Deep Integration of Physical Humanoid Control and Crowd Navigation

Brandon Haworth[*]
bhaworth@uvic.ca
University of Victoria
Victoria, British Columbia, Canada

Glen Berseth[*]
gberseth@berkeley.edu
University of California, Berkeley
Berkeley, California, USA

Seonghyeon Moon
sm2062@cs.rutgers.edu
Rutgers University
New Brunswick, New Jersey, USA

Petros Faloutsos
pfal@eecs.yorku.ca
York University
University Health Network:
Toronto Rehabilitation Institute
Toronto, Ontario, Canada

Mubbasir Kapadia
mk1353@cs.rutgers.edu
Rutgers University
New Brunswick, New Jersey, USA

ABSTRACT

Many multi-agent navigation approaches make use of simplified representations such as a disk. These simplifications allow for fast simulation of thousands of agents but limit the simulation accuracy and fidelity. In this paper, we propose a fully integrated physical character control and multi-agent navigation method. In place of sample complex online planning methods, we extend the use of recent deep reinforcement learning techniques. This extension improves on multi-agent navigation models and simulated humanoids by combining Multi-Agent and Hierarchical Reinforcement Learning. We train a single short term goal-conditioned low-level policy to provide directed walking behaviour. This task-agnostic controller can be shared by higher-level policies that perform longer-term planning. The proposed approach produces reciprocal collision avoidance, robust navigation, and emergent crowd behaviours. Furthermore, it offers several key affordances not previously possible in multi-agent navigation including tunable character morphology and physically accurate interactions with agents and the environment. Our results show that the proposed method outperforms prior methods across environments and tasks, as well as, performing well in terms of zero-shot generalization over different numbers of agents and computation time.

CCS CONCEPTS

• **Computer systems organization → Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

Multi-Agent Learning, Crowd Simulation, Physics-based Simulation

[*]Both authors contributed equally to the paper

MIG '20, October 16–18, 2020, Virtual Event, SC, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8171-0/20/10…$15.00
https://doi.org/10.1145/3424636.3426894

ACM Reference Format:
Brandon Haworth, Glen Berseth, Seonghyeon Moon, Petros Faloutsos, and Mubbasir Kapadia. 2020. Deep Integration of Physical Humanoid Control and Crowd Navigation. In *Motion, Interaction and Games (MIG '20), October 16–18, 2020, Virtual Event, SC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3424636.3426894

1 INTRODUCTION

The simulation and animation of crowds, or multi-agent navigation, is an important and difficult task. Methods which produce solutions that operate in more active environments have many uses, from NPCs in computer games to simulating crowds in engineering applications. Given the many uses for such models, we are motivated to construct as realistic and high fidelity a model as we can. However, simulating the complex dynamics of numerous characters and their intentions is difficult. In addition, due to limited information about the intentions of other agents it is extremely difficult to construct rules, policies, or plans that are not invalidated by the actions of other agents. This paper seeks to address the issues inherent in learning such policies for high-fidelity physically-enabled characters.

Most recently, methods have been proposed to address prior limitations in multi-agent navigation with Deep Reinforcement Learning (DRL) approaches. While, DRL has been successful in solving complex planning tasks given a sizeable computational budget [30, 40], the multi-agent navigation problem turns the Reinforcement Learning (RL) problem into a Multi-Agent Reinforcement Learning (MARL) problem. However, MARL is a very difficult problem. The non-stationary learning of multiple changing policies in largely heterogeneous environments is not easily overcome by collecting more data [32]. The trend to make progress on MARL for multi-agent navigation has been to simplify the learning problem. For example, converting the multi-agent problem into a single agent *Centralized* model results in gains in performance but can increase the number of network parameters significantly and impose a constraint on generalizing to different numbers of agents [26]. While these methods have shown promise, they require significant amounts of compute and have not yet displayed success in complex and dynamic multi-agent environments with articulated heterogeneous agents.

Most multi-agent steering and navigation approaches, including the recent RL approaches, represent individual agents as a simplified particle disk, or point-mass. The underlying steering models prior to applications of RL have been generally data-driven or built on top of expertly defined rules. This leads to plausible emergent but highly approximated microscopic behaviours and interactions. These approaches also result in a decoupled steering and locomotion system, which can limit a simulation from producing important behaviours. Steering and collisions are high fidelity phenomenon in the real world. In computer animation, often the steering and navigation layer is separate from the animation layer which handles artifacts like footskate and produces visible reactions to collisions or falls. Due to this separation, animation layers need to produce complex physical phenomenon in navigation, steering, and collision avoidance/response from low dimensional information. The domain of physical character control has recently made great progress using RL-based methods to improve the simulation of physical phenomenon for robust articulated characters. However, most approaches assume a closed environment without additional characters with agency of their own and instead focus on controlling the biomechanical aspects of a single character.

In this work, we couple, for the first time, physical character control and multi-agent navigation for robust physical animation of interacting characters. Specifically, we propose a method to reduce the complexity in the MARL policy learning problem by separating physical locomotion and navigation policies while encapsulating them in one sensory-motor feedback loop inspired by human locomotion. This is achieved by enforcing a mid-level representation (footstep plans) and learning a ubiquitous and task-agnostic lower-level skill controller (bipedal walking skills) for task-agnostic portions of the policy. The higher-level policy learns navigation and behaviour skills guided by rewards. This use of Hierarchical Reinforcement Learning (HRL), with a goal conditioned lower module [17], allows for exploratory behaviour that is more consistent in space and time. This approach also allows for heterogeneous high-level behaviour in a MARL setting where agents are expected to interact. The combination of high-fidelity physical simulation, adding structure to a difficult learning problem, and data-driven machine learning produces a new approach which affords the simulation and animation of high-fidelity, physically-enabled crowds. This method represents the first method for heterogeneous multi-agent physical character control for locomotion, navigation, and behaviour.

2 RELATED WORK

In this section, we outline the most related work in the areas of character and multi-agent control.

2.1 Multi-Agent Navigation

Human movement and behaviour simulation has a long and rich history in the literature [18, 33, 45]. This includes data-driven, physical, geometric, probabilistic, and optimization based methods. In this review we focus on machine learning and physical methods related to the proposed method. A standard method used to implement human-like behaviours is to represent the components that humans are concerned with during navigation as physical forces, pushing and pulling the agent toward their goal and away from collisions. This approach is famously derived from the particle-based Social Forces model [14, 15, 20]. As well, the class of velocity obstacle approaches has been used extensively in games for its fast and collision free solutions [48, 49, 52]. These velocity obstacle approaches have been combined with external force constraints to create more physically enabled approaches [21]. As well, footstep-based models derived from physical models representing bipedal locomotion as an inverted pendulum produce tight-packing, high fidelity steering in crowds [3, 41]. However, these methods consider a geometric approximation to the biomechanical physical model, represent humans as more particles, and choose step actions as a function of step-wise energy costs. This does not consider balance control or complex steps. As well, learning methods may learn new steps not previously seen, and the proposed method can produce complex balance control of arbitrarily detailed fully articulated characters.

