishing returns. That is, stacking in this model is used to increase performance for policies handling time-variant heterogeneous agent features, such as velocity, acceleration, and jerk. We adjust the number of stacked observation state vectors over several training experiments to find the critical point at which stacking no longer produces useful returns. We sample six levels of observation stacking 1, 2, 4, 8, 12, 16, or values for N in $\mathcal{S}_N$. To choose a final stacking level we observe the training performance, as well as, several multi-agent navigation-specific metrics. These metrics are six separate path integrals, summed over all agents, which measure key aspects of collision and steering performance: unique collisions count, collision magnitude, completion time, acceleration, distance travelled, and kinetic energy. For this latter study, each stacking level is evaluated over 25 simulations of the training environment with random initial conditions for the obstacles and 20 agents.

In addition to a thorough stacking experiment, we also conduct a simple experiment on out-of-distribution parameters (desired speed outside the trained range). In particular, we sample at random across all agents in the ranges of $\{4, 6, 8\}$m/s. In typical Deep Reinforcement Learning applications, the action space is clamped to ensure that unexpected values still fall within the expected range. We conduct this experiment twice, first where we limit the action outputs to the max speed in the originally trained range (i.e., we limit within distribution) and second, where there is no limit on the output action. We conduct these experiments using the final network with the chosen stacking level (level 4, see Results & Discussion below)

4.1.1 *Results & Discussion*

The training performance in terms of mean reward during training iterations, of these stacking levels, can be seen in Figure 5. This data is too noisy and without a clear and obvious candidate for selection, so we must pair it with quantitative results on the quality of the policy to understand the impact of stacking state observations. The results for quantitative policy quality are shown in Figure 6: (a) unique collisions count, (b) collision magnitude, (c) completion time, (d) acceleration, (e) distance travelled, and (f) kinetic energy.

The evaluation of stacking levels with respect to learning performance over training rewards and steering policy quality measures show clearly that the 4 level observation stack produced the best results. This stacking level minimizes collision counts and collision magnitudes while performing on par and at acceptable levels with all other metrics. The rest of the paper uses this model for evaluation.

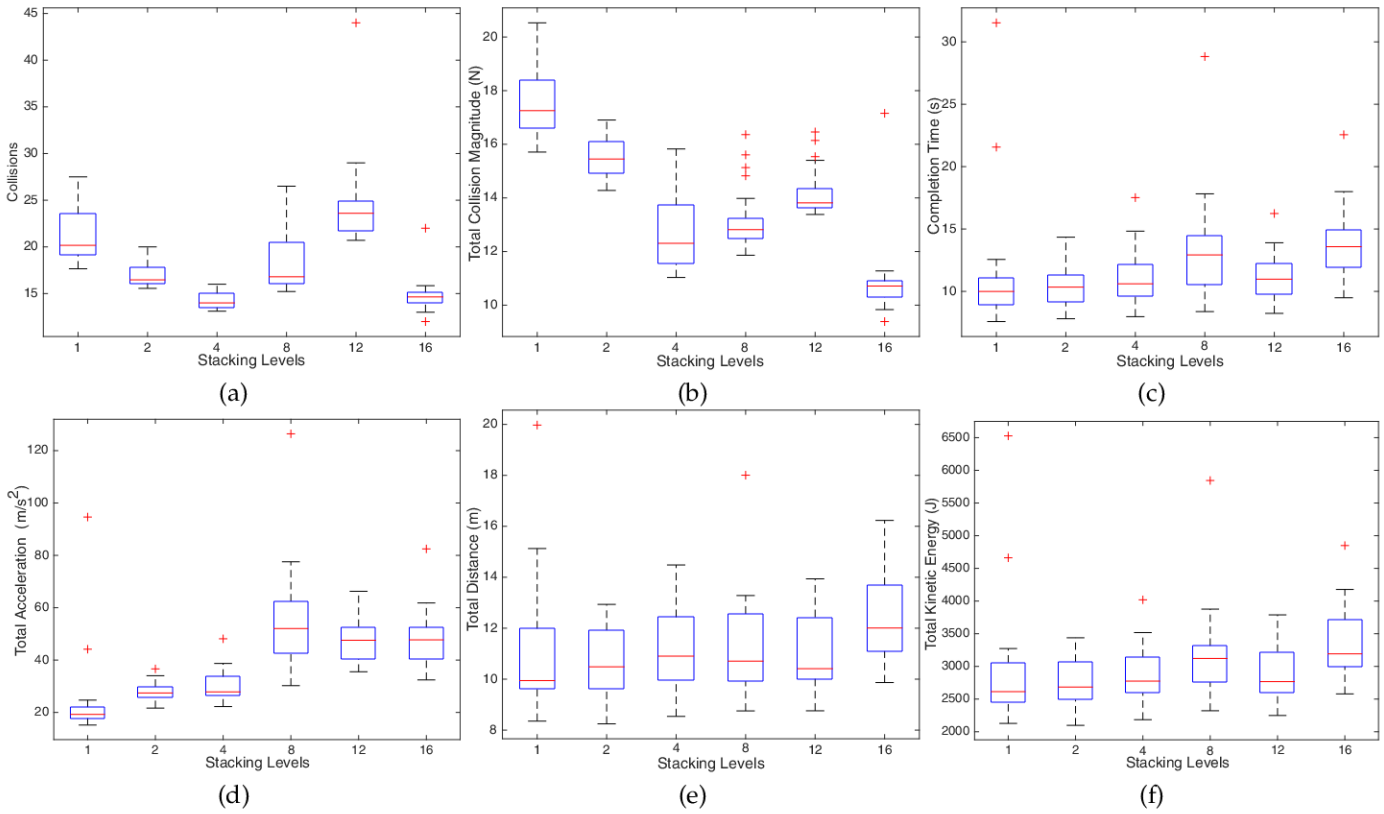


Fig. 6: (a) Collisions, (b) Collision magnitudes, (c) Completion times, (d) Total accelerations, (e) Total distance travelled, and (f) Total kinetic energy in the observation stacking experiment. Each stacking level's {1,2,4,7,12,16} performance is represented by a box plot illustrative summary statistics where the median is the red line, the bottom and top edges of the box (blue) indicate the 25th and 75th percentiles, respectively, while the whiskers cover the range of all values except outliers which are shown individually using the '+' (red) symbol. Several stacking levels are compared across several common crowds metrics. We find that a stacking level of depth 4 outperforms or is on par across all metrics.

Our out-of-distribution parameter experiment revealed some interesting qualitative behaviours. In both the limited and unlimited max speed tests, agents can still navigate to their final goals. In the limited scenario, agents produce inefficient paths, often seemingly confused with how to handle agent-obstacle collision avoidance. This is expected as the desired speed and max speed outcome do not align. That is, the policy has learned the distribution of speeds needed to meet the set input parameter. This is further shown in the unlimited max speed test. In this test, it is clear the learned policy has generalized to unseen desired speed values, as the agents move faster than they were trained on and approximate the out-of-distribution parameter. However, their ability to navigate is hindered by apparent undamped steering, that is, the agents often overshoot their navigation goal even though they eventually finish.