More recent learning-based methods using RL have shown to map particularly well to the agent movement simulation problem both conceptually and in practice [27, 46]. Models have learned continuous actions using a curriculum training approach, like prior expertly defined models [24]. Most recently, Generative Adversarial Networks have been used to generate *socially acceptable* trajectories to resolve the collision free steering problem of crowds [12]. This approach resolves the issue of learning an average or singular policy for collision avoidance outcomes. Instead, the GAN approach affords several different but acceptable possible outcomes. In contrast, our method avoids particle-based or trajectory-based models entirely, in favour of drastically more complex humanoid models to enabled detailed physical simulations that allow us to capture additional dynamical aspects of multi-agent interactions. The proposed method resolves physical full-body articulation of humanoid characters, navigation, and locomotion together in one cohesive approach.

2.2 Character Control

Simulated robot and physical character control is a vibrant area of research with many solutions and approaches. This area of research is beyond the scope of this paper. Here, we focus on a particular set of representative approaches that use optimization, learning, and Artificial Neural Network (ANN) based methods. Early biped models recreated neural oscillators to produce walking patterns [42]. Neural models focused on training neural networks by receiving joint or body sensor feedback as input and producing appropriate joint angles as output [11, 23, 29]. It has been shown that this type of walking behaviour can be evolved by using evolutionary optimization techniques on complex neural networks, which eventually produce oscillator patterns [1, 2]. A biped character's movement controller set can also be manually composed using simple control strategies and feedback learning [9, 53].

Recent RL methods have used humanoid control as a benchmark for their learning techniques [35, 37, 39]. This work has encountered a *representation learning bottleneck*: the policy must first learn a good representation of the state before it can produce an effective policy [16, 51]. HRL has been proposed as a solution to handling challenges with current RL techniques that have trouble estimating

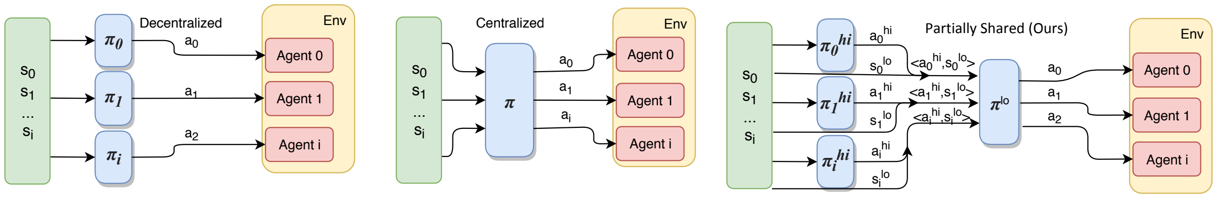


Figure 1: The *Decentralized* method is the most general but also the most difficult to optimize due to non-stationary environments with no assumptions about shared structure across agents. This approach is the closest to fully autonomous agent-based models used in crowds. The *Centralized* method effectively converts the problem into a single agent system, which in turn limits its application and assumes access to the full state. In this paper, we propose a fully decentralized hierarchical approach with partial parameter sharing in hierarchy. Our *decentralized partial sharing* approach strikes a balance between these models, preserving generality while introducing beneficial structure.

rewards over long horizons and sparse signal by enforcing an important goal representation. One difficulty in HRL design is finding a reasonable communication representation to condition the lower level. Some methods pretrain the lower level on a random distribution [28, 34]. While these methods have made great progress on physics-based humanoid character control the proposed method addresses, in addition, the multi-agent navigation problem. In this setting, there are other characters in the environment with their own agency and goals that can directly interfere with other agents and how behaviours change over time as their policies are trained, resulting in a very complex optimization problem to make progress on. Most physical character control approaches solve a closed-loop problem, while multi-agent navigation is an open-loop control problem. These issues make the problem this paper addresses extremely difficult. The proposed approach is novel and there are no existing prior approaches which address it.

2.3 Multi-Agent Reinforcement Learning

There are many types of multi-agent learning problems, including cooperative-competitive and with-without communication [6, 8, 43]. While progress is being made, MARL is notoriously tricky due to the non-stationary optimization issue, even in the cooperative case [7, 31]. Recent work, converts the MARL problem to a single agent setting by using a single Q-function across all agents [26]. Other recent work has begun to combine MARL and HRL but is limited to simple discrete grid environments, uses additional methods to stabilize the optimization, and includes communication [13, 44]. Instead, our work tackles multi-agent articulated humanoid simulation by applying a combination of goal conditioned learning and partial *parameter sharing* by assuming all agents share task-agnostic locomotion and optimize similar goals which allows us to keep the modularity and autonomy benefits of decentralized methods while significantly reducing the model size.

3 BACKGROUND

Reinforcement learning is formulated on the Markov Dynamic Process (MDP) framework: at every time step t, the world (including the agent) exists in a state $s_t \in S$, wherein the agent is able to perform an action $a_t \in A$, sampled from a parameterized policy $\pi(a_t|s_t, \theta)$ which results in a new state $s_{t+1} \in S$ according to the transition probability function $P(s_{t+1}|a_t, s_t)$ with the initial state distribution $p_0(s_0)$. Performing action a_t in state s_t produces a reward $r_t = R(s_t, a_t)$ from the environment; the expected future discounted reward from executing a policy with parameters θ is:

$$J(\theta) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t,\theta), s_{t+1} \sim P, s_0 \sim p_0} \left[\sum_{t=0}^{T} \gamma^t R(s_t, a_t) \right] \quad (1)$$

where T is the maximum time horizon, and γ is the discount factor, indicating the planning horizon length. The agent's goal is to optimize its policy, $\pi(\cdot|\cdot, \theta)$, by maximizing $J(\theta)$.

3.1 Hierarchical Reinforcement Learning

In HRL, the original MDP is separated into different MDPs that are each easier to solve. In practice, this is accomplished by learning two different policies in two different temporal resolutions. The lower-level policy is trained first and is often conditioned on a latent variable or goal g. The lower-level policy $\pi(a|s, g, \theta^{lo})$ is constructed in such a way as to give it temporally correlated behaviour depending on g. After the lower level policy is trained, it is used to help solve the original MDP using a separate policy $\pi(g|s, \theta^{hi})$. This policy learns to provide goals to the lower policy to maximize rewards. This improves exploration and, in our proposed approach, reduces the MARL problem from learning the details of locomotion to learning goal-based footstep plans for each agent.

3.2 Multi-Agent Reinforcement Learning

The extension to the MDP framework for MARL is a partially observable Markov game [25]. A Markov game has a collection of N agents, each with its own set of actions $A^0, \ldots, A^N$ and partial observations $X^0, \ldots, X^N$ of the full state space S. Each agent i has its own policy $\pi(a|x^i, \theta^i)$ that models the probability of selecting an action for each agent. The environment transition function is a function of the full state and every agent's action $P(S'|S, a^0, \ldots, a^N)$. Each agent i receives a reward r^i for taking a particular action a^i given a partial observation x^i and its objective is to maximize this reward over time $\sum_{t=0}^{T} \gamma^t r_t^i$, where γ is the discount factor and T is the time horizon. The policy gradient can be computed for each agent as

$$\nabla_{\theta^i} J(\pi(\cdot|\cdot, \theta^i)) = \int_{X^i} d_{\theta^i}(x^i) \int_{A^i} \nabla_{\theta^i} \log(\pi(a^i|x^i, \theta^i)) A_{\theta^i}(x^i, a^i) \, da^i \, dx^i \quad (2)$$

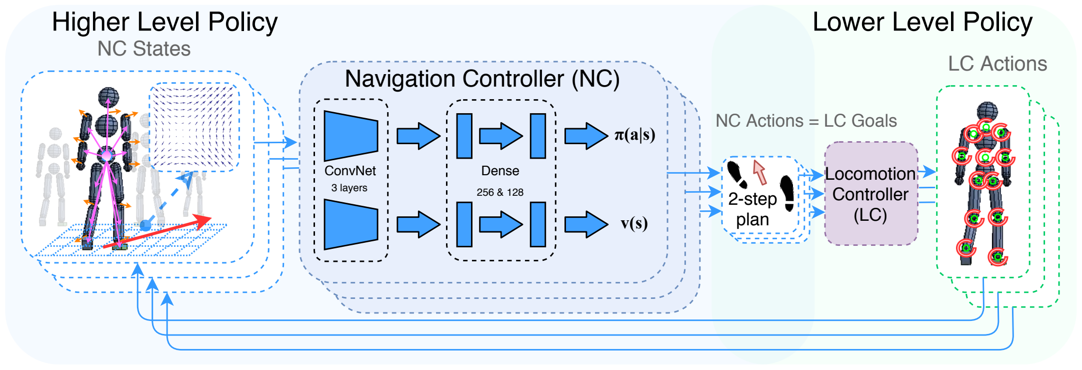


Figure 2: From left to right: the Multi-Agent Navigation Controller (NC) state includes the relative goal position and distance, an egocentric velocity field of the relative velocity of obstacles and other agents, and the agent link positions and linear velocities; for each agent this state is input to a multi-layer Convolutional Neural Network (CNN), including two dense hidden layers, and outputs actions–the value function uses a similar network. These high-level actions, in the form of two-step plans, are given to the Locomotion Controller (LC) as a g, which produces the angle-axis targets for each joint.

where $d_{\theta^i} = \int_{X^i} \sum_{t=0}^{T} \gamma^t p_0^i(x_0^i)(x_0^i \rightarrow x^i | t, \theta^i) dx_0^i$ is the discounted state distribution, $p_0^i(x^i)$ represents the initial state distribution for agent i, and $p_0^i(x_0^i)(x_0^i \rightarrow x^i | t, \theta^i)$ models the likelihood of reaching state x^i by starting at state x_0^i and following the policy $\pi(a^i, x^i | \theta^i)$ for T steps [39]. Here $A_{\theta^i}(x^i, a^i)$ represents the advantage function estimator GAE(λ) [36].

There are numerous approaches to solving the MARL problem. In the rest of the paper, we outline our proposed method in relation to prior approaches, as seen in Figure 1.

4 MULTI-AGENT HIERARCHICAL REINFORCEMENT LEARNING

We construct a multi-agent learning structure that takes advantage of hierarchical design, which we refer to as Multi-Agent Hierarchical Reinforcement Learning (MAHRL). We start by describing the lower level policy design, then we detail the multi-agent higher level.

4.1 Task-agnostic LC

The LC, the lower-level policy in our design, is designed to learn a robust and diverse policy $\pi(a_t | x_t, g_t^L, \theta^{lo})$ based on a latent goal g_t^L variable. The latent goals $g^L = \{\hat{p}_0, \hat{p}_1, \hat{\phi}_{root}\}$ consist of the agent root-relative distances of the next two footsteps on the ground plane and the desired facing direction at the first step's end. This goal description is motivated by work that shows people may plan steering decisions two-foot placements ahead [54]. The LC is trained to place its feet as accurately as it can to match these step goals using the reward $r_{Lg} = \exp(-0.2||\mathbf{s}_{char}^g - g^L||^2)$ where $\mathbf{s}_{char}^g$ is the foot placement action. This RL objective is defined as

$$\eta_L(\theta^{lo}) = \mathbb{E}_{s \sim p_{\theta^{lo}}} \left[\sum_{t=0}^{k} \gamma^t r_{Lg}(s_t, g_t^L, \pi(s_t, g_t^L | \theta^{lo})) \right]. \tag{3}$$

The better the LC learns this behaviour, the more responsive the controller will be to provided goals. More details on the LC network design an training can be found in Section 5.2.

4.2 Hierarchical Multi-Agent Learning

Each agent has its own higher-level policy (NC) $\pi(g^L | x, \theta^i)$ and a shared task agnostic lower level policy (LC) $\pi(a | x, g^L, \theta^{lo})$ then the full optimization objective with decentralized hierarchical policies and multiple agents receiving observations is:

$$\eta'_m = \eta_H(\theta^i) + \eta_L(\theta^{lo}) \tag{4}$$

$$= \mathbb{E}_{x^i \sim p_{\theta^i}} \left[\sum_{t=0}^{T/k} \left[\gamma^t (r_H(x_t^i, \pi(\cdot | x_t^i, \theta^i)) \right] \right] \tag{5}$$

$$+ \mathbb{E}_{x^i \sim p_{\theta^{lo}}} \left[\sum_{t=0}^{k} \gamma^t r_L(x_t^i, g_t^L, \pi(\cdot | x_t^i, g_t^L, \theta^{lo})) \right] \tag{6}$$

where the higher-level policy operates once every k steps. We notice that if the separation between the two control levels is chosen carefully, we can reduce the complexity of this optimization with no loss in generality. In particular, in multi-agent navigation we can conceptually separate two control policies by looking at the human locomotor control loop of bipeds for inspiration. The human sensory-motor control loop involves supraspinal input and reasoning at the highest level with Central Pattern Generators at the mid-level and motor/sensory neurons controlling functional morphology at the lowest level [47]. We separate the proposed control policies into high-level sensing of the environment for planning, navigation, and behaviour, and then low-level physical control of joints and cyclic locomotion. We find that each of the lower-level policies is solving the same goal conditioned MDP and can, therefore, be shared across the independent higher-level policies. This method allows us to introduce more structure into the difficult multi-agent optimization problem. This change alters the underlying MDP, such that the policy is queried for a new action every k timesteps. This also changes the MDP method by reducing the dimensionality of the action space to specifying goals g while using the low-level policy to produce more temporally consistent behaviour in the original action space and further reduce variance introduced into the problem.