4.2 Qualitative Multi-Agent Navigation Performance

In this and following sections, *HOP-RL* is analyzed with respect to common steering models of similar capability. That is, each model attempts to address the reciprocal predictive collision avoidance problem, has similar inputs or state space, and is specifically a steering model. Model-free approaches must be compared against established crowd simulation techniques in a fair and unbiased manner before they can be truly used in practical contexts. Each model in the comparative analyses uses the same A* long-term path planner with simplified string-pulling described in Section 3.6.2. These models are:

- *HOP-RL*– a model-free RL-based approach with parametric policy learning using observation stacking.
- Reciprocal Velocity Obstacles (RVO) – a computational geometry-based approach that optimizes collision-free actions in velocity space [8].
- Predictive Avoidance Model (PAM) – a physics-based approach that predicts linear collisions and applies appropriate avoidance forces [9].

We show several examples of classic scenario benchmarks in crowd evaluations where each scenario has expected qualitative outcomes. We evaluate over scenarios of increasing difficulty in a similar fashion to [25]. In particular, these benchmarks are intended to exhibit specific controlled behaviours making up the components of a successful steering policy.

Agents must be able to navigate towards goals while avoiding collisions with each other. In the crowd simulation literature, this is achieved in either a reactionary way (adequate) or an anticipatory way (desirable). We show several examples of agents successfully learning to complete this task in an anticipatory manner in Figure 7. *HOP-RL*

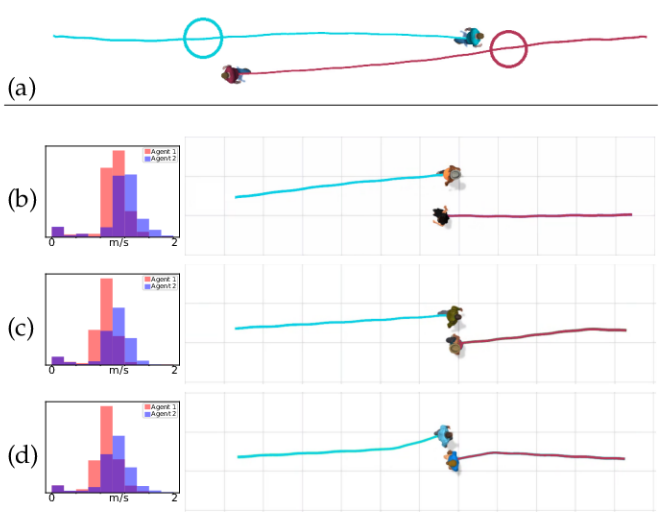


Fig. 7: Agents learn a high value predictive reciprocal collision avoidance behaviour. In (a), predictive avoidance can be seen in the early movements away from the collision path. This point is highlighted with circles of the agent colour. In (b-d) we qualitatively compare *HOP-RL*, *PAM*, and *RVO* respectively. We include the distribution of agent speeds (red right, blue left)–each figure is over the domain [0.0, 2.2]m/s. Distribution over the scenario and across methods is similar. *HOP-RL* agents learn to anticipate and smoothly avoid collisions early. *PAM* agents have sudden changes in trajectory when the predictive force becomes stronger. *RVO* agents produce reciprocal collision avoidance only when the agents are near and the trajectory has a mechanical quality.

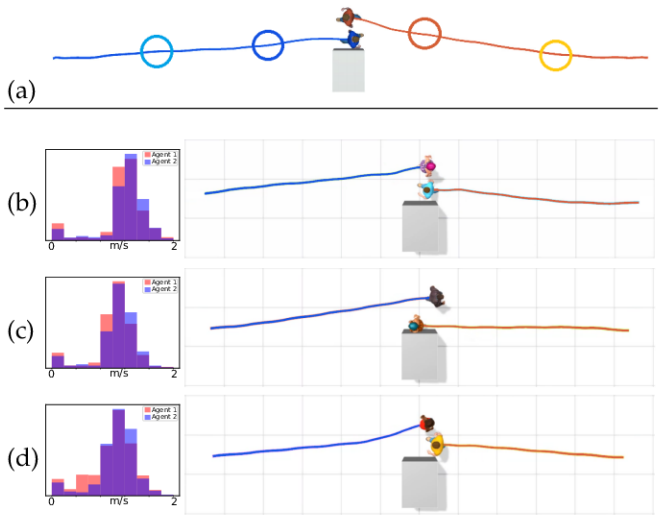


Fig. 8: Agents learn to avoid collision with each other and with simple obstacles. The point predictive obstacle collision avoidance learned by *HOP-RL* is highlighted in (a) a circle of the lighter agent colour, and the point reciprocal collision avoidance begins is highlighted with the circle of the agent colour. In (b-d) we compare *HOP-RL*, *PAM*, & *RVO* respectively. We include the distribution of agent speeds (red right, blue left)–each figure is over the domain [0.0, 2.2]m/s. Distribution over the scenario and across methods is similar. *HOP-RL* agents learn to anticipate and avoid the obstacle and each other early. *PAM* agents may violate obstacle boundaries when collision avoidance and predictive forces coincide and overcome obstacle forces. *RVO* agents mostly slow down to avoid the conflicting paths.

agents learn to anticipate and smoothly avoid collisions early. *PAM* agents have sudden changes in trajectory as the predictive force becomes stronger. *RVO* agents produce reciprocal collision avoidance only when the agents are near and the reciprocity is exact, giving the trajectory a mechanical quality.

Agents must be able to avoid obstacles and each other while navigating towards their goals. We show this in multiple examples of increasing difficulty. First, in Figure 8, we show agents avoiding a single obstacle. Second, as shown in Figure 4, agents continuously avoid each other and obstacles in a complex environment. The latter is an example instantiation of the training environment. *HOP-RL* agents learn to anticipate and avoid obstacles and each other early. *PAM* agents may violate obstacle boundaries when collision avoidance and predictive forces coincide and overcome obstacle forces. *RVO* agents mostly slow down to avoid the conflicting paths.

Synthetic crowd steering models must also produce expected emergent behaviours under certain conditions. First, we show that *HOP-RL* agents produce vortexes in a homogeneous diametric goals scenario–a seminal benchmark for steering models intended to exemplify emergent least effort paths. An example of this can be seen in Figure 9, which shows a diametric goal scenario approximately 20m in diameter at several key points, with both homogeneous and heterogeneous parameterizations across agents. Second, we show that *HOP-RL* produces laminar flow in crossing groups scenarios. An example of this can be seen in Figure 10. Laminar flow is an important qualitative behaviour found in crossing flows of human groups. Finally, we show *HOP-RL* in the classic heterogeneous bottleneck egress scenario in Figure 11 where crowding near the egress point forms because of the agent density. *RVO* tends to produce aggressive overtaking behaviours in this scenario while *HOP-RL* produces smoother more predictable behaviours.