The use of HRL is key to the method. When the challenge in MARL is dealing with what can be large changes in the distribution of states visited by the agent, the use of a temporally correlated structure given by the shared goal-conditioned LC significantly reduces the non-stationarity. Not only is each agent sharing its network parameters, but this portion has also been carefully constructed to provide structured exploration for the task. This is in contrast to *centralized* methods that take a step away from the goal of solving the heterogeneous problem in a scalable way. The use of the LC controls the way $d_{\theta^i}(x^i)$ can change for each agent, making it easier for each agent to estimate other agents potential changes in behaviour due to the shared LC being trained to produce a useful behaviour that is a subset of the full space. As we will show later in the paper, this combination allows us to train capable humanoid navigation agents in a single day with modist compute.

5 LEARNING HIERARCHICAL MULTI-AGENT CONTROL

To solve the hierarchical learning problem, we train in a bottom-up fashion, training the LC first, and then sharing the LC policy among heterogeneous NC policies. The levels of the hierarchy communicate through shared actions and state in a feedback loop that is meant to reflect human locomotion, as seen in Figure 2. The policy parameters are optimized over the RL objective using the Proximal Policy Optimization (PPO) algorithm [37] unless otherwise specified.

5.1 Action & State Spaces

The LC is responsible for stable character locomotion. As such the LC action space controls per joint target positions for per joint PD controllers. The LC state is largely inspired by DeepLoco [34]. The NC's objective is to provide footstep placement goals $a_H = g_L$ for the LC. This shared action and state space allows the levels of the hierarchy to be fundamentally tied together. In addition to the footstep placement goals the LC takes in a number of other information useful for cyclic locomotion patterns, i.e. the articulated character state. This includes a desired goal oriented heading, the current centres of mass of each link in the character, their relative rotation, and angular velocities. Additionally, a phase variable provides information on stride progress. The LC is queried at 30Hz.

The NC uses as input an egocentric relative velocity field, located with respect ot the agent as in Figure 6a & 6a. This egocentric relative velocity field E is $32 \times 32 \times 2$ samples over a space of 5x5 m, starting 0.5 m behind the agent and extending 4.5 m in front, shown in the left hand side of Figure 2. The egocentric relative velocity field consists of two image channels in the x and y directions of a square area directly in front of the agent, such that each point in the field is a vector (x,y). Each sample is calculated as the velocity relative to the agent, including both agents and static obstacles [5, 50]. The current pose of the agent is included next, followed by the NC goal. The NC goal g^H consists of two values, the agent relative direction and the distance to the current spatial goal location. As noted in Section 5.1, the actions space of the NC is a two-step, or stride, plan passed to the LC as input. The NC is queried at 2 Hz.

5.2 Network and Training

5.2.1 NC. The NC uses convolutional layers followed by dense layers. The particular network used is as follows: 8 convolutional filters of size 6×6 and stride 3×3, 16 convolutional filters of size 4×4 and stride 2×2, the structure is flattened and the character and goal features $\mathbf{s}_{char}, g_H$ are concatenated, a dense layer of 256 units and a dense layer of 128 units are used at the end. The network uses Rectified Linear Unit (ReLU) activations throughout except for after the last layer which uses a *tanh* activation that outputs values between $[-1, 1]$. All network inputs, S, are normalized $\hat{s} \leftarrow (s - mean(\hat{S}))/var(\hat{S})$ over all states observed so far $\hat{S}$. A similar running variance over all rewards scales the rewards used for training. That is, the variance is computed from a running computation during training that is updated after every data collection step. The batch size used for PPO is 256, with a smaller batch size of 64 for the value function. The policy learning-rate and value function learning-rate are 0.0001 and 0.001, respectively. The value function is updated four times for every policy update. The NC also uses the Adam stochastic gradient optimization method [22] to train the ANN parameters. In all environments, we terminate the episode when more than half of the agents have fallen down. The target value for a fallen agent is set to zero.

5.2.2 LC. We train the LC using a similar learning system, where the network does not have a convolutional component and instead includes a bi-linear phase transform as the first layer [34]. The agent is trained to match motions from a database of stepping actions at different angles and distances by finding the proper sequence of actions in the form of PD controller target positions for each joint. Depending on the particular goal g_t^L the mocap motion that will result in the closest match to the footstep locations will be chosen.

5.3 Rewards, Environments, & Tasks

5.3.1 LC. The reward function used for this policy will encourage the agent to both match the motion capture and the desired footstep locations from the goal. We note that the source motion capture data is purposefully small. We train the LC on few samples of steps totalling less than minute of singular strides. Our hypothesis is, the combination of curriculum learning at the LC level and the proposed approach will lead to learning robust stepping and balance control from little data. To further improve robustness during crowded locomotion, we constructed a curriculum of simulated pushes and bumps. This curriculum is designed to simulate the types of interactions the agent will encounter in a crowded environment with other agents. This curriculum consists of applying temporary forces in random directions to the upper and lower body of the agent, including the shoulders, as well as hitting the character with projectiles.

5.3.2 NC. We construct a collection of physics-based simulation tasks to train and evaluate the proposed method within a rich physically-enabled RL environment [4]. At initialization, each agent is randomly rotated, and the initial velocities of the agent's links are determined by a selected motion capture clip using finite differences and rotated to align with the agent's reference frame. Goal locations g_i^H for the NC are randomly generated in locations that are at least 1 m away from any obstacle. Each agent is randomly placed in a

Name	Agents	Obstacles	Size	Task Type
Procedural	[3, 5]	[0, 10]	10×10 m	Cooperative
Bottleneck	[3, 5]	4	10×20 m	Cooperative
Pursuit	3	[0, 10]	10×10 m	Competitive

Table 1: Scenarios and their main parameters.

collision free starting space in the scene. The number and density of agents in the simulation vary depending on the task. The reward function used for each of the tasks is a combination of distance-based rewards $||pos(agent_i) - g_i^H||$, where $pos(agent_i)$ computes the location of agent i, and large positive reward for reaching a goal:

$$r_H = \begin{cases} 20 & \text{if } ||pos(agent_i) - g_i^H|| < 1 \\ exp(-1||pos(agent_i) - g_i^H||^2) & \text{otherwise.} \end{cases} \tag{7}$$

The sparse reward component places value on reaching goals as quickly as possible, while the continuous component helps with learning policies which steer toward goals. We note that predictive reciprocal collision avoidance policies emerge in training as high value approaches to maximizing the above rewards.

The following are descriptions of our physics-based training and testing environments. We summarize the technical details of the environments in Table 1.

Procedural This environment, shown in Figure 3a, represents the challenging task of articulated multi-agent navigation in an environment with other agents and procedurally generated obstacles.