4.3 Quantitative Multi-Agent Navigation Performance

In this section, we evaluate the quantitative quality of our learned steering policy by comparing *HOP-RL* with both the industry baselines and two baseline RL-based approaches over the steering models metrics introduced in Section 4.1. Our baseline RL approaches use the same state-action-reward definition as *HOP-RL* but we remove the parameter learning and train new models with and without stacking state observations.

This experiment analyzes the performance of all five models within the procedural training environment described in Section 3.6.1. For each model, the experimental scenario is randomly initialized 20 times such that obstacle placements, agent initial conditions (placements and desired velocities), and agent goals are randomized. Each simulation involves 100 agents navigating the environment until they reach their final navigation goal.

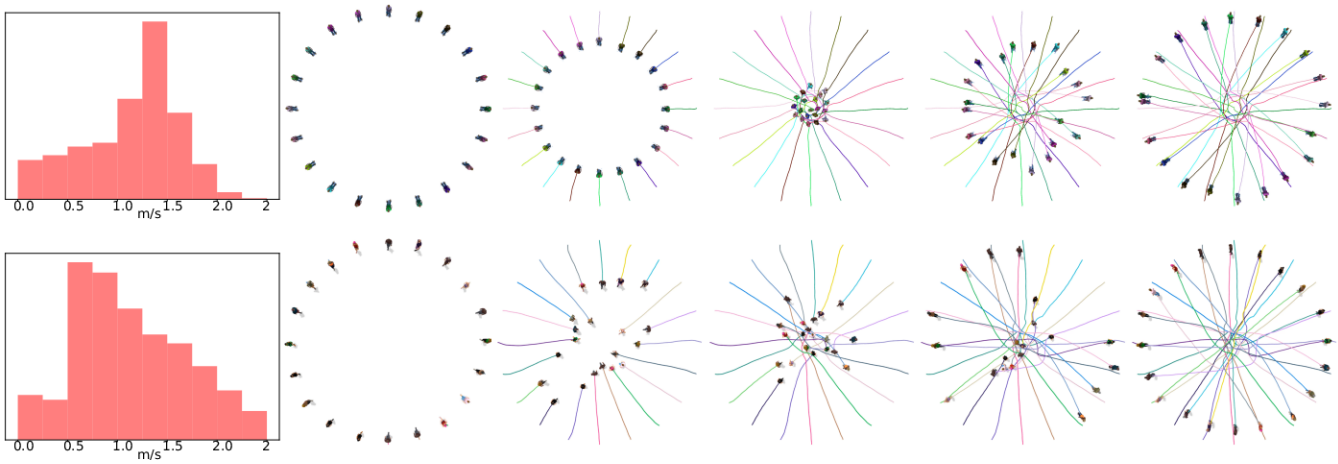


Fig. 9: Frames, ordered from left to right, in the top row show agents learn to continuously avoid each other in a diametric scenario by producing an emergent vortex behaviour. In the bottom row, the same scenario repeated but with randomly distributed desired velocities. We include the distribution of agent speeds–each figure is over the domain [0.0, 2.2]m/s. Both rows are using the same trained policy with no retraining or updating.

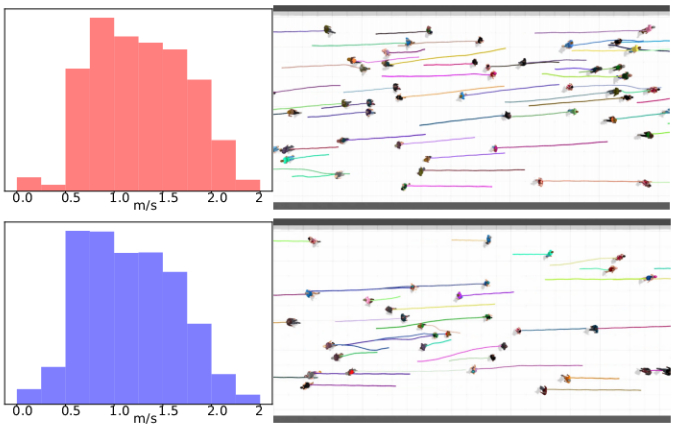


Fig. 10: Agents learn to avoid collision with each other in a crossing groups scenario. *HOP-RL* (top) and *RVO* (bottom) produce lane forming behaviour. However, *RVO* may produce slow down artifacts when agents of very different speeds must steer with respect to each other. We include the distribution of agent speeds–each figure is over the domain [0.0, 2.2]m/s.

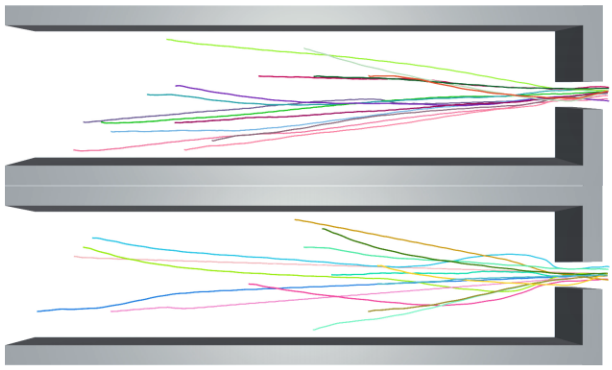


Fig. 11: The model produces smooth movement in heterogeneous bottleneck scenarios and crowding near the bottleneck as density increases. In the top row *HOP-RL*, and in bottom row the *RVO* model can be seen for reference. In this scenario, *RVO* produces aggressive overtaking behaviour which occasionally causes the violation of agent bounds, i.e. passing through agents, while *HOP-RL* produces predictable overtaking and passing behaviours without artifacts.

4.3.1 Results & Discussion

The results for policy quality with respect to baselines are shown in Figure 12. (a) unique collisions count, (b) collision magnitude, (c) completion time, (d) acceleration, (e) distance travelled, and (f) kinetic energy. The quantitative crowd results show that *HOP-RL* performs on par with, or better than, *RVO* in terms of collision count, and on par with *PAM* in terms of collision magnitude and total accelerations. Importantly, *HOP-RL* outperforms other models in terms of completion time and total kinetic energy. *PAM* may on occasion violate agent bounds because of the exponentially increasing forces as agents approach their boundaries. In contrast, *RVO* produces a significantly higher total acceleration than *PAM* or *HOP-RL* due to sudden changes in direction and repeated stopping conditions. We show that the baseline RL methods with and without stacking, both do not perform as well in terms of collisions, completion times, total accelerations, total distance, and total kinetic energy. The lack of performance in completion times, distance, and kinetic energy are mostly due to the homogeneous nature of the agents causing more bottlenecks through equilibrium states particularly in this scenario where density can rapidly rise (all agents have the same movement profile so often they get stuck and have to push around each other). All other models in the experiment are heterogeneously parameterized and naturally break these *ties* in dense movement. Taken together *HOP-RL* is performative with respect to baselines and also produces lower cost, more efficient paths.

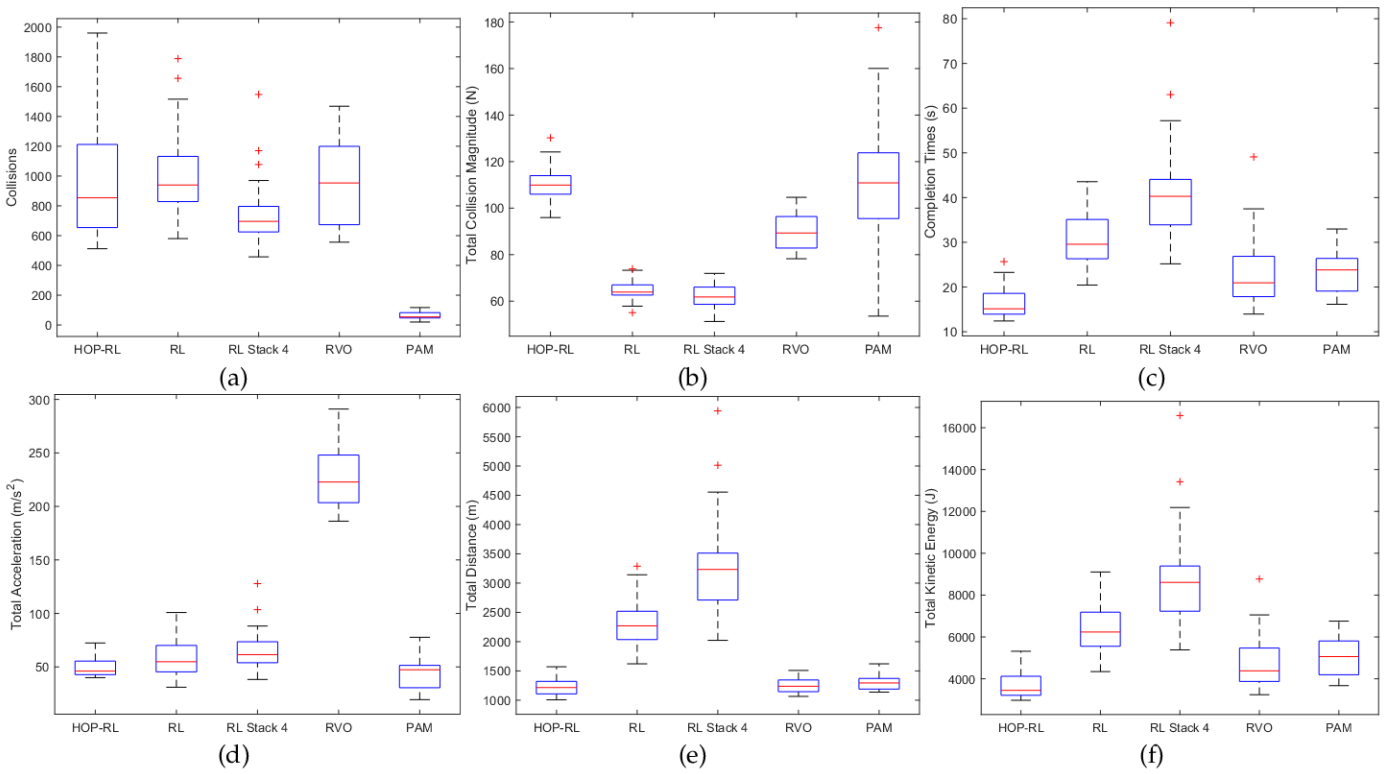


Fig. 12: (a) Collisions, (b) Collision magnitudes, (c) Completion times, (d) Total accelerations, (e) Total distance travelled, and (f) Total kinetic energy in the comparative quantitative experiment. Each model's {*HOP-RL*, RL, RL Stack 4, *RVO*, *PAM* } performance is represented by a box plot illustrating summary statistics where the median is the red line, the bottom and top edges of the box (blue) indicate the 25th and 75th percentiles, respectively, while the whiskers covers the range of all values except outliers which are shown individually using the '+' (red) symbol.

4.4 Heterogeneity Dependent Performance

In the application of synthetic crowds, across many domains, heterogeneity is of utmost importance. This may be visual only heterogeneity via avatar rendering styles, 3D models, accessories, etc. However, in crowd visualization (games, animation, film, VR), the factor in crowd dynamics which most diversifies agents perceptually is velocity and motion profiles [21]. *HOP-RL* learns this form of heterogeneous steering using parametric policy learning via observation stacking. So far we have shown that *HOP-RL* is performant with respect to common crowds measures in heterogeneous environments – successfully producing reciprocal collision avoidance and performing on par with model-based approaches quantitatively and qualitatively. This experiment serves to complete that analysis focusing on levels of heterogeneity within the makeup of the crowd.

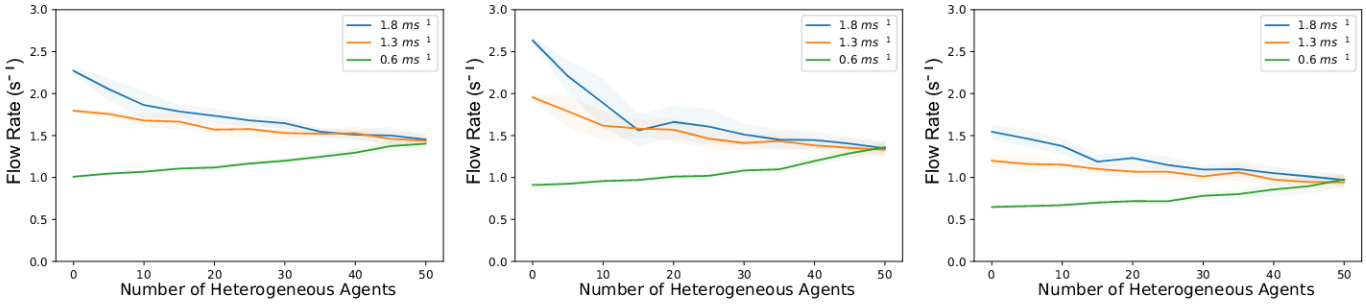
In this experiment, we show the important quantitative impacts of heterogeneity in crowds, specifically that our model successfully produces divergent outcomes between completely homogeneous and completely heterogeneous parameterizations as found in traditional methods. To show this, we sample several mixtures of heterogeneous and homogeneous crowds in a one-way egress environment. Starting with an all homogeneous crowd we incrementally increase the number of heterogeneous agents, with randomly assigned desired speeds $v_d \in [0.5, 1.6]$ m/s. We repeat this for three initial homogeneous speed profiles (0.6, 1.3, and 1.8 m/s) in contrast to the heterogeneous portion of the crowd, measuring the flow rate for each scenario. In this work, flow rate is defined simply as the rate (in agents per second) at which N agents reach their goal, $f(\mathbf{p}) = \frac{N}{t_l}$ where t_l is the completion time of the last agent.