Bottleneck In this environment, agents need to learn to cooperatively pass through the *Bottleneck* to avoid knocking each other over. This environment, shown in Figure 3b, represents the challenging task of articulated multi-agent navigation in density modulated environments.

Pursuit In this environment, one agent (agent 0, or the pursued agent) has the same navigation goal as in the *Procedural* environment. Two additional agents (pursuer 0,1) have the goal of chasing, or being as close to agent 0 as possible, shown in Figure 3c. The different goals of the agents result in a challenging, multi-agent competitive environment.

6 RESULTS

In this section, we demonstrate the efficacy of MAHRL from several perspectives. First, we review pitfalls of comparative crowds analysis with respect to the proposed method, and propose adequate baseline methods drawn from the state-of-the-art in similar problem spaces. We examine the performance of MAHRL in terms of learning, collisions, physical robustness, and strategy learning. Then we examine the performance in terms of computation cost and generalizability over the number of agents in the environment.

6.1 Baselines

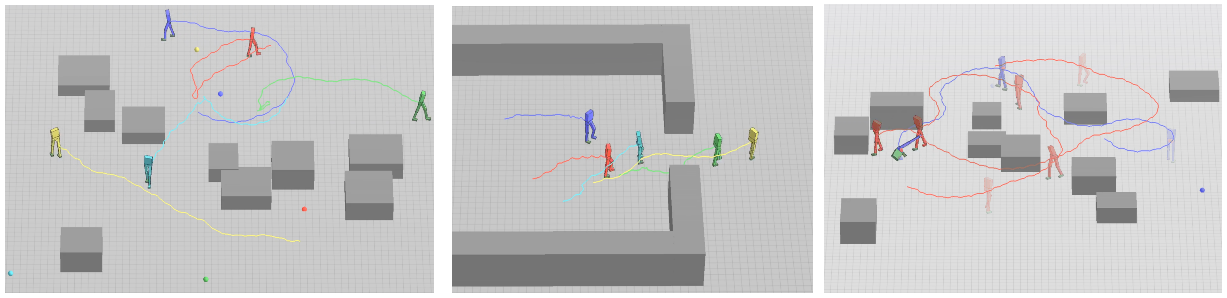
Comparative analysis with prior methods is difficult because the proposed method represents a new form of crowd simulation that has no baseline. This problem is illustrated in depth in Figure 4. Because of this we attempt to learn the problem we solve using other, state-of-the-art, methods in similar control problems.

To understand the performance of the proposed method, we compare the performance of MAHRL with respect to *Centralized* and *Decentralized* methods on the environments and tasks outlined in Section 5.3.2. Specifically, we compare MAHRL to Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [26], Nonhierarchical [38], and MAHRL using the TD3 method [10] in heterogeneous and homogeneous environments for the NC training as described in Section 5.2. Nonhierarchical in this paper refers to training a policy in a flat approach, i.e. without hierarchy, using out-of-the-box methods where locomotion and navigation behaviour are a single policy. MADDPG is an approach to training multi-agent navigation and behaviours where, during training, the value network of the deep learning model is shared among agents and observes all agent states. MADDPG has state-of-the-art results in the complex navigation domain and represents a partially *Centralized* approach to MARL. Additionally, we train the proposed method, MAHRL, with and without using TD3 to understand the sensitivity of the method to training technique, and the value of undervaluing policies during training. To evaluate the robustness of our method, we also evaluate two settings, homogeneous (*Homo*) and heterogeneous (*Hetero*) agents. In the *Homo* case, the agents share the same high level θ^i policy parameters. The *Hetero* case is the more common MARL setting with individual policies for each agent as seen in the right-hand side of Figure 2.

6.2 Learning Results

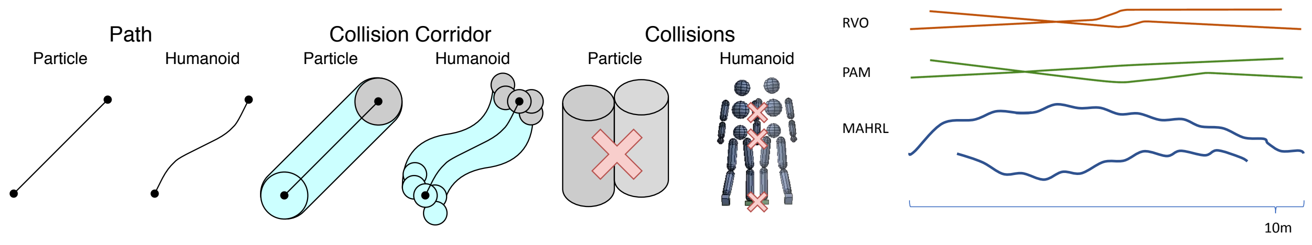
To understand the base performance of the proposed method, we evaluate the mean reward signal during training first. Figure 5a captures comparative training experiments showing the value of the proposed method with respect to mean reward and training steps. In the *Procedural* environment, MAHRL outperforms baselines and using TD3 further improves performance. The proposed method specifically maximizes reward in the heterogeneous environment where all agents are learning their own policies. The baseline, MADDPG, learns a policy good enough to locomote but not to learn coordinated behaviour, hence the low mean reward. This is likely related to the large Q-network needed for *Centralized* approaches that are a function of the number of agents. In the *Pursuit* environment, MAHRL outperforms MADDPG in terms of mean reward over iterations and quality of the policy. Agents learn not only to navigate but beneficial strategies for the environment begin to emerge. Qualitatively, throughout training, our method learns successful navigation strategies shown in Figure 3a as well as in Figure 9 on a full humanoid character (we later use this humanoid character to qualitatively evaluate physical robustness). The results for the *Decentralized* (Nonhierarchical) approach are not shown as the method failed to learn even simple standing behaviour.

We average the gradients on the input features for the learned value function over 16 rollouts, we show that MAHRL also learns an interesting bias in its value function, an estimate of the agent's future reward, which encourages agents to make right turns over left. This distance attenuated bias toward the rightward direction shows that MAHRL learns to value predictive reciprocal collision avoidance. There is a symmetric policy for left-side bias as well,



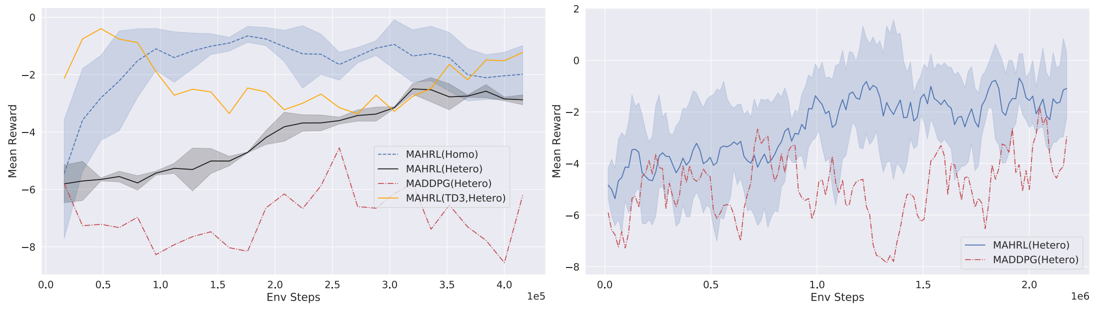
(a) *Procedural* (b) *Bottleneck* (c) *Pursuit*

Figure 3: (a) Agents reaching a series of targets in arbitrary environments. (b) Egress scenarios with a group of 5 agents. (c) Rasterized overlays from the pursuit environment, where the pursuer agents (red) learn to work together to the corner and tackle the navigating agent (blue).



(a) (b)

Figure 4: The proposed method is the first of its kind, fundamentally combining physical character control and crowd simulation. Because of this, primary methodologies in comparative crowds analysis have shortcomings which makes methods not representative as outline in (a). First the humanoid's centre of mass trajectory forms a piecewise parabolic curve unlike particle models where the same path is a straight line. Similarly, the underlying humanoid representation is composed of several capsule and sphere colliders, making the collision corridor and collision detection of the humanoid more complicated, whereas particle models can only detect collision within their particle bounds–typically a single circle or capsule. (b) compares the final trajectories of RVO[48], PAM [20], and MAHRL in a typical oncoming collision task between two agents.



(a) *Procedural* (b) *Pursuit*

Figure 5: Comparative studies of the learning curves of MAHRL, MAHRL (TD3), MADDPG, and Nonhierarchical for the *Procedural* environment and MAHRL & MADDPG for the pursuit environment. Nonhierarchical is not shown here as it did not make progress on any task. In the *Pursuit* environment, we compare the MAHRL with the state-of-the-art MADDPG approach. The proposed MAHRL approach outperforms across environments and learns well with heterogeneous agents, even with few training steps.

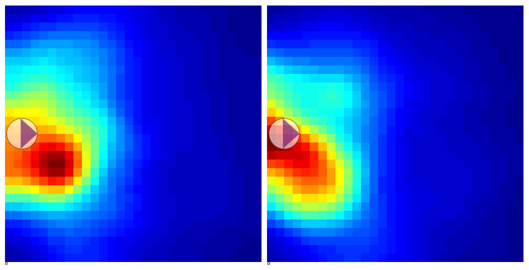

(a) ego field x (b) ego field y

Figure 6: Averaging the gradient magnitudes of the value network velcity field inputs reveal that the method learns to value an egocentric field (agent centred middle left and facing right) with a right side bias. This affords predictive reciprocal collision avoidance – a high value strategy in navigation.

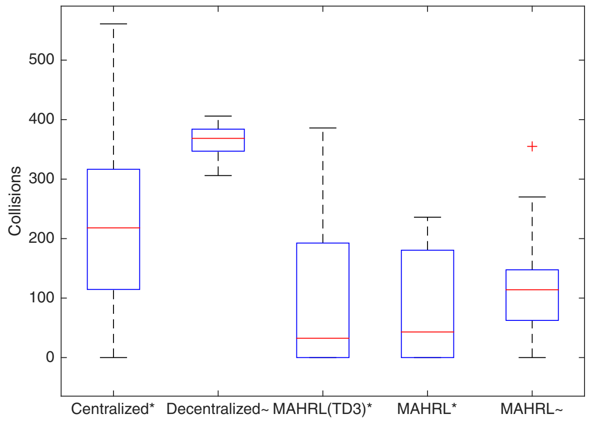


Figure 7: Comparative analysis of collisions counts across all baselines, MADDPG (*Centralized*), MAHRL with (*) and without (̃) heterogeneous agents, and Nonhierarchical (*Decentralized*) in the *Procedural* environment using a Kruskal-Wallis rank-sum test and post-hoc Conover's test with both FDR and Holm corrections for ties. MAHRL outperforms both MADDPG and Nonhierarchical methods in collision avoidance during steering and navigation.

and the selection of right over left here represents chance. Though bias could be purposefully introduced through rewards, this bias emerges through learning. This is shown in Figure 6a and Figure 6b.

6.2.1 Collision Analysis. To evaluate learned navigation policies quantitatively, we capture the mean number of collision events over all agents for each episode in several instantiations of the *Procedural* environment. Collisions are a common metric in synthetic crowds and navigation methodologies. Here, we extend the common definition of collision from overlaps of the agent disk model to physical collisions with body segment colliders. For each method, we perform 155 policy rollouts over several random seeds. The results are shown in Figure 7. A Kruskal-Wallis rank-sum test and post-hoc Conover's test with both False Discovery Rate (FDR) and Holm corrections for ties show the MAHRL methods significantly outperform others ($p < 0.01$). We can see that MAHRL produces fewer collisions than other methods.

6.2.2 Physical Robustness. To evaluate learned navigation policies qualitatively, we show that agents can successfully and continuously navigate complicated environments of forced interactions of the *Procedural* environment, as seen in Figure 3a [19]. Agents also learn tight packing behaviour in the *Bottleneck* environment as shown in Figure 3b. What is most interesting is that agents learn these environment and task specific behaviours, in addition to navigation and collision avoidance, when using only the rewards described in Section 5.3.2. While environment and goal conditioning drives the emergent policies, the reward signal is maximized when agents learn reciprocal collision avoidance and navigation policies.

To evaluate physical robustness, we show that humanoid character agents can handle bottleneck scenarios of increasing density. Starting with one agent and moving to 50 agents we show that the humanoids learn to successfully complete the scenario up until a critical point (> 0.35 agents/m^2 spawning). The results for 10, 20, and 50 agents can be seen in Figure 10. In the 50 agent scenario, the agents begin to experience critical stability failures, where physical interactions lead to tripping, falling, and eventually trampling. In this paper, as noted in Section 5.2 with respect to the LC, we are less interested in the fall animations themselves and more in the ability for the agents to learn robust *walking* policies. In our simulations, we leave the policy controller running when agents fall, hence the unnatural look post-fall. However, several additions could be used to handle falls including the addition of more motion capture data, ragdoll physics switching, get up controller/policy learning. Prior to adverse fall conditions, our full body agents are capable of staying upright even in the presence of crowded pushing, tripping, and stumbling. This level of physical fidelity is not possible with past multi-agent navigation methods, where these behaviours are often handled by a separate animation system.

6.2.3 Multi-Agent Games. From a training perspective, we note that quantitatively the three agents all begin to increase their average reward via their navigation objective using MAHRL. As learning progresses the pursuing agents outperform the pursued agent (agent 0), this can be seen in Figure 8. Qualitatively, as they get better, the pursued agent has an increasingly difficult time reaching its navigation targets while being chased. We show a rasterized version of an example episode from the *Pursuit* environment in Figure 3c, where the pursuer agents have learned to corner and tackle the pursued agent.