4.4.1 *Results & Discussion*

The zero value on the x-axis of Figure 13 is the flow rate profile of an entirely homogeneous crowd. As the curves move more to the right, the crowd is more and more heterogeneous (x-axis is the number of heterogeneous agents within the crowd). It can be seen that homogeneous crowds of the selected speeds are separate with mean values above their v_d setting. This is expected as steering algorithms strive to meet their desired speed but require divergent movements to handle collision avoidance cases. The deviation from the set homogeneous values is greater with RVO, while PAM closely matches desired speeds at lower values but is under desired speed at higher heterogeneous mixtures. Generally, RVO will select valid speeds (between min. and max. speed values) while PAM is much more conservative due to the anticipatory collision avoidance force. *HOP-RL* is more conservative than RVO but closely matches in performance. At the point of full heterogeneity, all curves for all algorithms converge on an average value.

4.5 Density-Dependent Performance

The literature on pedestrian dynamics, both synthetic and real, has shown density-dependent artifacts with respect to



(a) *HOP-RL* (b) *RVO* (c) *PAM*

Fig. 13: Flow-heterogeneity fundamental diagrams over several heterogeneity mixtures reveal a heterogeneity dependent artifact in the results for each baseline model and *HOP-RL*. Both homogeneous and heterogeneous results use the same crowd densities and environments (approximately 1.9 agents/m^2 in the bottleneck scenario). The large standard deviation in the less heterogeneous scenarios is due to singular agents with random velocities in the extremes of the full sampling range [0.5,1.6]m/s. Very slow agents may induce laminar, or lane-forming flows, while very fast agents may induce rapid oscillations in the crowd from pushing and overtaking.

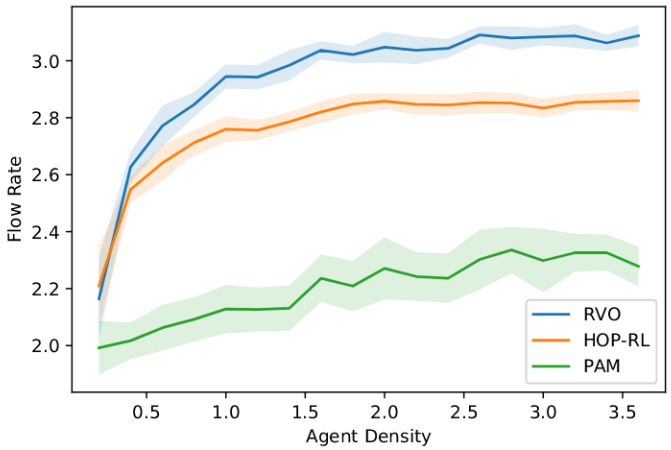


Fig. 14: Density dependent artifacts found in synthetic crowd steering models. Both *RVO* and *HOP-RL* reproduce the plateau artifacts shown in most other traditional steering models without the aid of additional density modelling. *PAM* has a lower overall flow rate and can struggle to deal with the balance of forces since static obstacles induce no predictive avoidance forces.

quantitative multi-agent navigation outcomes such as flow $f(\mathbf{p})$ and speed. In real crowds, the benefits of increasing crowd density are lost at a critical point at which increasing density causes rapid decreases in throughput (measured by crowd speed or flow) [87], [88]. This artifact is not often reproduced by synthetic crowds models without an additional density model [24], [89]. We show that *HOP-RL* replicates this behaviour found in traditional methods.

In this experiment, we vary the density of a crowd in a simple bottleneck egress scenario. All the agents share the same desired walking speed of 1.33m/s and the same goal area after the egress point. We measure the flow rate of the crowd with respect to density for each model.

4.5.1 *Results & Discussion*

We compare density flow rate curves of the methods in Figure 14. The *RVO* model produces an increasing flow rate without a clear critical point and instead plateaus as expected [89]. *HOP-RL* performs similarly to the *RVO* model. The *PAM* model struggles in this scenario as the lack of a predictive force for static objects produces qualitative artifacts, like the violation of agent bounds i.e. passing through agents, at higher densities.

4.6 Computational Performance

We evaluate the computational performance of our final proposed model. This is done by randomly sampling a procedural floor plan generator. Previous work has shown that modern building energy modelling forms a type of typology [90]. We adapt this floorplan typology matrix to generate arbitrarily large-scale realistic buildings to facilitate arbitrarily large crowds. All agents are randomized (random targets and initialization) and we capture the average frames-per-second (FPS) value of the scenario. This measure allows us to view at which point performance drops below real-time as a function of the number of agents. We generate scenarios with varying numbers of agents from 1 to 3000 total simultaneously navigating agents.

We compare the computational performance (FPS) curves of the methods in Figure 15. The results show that *HOP-RL* performs on par with the industry standard *RVO* steering model used in games, animation, robotics, and crowd analysis. This *RVO* model is a highly optimized internal (black box) implementation shipped with the Unity game engine. Both have a real-time use (30 FPS) at around 500-600 agents and converge on performance after around 2000 agents. The *PAM* model evaluated is our own implementation which performs marginally worse overall.

4.7 Extensibility

Section 3.2 outlines the conditions needed, or expected, for parametric policy training to succeed. These conditions include 1) the parameter being visible in the state, 2) the parameter having some effect on or being valued by the reward, and 3) having some effect on the action space. We evaluate the extensibility of *HOP-RL* by training over another parameter. We note that the agent radius, or personal

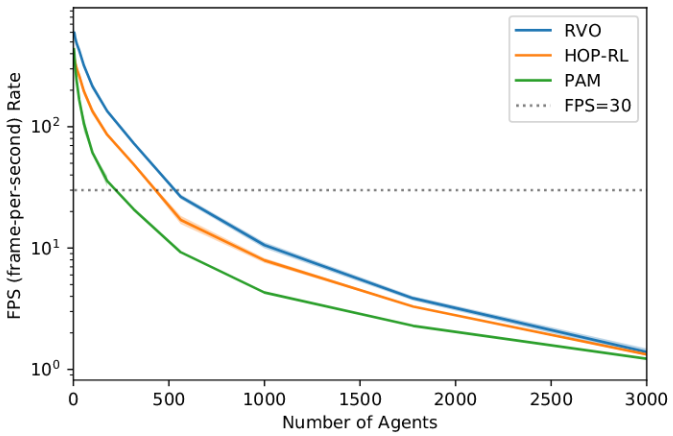


Fig. 15: Mean frames per second (FPS), with standard deviations, over several different agent counts. Here, *RVO* is an internal (black box) highly optimized implementation shipped in the Unity game engine. This experiment shows *HOP-RL* performs similarly to this version of *RVO*, while our implementation of *PAM*, another force-based predictive collision avoidance approach, marginally underperforms. This is, in part, because the calculations that a forced-based approach would do (i.e. neighbour lookups, and pairwise predictive avoidance checks) are avoided in the proposed method, a benefit of relatively simple neural network architectures and well-crafted state observations. The real-time performance cut-off (30 FPS) is highlighted in the figure.

Fig. 16: *HOP-RL* agents learn policies with emergent laminar flow with both heterogeneous desired speed and personal space in the two way hallway scenario.

space, already impacts the collision portion of the reward and affects the decisions made in the action space. We repeat the training procedure of *HOP-RL* but include agent radius (or personal space), in addition to the desired speed, by uniformly sampling over the interval of $[0.3, 1.2]$m.

The result of this experiment is a model that is also parameterized on personal space. *HOP-RL* successfully learns a parametric policy that affords highly heterogeneous agents, which respects the bounds of the parametric personal space for each agent. This parameter allows a practitioner to make tight packing crowds and less uniform behaviour as smaller agents may try to squeeze between others. Additionally, it may be used to model pertinent behaviours such as social distancing. This can be seen in Figure 16 in a two-way hallway and in Figure 17 in a four-way hallway scenario.

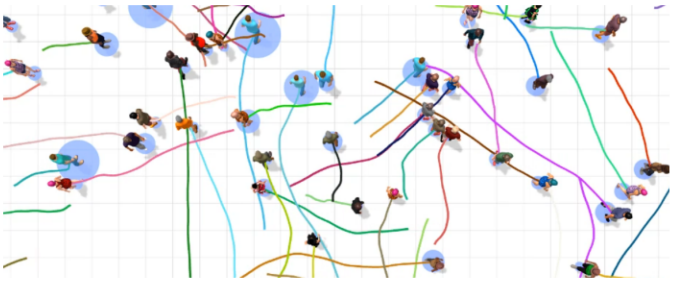


Fig. 17: *HOP-RL* agents learn good policies for navigating the difficult four-way hallway scenario with both heterogeneous desired speed and personal space.

5 Conclusion

In this paper, we have presented a parametrizable reinforcement learning-based and "model-free" approach to heterogeneous agent-based crowd simulation. We have shown that the model learns predictive reciprocal collision avoidance. Additionally, we performed several quantitative experiments that show that the method has competitive computational performance, reproduces traditional behaviours in high-density scenarios, and produces diverse heterogeneous results. An important outcome of this work was to show that reinforcement learning can produce post-training parametric models. This directly supports the authoring of diverse crowds by practitioners in a usable manner.

5.1 Limitations & Future Work

We recognize that there are numerous diverse gaits, cultural observations, contexts (like COVID-19, social distancing) that dictate how agents representing humans may interact and that the proposed model does not directly account for all of these. Often crowd simulators and practitioners must carefully balance parameter settings or add additional layers of intelligence, such as decision or behaviour trees, to capture specific behaviours. Practitioners could our method by adding their own rewards or penalties during training to guide learning toward policies they want or add their own behavioural parameters as we have done here with desired speed and personal space. In fact, there are many more parameters one can imagine being made open to the end-user. Deep Reinforcement Learning also allows us to learn more complex tasks and behavior and another avenue of future exploration would be higher level behaviours, possibly using hierarchical reinforcement learning or behaviour cloning, to break up the problem into layers of intelligence similar to classic crowds does.

Heterogeneity in this context is concerned largely with the past literature and is related to velocity control and personal space. However, human locomotion heterogeneity is much more complex and the modelling of diverse crowds is of deep importance in fields that use synthetic crowds. For instance, models with different underlying action spaces, for example, non-holonomic differential controllers (like wheelchairs) are not modelled directly in this approach. However, we hypothesize that it is possible to learn heterogeneous control in synthetic crowds. To achieve this, future work will include investigating mixed-model diversity learning.

ACKNOWLEDGMENTS

The research was supported in part by the Murray Postdoctoral Fellowship, NSERC Create DAV, Ontario Research Foundation (Grant No. RE08-054), NSERC Discovery [funding reference number RGPIN-2021-03541], and NSF awards: IIS-1703883, S&AS-1723869, IIS-1955404, IIS-1955365, RETTL-2119265, and EAGER-2122119.

REFERENCES

[1] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," in *Computer graphics forum*, vol. 26, no. 3. Wiley Online Library, 2007, pp. 655–664.

[2] J. Ondřej, J. Pettré, A.-H. Olivier, and S. Donikian, "A synthetic-vision based steering approach for crowd simulation," in *ACM TOG*, vol. 29, no. 4. ACM, 2010, p. 123.

[3] D. Wolinski, M. C. Lin, and J. Pettré, "Warpdriver: context-aware probabilistic motion prediction for crowd simulation," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, p. 164, 2016.

[4] G. Berseth, M. Kapadia, and P. Faloutsos, "Robust space-time footsteps for agent-based steering," *Computer Animation and Virtual Worlds*, 2015.

[5] S. Singh, M. Kapadia, G. Reinman, and P. Faloutsos, "Footstep navigation for dynamic crowds," *Computer Animation and Virtual Worlds*, vol. 22, no. 2-3, pp. 151–158, 2011.

[6] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *ACM Siggraph Computer Graphics*, vol. 21, no. 4. ACM, 1987, pp. 25–34.

[7] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*. Springer, 2011, vol. 70, pp. 3–19. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-19457-3-1

[8] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1928–1935.

[9] I. Karamouzas, P. Heil, P. van Beek, and M. H. Overmars, "A predictive collision avoidance model for pedestrian simulation," in *MiG*. Springer, 2009, pp. 41–52.

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

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

[12] S. Singh, M. Kapadia, B. Hewlett, G. Reinman, and P. Faloutsos, "A modular framework for adaptive agent-based steering," in *Proceedings of I3D*. ACM, 2011, pp. 141–150. [Online]. Available: http://doi.acm.org/10.1145/1944745.1944769

[13] X. Lan, Y. Liu, and Z. Zhao, "Cooperative control for swarming systems based on reinforcement learning in unknown dynamic environment," *Neurocomputing*, vol. 410, pp. 410–418, 2020.

[14] L. Sun, J. Zhai, and W. Qin, "Crowd navigation in an unknown and dynamic environment based on deep reinforcement learning," *IEEE Access*, vol. 7, pp. 109 544–109 554, 2019.

[15] 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*. ACM, 2018, p. 2.

[16] F. Martinez-Gil, M. Lozano, and F. Fernandez, "Emergent behaviors and scalability for multi-agent reinforcement learning-based pedestrian models," *Simulation Modelling Practice and Theory*, vol. 74, pp. 117–133, 2017.

[17] F. Martinez-Gil, M. Lozano, and F. Fernández, "Strategies for simulating pedestrian navigation with multiple reinforcement learning agents," *Autonomous Agents and Multi-Agent Systems*, vol. 29, no. 1, pp. 98–130, 2015.

[18] ——, "Marl-ped: A multi-agent reinforcement learning based framework to simulate pedestrian groups," *Simulation Modelling Practice and Theory*, vol. 47, pp. 259–275, 2014.

[19] L. Torrey, "Crowd simulation via multi-agent reinforcement learning," in *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, ser. AIIDE'10. AAAI Press, 2010, pp. 89–94. [Online]. Available: http://dl.acm.org/citation.cfm?id=3014666.3014683

[20] L. Hoyet, A.-H. Olivier, R. Kulpa, and J. Pettré, "Perceptual effect of shoulder motions on crowd animations," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–10, 2016.