6.3 Computation and Generalization

In this section, we show two results, the computational costs of increasing the number of agents and the model generalization to increased numbers of agents. For two scenarios, *Procedural* and *Bottleneck*, the number of agents is increased, and we record the average reward and computational performance (defined as the amount of time it takes to perform the 16 rollouts (each rollout is 64 * 15 control steps) for training using a single thread). The agent-computation time curve in Figure 11a indicates a linear trend

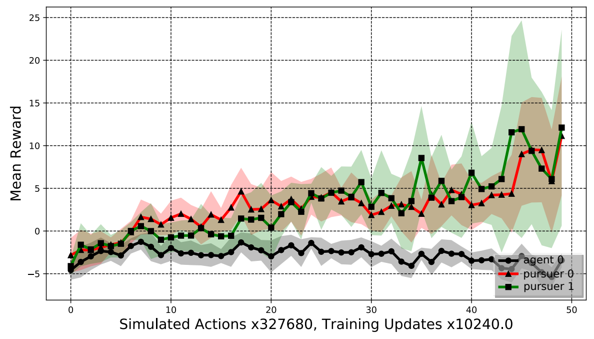


Figure 8: Learning curve for the 3 individual agents in the *Pursuit* simulation. The agent 0's ability to reach its goal reduces as the other pursuers improve at seeking agent 0.

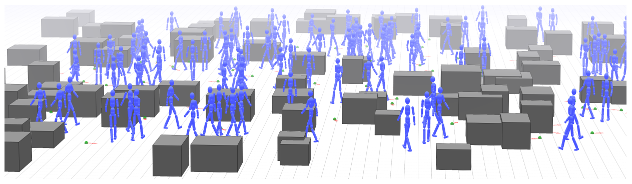


Figure 9: Large numbers of humanoid characters navigating the *Procedural* environment.

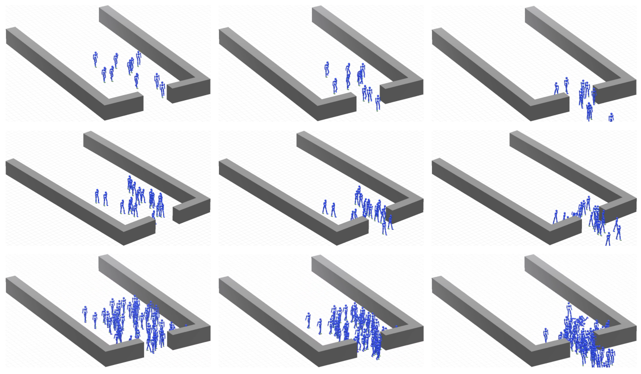
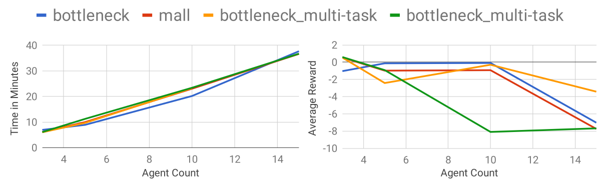


Figure 10: Increasing density in a bottleneck lead to robust physical interactions in top row: 10 agent, middle row: 20 agent, and bottom row: 50 agent humanoid egress scenarios. In critical scenarios (> 0.35 agents/m^2), physical interactions lead to tripping, falling, and even trampling.

in computational cost. While at agents' counts in the 20s the simulation is not real-time; the most computationally expensive part is not the learning system but the physics simulation.

The typical MARL framework is designed for a fixed number of agents. Here we show that our method provides some ability to generalize to different numbers of agents without additional training which also allows us to increase training efficiency by enabling training on fewer agents while being able to simulate with many more during test time. The average reward for two different types of policy training styles are compared. The first



(a) Computational Cost **(b) Agent num Generalization**

Figure 11: The performance of MAHRL from two quantitative perspectives, (a) the computational performance with respect to agent count and (b) the generalization performance with respect to average reward value. The yellow bottleneck_multi-task curve is a policy learned over multiple environments. The green multi-task curve is a single environment policy tested over multiple environments.

method trains on a single task at a time; the second method uses *multi-task* learning, training across all tasks at once, in hopes that a more generalizable, task-independent structure is acquired. The *multi-task* method, often preferring to optimize easier tasks, does not appear to learn more robust policies compared to the *Procedural* based method. All generalization results can be seen in Figure 11b. However, generalization remains a known and open issue of DRL methods [55].

7 CONCLUSION

The proposed approach represents the first model to produce fully articulated physical crowd agents. The evaluation of this approach shows how valuable it is for addressing the non-stationary learning problem of MARL in complex multi-agent navigation scenarios. We suspect methods such MAHRL will be useful in high fidelity interactions in gaming to produce more rich interactions with virtual characters and NPCs. In particular, virtual worlds may benefit from high fidelity physically-enabled and reactive crowds. As well, these methods can be applied in safety-critical analysis to drive rich dynamic analysis of dangerous situations where physical interactions are key indicators of safety failures.

While our method produces promising results, the work is limited by the fixed LC partial parameter sharing. Since this approach, while it mitigates the non-stationary problem, leaves the agents' locomotion skill-set homogeneous. There is room for research in the area of training the LC and NC concurrently. There is also room for broader LC skill-set training and richer shared action representations which may mitigate this problem. For the NC, we introduced a set of reward functions to encourage human-like behaviour while navigating with other agents. The literature motivates these rewards, but balancing them is its own challenge. In the future, it may be beneficial to use additional data-driven imitation terms to assist in learning from human-like paths. Finally, considerable effort was made to create combined locomotion, navigation, and behaviour controller that is robust to the number of agents in the simulation. However, robust generalization remains an open problem.

REFERENCES

[1] Brian Allen and Petros Faloutsos. 2009. Complex networks of simple neurons for bipedal locomotion. In *Proceedings of the IEEE/RSJ International Conference on*

Intelligent Robots and Systems. IEEE, 4457–4462.

[2] Brian Allen and Petros Faloutsos. 2009. Evolved controllers for simulated locomotion. In *Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*, Vol. 5884 LNCS. Springer, 219–230.

[3] Glen Berseth, Mubbasir Kapadia, and Petros Faloutsos. 2015. Robust space-time footsteps for agent-based steering. *Computer Animation and Virtual Worlds* (2015).

[4] Glen Berseth, Xue Bin Peng, and Michiel van de Panne. 2018. Terrain RL Simulator. *CoRR* abs/1804.06424 (2018). arXiv:1804.06424 http://arxiv.org/abs/1804.06424

[5] Hugo Bruggeman, Wendy Zosh, and William H Warren. 2007. Optic flow drives human visuo-locomotor adaptation. *Current biology* 17, 23 (2007), 2035–2040.

[6] Lucian Bu, Robert Babu, Bart De Schutter, et al. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172.

[7] Caroline Claus and Craig Boutilier. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI* 1998 (1998), 746–752.