[21] R. McDonnell, M. Larkin, S. Dobbyn, S. Collins, and C. O'Sullivan, "Clone attack! perception of crowd variety," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, p. 26, 2008.

[22] M. Kapadia, M. Wang, S. Singh, G. Reinman, and P. Faloutsos, "Scenario space: characterizing coverage, quality, and failure of steering algorithms," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 2011, pp. 53–62.

[23] M. B. Haworth, "Biomechanical locomotion heterogeneity in synthetic crowds," Ph.D. dissertation, York University, Toronto, Canada, November 2019.

[24] B. Haworth, M. Usman, G. Berseth, M. Kapadia, and P. Faloutsos, "On density–flow relationships during crowd evacuation," *Computer Animation and Virtual Worlds*, vol. 28, no. 3-4, 2017.

[25] S. Singh, M. Kapadia, P. Faloutsos, and G. Reinman, "Steerbench: A benchmark suite for evaluating steering behaviors," *Computer Animation and Virtual Worlds*, vol. 20, no. 5-6, pp. 533–548, 2009.

[26] F. Martinez-Gil, M. Lozano, I. García-Fernández, and F. Fernández, "Modeling, evaluation, and scale on artificial pedestrians: a literature review," *ACM Computing Surveys (CSUR)*, vol. 50, no. 5, p. 72, 2017.

[27] N. Pelechano, J. M. Allbeck, and N. I. Badler, *Virtual Crowds: Methods, Simulation, and Control*. Morgan & Claypool Publishers, 2008.

[28] S. Huerre, J. Lee, M. Lin, and C. O'Sullivan, "Simulating believable crowd and group behaviors," in *ACM SIGGRAPH ASIA 2010 Courses*, pp. 13:1–13:92.

[29] D. Thalmann and S. R. Musse, *Crowd Simulation, Second Edition*. Springer, 2013.

[30] G. Taga, Y. Yamaguchi, and H. Shinizu, "Self-organized control of bipedal locomotion by neural oscillators in unpredicatable environments," *Biological Cybernetics*, vol. 65, no. 3, pp. 147–159, 1991.

[31] T. Geng, B. Porr, and F. Wörgötter, "A reflexive neural network for dynamic biped walking control." *Neural Computation*, vol. 18, no. 5, pp. 1156–96, 2006.

[32] A. Kun and W. T. Miller III, "Adaptive dynamic balance of a biped robot using neural networks," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. pages. IEEE, 1996, pp. 240–245.

[33] W. T. Miller III, "Real-time neural network control of a biped walking robot," *Control Systems, IEEE*, vol. 14, no. 1, pp. 41–48, 1994.

[34] D. Holden, T. Komura, and J. Saito, "Phase-functioned neural networks for character control," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–13, 2017.

[35] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 41, 2017.

[36] B. Haworth, G. Berseth, S. Moon, P. Faloutsos, and M. Kapadia, "Deep integration of physical humanoid control and crowd navigation," in *Motion, Interaction and Games*, 2020, pp. 1–10.

[37] P. Faloutsos, M. Van de Panne, and D. Terzopoulos, "Composable controllers for physics-based character animation," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 251–260.

[38] W. Huang, I. Mordatch, and D. Pathak, "One policy to control them all: Shared modular policies for agent-agnostic control," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4455–4464.

[39] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[40] J. Won and J. Lee, "Learning body shape variation in physics-based characters," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–12, 2019.

[41] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6015–6022.

[42] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 007–10 013.

[43] W. van Toll and J. Pettré, "Algorithms for microscopic crowd simulation: Advancements in the 2010s," in *Computer Graphics Forum*, vol. 40, no. 2. Wiley Online Library, 2021, pp. 731–754.

[44] M. Kapadia, N. Pelechano, J. Allbeck, and N. Badler, "Virtual crowds: Steps toward behavioral realism," *Synthesis lectures on visual computing: computer graphics, animation, computational photography, and imaging*, vol. 7, no. 4, pp. 1–270, 2015.

[45] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[46] ——, "Motion planning in dynamic environments using the relative velocity paradigm," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 560–565.

[47] G. Berseth, M. Kapadia, B. Haworth, and P. Faloutsos, "Steerfit: Automated parameter fitting for steering algorithms," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 2014, pp. 113–122.

[48] Q. Cheng, Z. Duan, and X. Gu, "Data-driven and collision-free hybrid crowd simulation model for real scenario," in *International Conference on Neural Information Processing*. Springer, 2018, pp. 62–73.

[49] 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.

[50] H. Li, B. Weng, A. Gupta, J. Pan, and W. Zhang, "Reciprocal collision avoidance for general nonlinear agents using reinforcement learning," *arXiv preprint arXiv:1910.10887*, 2019.

[51] N. Bisagno, N. Garau, A. Montagner, and N. Conci, "Virtual crowds: An lstm-based framework for crowd simulation," in *International Conference on Image Analysis and Processing*. Springer, 2019, pp. 117–127.

[52] 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.

[53] 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.

[54] J. Amirian, J.-B. Hayet, and J. Pettré, "Social ways: Learning multi-modal distributions of pedestrian trajectories with gans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[55] B. Brito, H. Zhu, W. Pan, and J. Alonso-Mora, "Social-vrnn: one-shot multi-modal trajectory prediction for interacting pedestrians," *arXiv preprint arXiv:2010.09056*, 2020.

[56] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*. Springer, 2020, pp. 683–700.

[57] K. Mangalam, Y. An, H. Girase, and J. Malik, "From goals, waypoints & paths to long term human trajectory forecasting," *arXiv preprint arXiv:2012.01526*, 2020.

[58] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 981–986.

[59] F. Martinez-Gil, M. Lozano, and F. Fernández, "Multi-agent reinforcement learning for simulating pedestrian navigation," in *International Workshop on Adaptive and Learning Agents*. Springer, 2011, pp. 54–69.

[60] L. Casadiego and N. Pelechano, "From one to many: Simulating groups of agents with reinforcement learning controllers," in *International Conference on Intelligent Virtual Agents*. Springer, 2015, pp. 119–123.

[61] L. Casadiego Bastidas, "Social crowd controllers using reinforcement learning methods," Master's thesis, Universitat Politècnica de Catalunya, 2014.

[62] B. Haworth, M. Usman, G. Berseth, M. Kapadia, and P. Faloutsos, "Evaluating and optimizing level of service for crowd evacuations," in *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*. ACM, 2015, pp. 91–96.

[63] W. Shao and D. Terzopoulos, "Autonomous pedestrians," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 2005, pp. 19–28.

[64] M. Kapadia, S. Singh, B. Allen, G. Reinman, and P. Faloutsos, "Steerbug: an interactive framework for specifying and detecting steering behaviors," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 2009, pp. 209–216.

[65] J. Pettré, J. Ondrej, A.-h. Olivier, A. Cretual, and S. Donikian, "Experiment-based modeling, simulation and validation of interactions between virtual walkers," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, vol. 2009, 2009, p. 189.

[66] M. Chraibi, T. Ensslen, H. Gottschalk, M. Saadi, and A. Seyfried, "Assessment of models for pedestrian dynamics with functional principal component analysis," *Physica A: Statistical Mechanics and its Applications*, vol. 451, pp. 475–489, 2016.

[67] M. Kapadia, M. Wang, G. Reinman, and P. Faloutsos, "Improved benchmarking for steering algorithms." in *Proceedings of the 4th International Conference on Motion in Games*. Springer, 2011, pp. 266–277.

[68] M. Kapadia, M. Wang, S. Singh, G. Reinman, and P. Faloutsos, "Scenario space: characterizing coverage, quality, and failure of steering algorithms," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 2011, pp. 53–62.

[69] D. Wolinski, S. J. Guy, A.-H. Olivier, M. C. Lin, D. Manocha, and J. Pettré, "Optimization-based pedestrian model calibration for evaluation," *Transportation Research Procedia*, vol. 2, pp. 228–236, 2014.

[70] ——, "Parameter estimation and comparative evaluation of crowd simulations," *Computer Graphics Forum*, vol. 33, no. 2, pp. 303–312, 2014.

[71] N. Pelechano, C. Stocker, J. Allbeck, and N. Badler, "Being a part of the crowd: towards validating vr crowds using presence," in *Proceedings of the 7th international Joint Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 136–142.

[72] B. C. Daniel, R. Marques, L. Hoyet, J. Pettré, and J. Blat, "A perceptually-validated metric for crowd trajectory quality evaluation," *arXiv preprint arXiv:2108.12346*, 2021.

[73] J. K. Terry, N. Grammel, A. Hari, L. Santos, and B. Black, "Revisiting parameter sharing in multi-agent deep reinforcement learning," 2021.

[74] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu, "Sim-to-real transfer for biped locomotion," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov 2019. [Online]. Available: http://dx.doi.org/10.1109/IROS40897.2019.8968053

[75] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *NIPS*, 1999.

[76] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[77] ——, "Proximal policy optimization algorithms," 2017.

[78] A. D. Rasamoelina, F. Adjailia, and P. Sinčák, "A review of activation function for artificial neural network," in *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. IEEE, 2020, pp. 281–286.

[79] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017.

[80] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019. [Online]. Available: https://robotics.sciencemag.org/content/4/26/eaau5872

[81] M. Kallmann and M. Kapadia, "Navigation meshes and real-time dynamic planning for virtual worlds," in *ACM SIGGRAPH 2014 Courses*, 2014, pp. 1–81.

[82] N. Sohre and S. J. Guy, "Spnets: Human-like navigation behaviors with uncertain goals," in *Motion, Interaction and Games*, 2020, pp. 1–11.

[83] S. Niu, S. Chen, H. Guo, C. Targonski, M. C. Smith, and J. Kovačević, "Generalized value iteration networks: Life beyond lattices," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[84] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 2154–2162.

[85] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[86] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *International Conference on Learning Representations (ICLR 2016)*, 2016.
[87] P. Wang, S. Cao, and M. Yao, "Fundamental diagrams for pedestrian traffic flow in controlled experiments," *Physica A: Statistical Mechanics and its Applications*, vol. 525, pp. 266–277, 2019.
[88] A. Seyfried, B. Steffen, W. Klingsch, and M. Boltes, "The fundamental diagram of pedestrian movement revisited," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 10, p. P10002, 2005.
[89] S. Curtis and D. Manocha, "Pedestrian simulation using geometric reasoning in velocity space," in *Pedestrian and Evacuation Dynamics 2012*. Springer, 2014, pp. 875–890.
[90] T. Dogan, E. Saratsis, and C. Reinhart, "The optimization potential of floor-plan typologies in early design energy modeling," in *Proceedings of BS2015: 14th Conference of International Building Performance Simulation Association*. International Building Performance Simulation Association (IBPSA), 12 2015.

Kaidong Hu is a Ph.D. candidate at Rutgers. She received her B.Sc. in Chemistry from Nanjing University in 2016. Her research interests focus on providing human behavior-like AI by emulate the natural development (learning) process. She is a member of the Intelligent Visual Interfaces Lab supervised by Dr. Mubbasir Kapadia.

Brandon Haworth is an Assistant Professor in the Department of Computer Science, Faculty of Engineering & Computer Science at the University of Victoria. He is also the Director of the Graphics, Artificial Intelligence, Design, & Games (GAIDG) Lab and a Research Fellow at the Institute on Aging and Lifelong Health at the University of Victoria. Brandon works within the broad areas of Graphics, Simulation, Artificial Intelligence, and Human-Computer Interaction. His primary research focuses are diversity in crowds simulations, locomotion & biomechanical modelling in human steering, multi-agent reinforcement learning, and human-centric artificial & augmented intelligence in simulation and design with a purpose to explore the intersections between visibility, representation, and decision-making in interactive technologies.

Glen Berseth is an assistant professor at the University of Montreal and a member of Mila. He was a Postdoctoral Researcher at the Berkeley Artificial Intelligence Research (BAIR) working with Sergey Levine. He completed his NSERC-awarded Ph.D. in Computer Science at the University of British Columbia in 2019, where he worked with Michiel van de Panne. His goal is to create systems that can learn and act in the world intelligently by developing deep learning and reinforcement learning methods to solve diverse, high-dimensional perception and planning problems.

Vladimir Pavlovic is a Professor of Computer Science at Rutgers University in New Jersey, USA. Vladimir's research interests include probabilistic machine learning, multimodal representation learning, and next generation human sensing. Over the past twenty years Vladimir has published extensively in the domains of computer vision and human-computer interaction, including his seminal works on human gesture modeling and recognition, human motion analysis, and non-verbal human affect understanding. At Rutgers, he co-leads the Center for Accelerated Real Time Analytics (CARTA), is a member of the Executive Committee of Rutgers Center for Cognitive Science (RUCCS), and an associate member of the Center for Quantitative Biology (CQB). Vladimir received his Ph.D. in Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign.

Petros Faloutsos is a Professor at the Department of Electrical Engineering and Computer Science at York University, and an affiliate Scientist at the UHN-Toronto Rehabilitation Institute. Before joining York, he was a faculty member at the Computer Science Department at the University of California at Los Angeles, where in 2002 he founded the first computer graphics lab at UCLA. Faloutsos received his PhD degree (2002) and his MSc degree in Computer Science from the University of Toronto, Canada and his BEng degree in Electrical Engineering from the National Technical University of Athens, Greece.

Mubbasir Kapadia is currently the Director of the Intelligent Visual Interfaces Lab and an Associate Professor with the Computer Science Department, Rutgers University, New Brunswick, NJ, USA. Previously, he was an Associate Research Scientist with Disney Research Zurich. His research lies at the intersection of artificial intelligence, visual computing, and humancomputer interaction, with a mission to develop intelligent visual interfaces to empower content creation for human-aware architectural design, digital storytelling, and serious games.