[8] Alain Dutech, Olivier Buffet, and François Charpillet. 2001. Multi-agent systems by incremental gradient reinforcement learning. In *International Joint Conference on Artificial Intelligence*, Vol. 17. Citeseer, 833–838.

[9] Petros Faloutsos, Michiel Van de Panne, and Demetri Terzopoulos. 2001. Composable controllers for physics-based character animation. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 251–260.

[10] Scott Fujimoto, Herke Van Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477* (2018).

[11] Tao Geng, Bernd Porr, and Florentin Wörgötter. 2006. A reflexive neural network for dynamic biped walking control. *Neural Computation* 18, 5 (2006), 1156–96.

[12] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. 2018. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2255–2264.

[13] Dongge Han, Wendelin Boehmer, Michael Wooldridge, and Alex Rogers. 2019. Multi-Agent Hierarchical Reinforcement Learning with Dynamic Termination. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 2006–2008.

[14] Dirk Helbing, Illés Farkas, and Tamas Vicsek. 2000. Simulating dynamical features of escape panic. *Nature* 407, 6803 (2000), 487–490.

[15] Dirk Helbing and Peter Molnar. 1995. Social force model for pedestrian dynamics. *Physical review E* 51, 5 (1995), 4282.

[16] Rico Jonschkowski and Oliver Brock. 2015. Learning state representations with robotic priors. *Autonomous Robots* 39, 3 (2015), 407–428.

[17] L P Kaelbling. 1993. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Vol. vol.2. 1094 – 8.

[18] Mubbasir Kapadia, Nuria Pelechano, Jan Allbeck, and Norm Badler. 2015. Virtual crowds: Steps toward behavioral realism. *Synthesis lectures on visual computing: computer graphics, animation, computational photography, and imaging* 7, 4 (2015), 1–270.

[19] Mubbasir Kapadia, Matt Wang, Shawn Singh, Glenn Reinman, and Petros Faloutsos. 2011. Scenario space: characterizing coverage, quality, and failure of steering algorithms. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 53–62.

[20] Ioannis Karamouzas, Peter Heil, Pascal Van Beek, and Mark H Overmars. 2009. A predictive collision avoidance model for pedestrian simulation. In *International workshop on motion in games*. Springer, 41–52.

[21] Sujeong Kim, StephenJ. Guy, Karl Hillesland, Basim Zafar, AdnanAbdul-Aziz Gutub, and Dinesh Manocha. 2014. Velocity-based modeling of physical interactions in dense crowds. *The Visual Computer* (2014), 1–15. https://doi.org/10.1007/s00371-014-0946-1

[22] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[23] Andrew Kun and W. Thomas Miller III. 1996. Adaptive dynamic balance of a biped robot using neural networks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. pages. IEEE, 240–245.

[24] Jaedong Lee, Jungdam Won, and Jehee Lee. 2018. Crowd simulation by deep reinforcement learning. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*. ACM, 2.

[25] Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*. Elsevier, 157–163.

[26] Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems 30*. 6379–6390.

[27] Francisco Martinez-Gil, Miguel Lozano, and Fernando Fernández. 2015. Strategies for simulating pedestrian navigation with multiple reinforcement learning agents. *Autonomous Agents and Multi-Agent Systems* 29, 1 (2015), 98–130.

[28] Josh Merel, Arun Ahuja, Vu Pham, Saran Tunyasuvunakool, Siqi Liu, Dhruva Tirumala, Nicolas Heess, and Greg Wayne. 2018. Hierarchical visuomotor control of humanoids. *CoRR* abs/1811.09656 (2018). arXiv:1811.09656 http://arxiv.org/abs/1811.09656

[29] W. Thomas Miller III. 1994. Real-time neural network control of a biped walking robot. *Control Systems, IEEE* 14, 1 (1994), 41–48.

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[31] Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. 2003. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, Vol. 3. 705–711.

[32] OpenAI. 2018. OpenAI Five. https://blog.openai.com/openai-five/.

[33] Nuria Pelechano, Jan M Allbeck, Mubbasir Kapadia, and Norman I Badler. 2016. *Simulating heterogeneous crowds with interactive behaviors*. CRC Press.

[34] Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 41.

[35] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. 1889–1897.

[36] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2016. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR 2016)*.

[37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. 2017. Proximal Policy Optimization Algorithms. *ArXiv e-prints* (July 2017). arXiv:1707.06347 [cs.LG]

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[39] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *Proc. ICML*.

[40] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, and et al. 2017. Mastering the game of Go without human knowledge. *Nature* 550, 7676 (Oct 2017), 354–359.

[41] Shawn Singh, Mubbasir Kapadia, Glenn Reinman, and Petros Faloutsos. 2011. Footstep navigation for dynamic crowds. *Computer Animation and Virtual Worlds* 22, 2-3 (2011), 151–158.

[42] Gentaro Taga, Yoko Yamaguchi, and Hiroshi Shinizu. 1991. Self-organized control of bipedal locomotion by neural oscillators in unpredicatable environments. *Biological Cybernetics* 65, 3 (1991), 147–159.

[43] Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*. 330–337.

[44] Hongyao Tang, Jianye Hao, Tangjie Lv, Yingfeng Chen, Zongzhang Zhang, Hangtian Jia, Chunxu Ren, Yan Zheng, Changjie Fan, and Li Wang. 2018. Hierarchical deep multiagent reinforcement learning. *arXiv preprint arXiv:1809.09332* (2018).

[45] Daniel Thalmann and Soraia Raupp Musse. 2013. . Springer.

[46] Lisa Torrey. 2010. Crowd Simulation via Multi-agent Reinforcement Learning. In *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (Stanford, California, USA) *(AIIDE'10)*. AAAI Press, 89–94.

[47] Michael R Tucker, Jeremy Olivier, Anna Pagel, Hannes Bleuler, Mohamed Bouri, Olivier Lambercy, José del R Millán, Robert Riener, Heike Vallery, and Roger Gassert. 2015. Control strategies for active lower extremity prosthetics and orthotics: a review. *Journal of neuroengineering and rehabilitation* 12, 1 (2015), 1.

[48] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. 2011. Reciprocal n-body collision avoidance. In *Robotics research*. Springer, 3–19.

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

[50] William H Warren Jr, Bruce A Kay, Wendy D Zosh, Andrew P Duchon, and Stephanie Sahuc. 2001. Optic flow is used to control human walking. *Nature neuroscience* 4, 2 (2001), 213.

[51] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. 2015. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*. 2746–2754.

[52] David Wilkie, Jur Van Den Berg, and Dinesh Manocha. 2009. Generalized velocity obstacles. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5573–5578.

[53] KangKang Yin, Kevin Loken, and Michiel van de Panne. 2007. SIMBICON: Simple Biped Locomotion Control. *ACM Transactions on Graphics* 26, 3 (2007), Article 105.

[54] Petr Zaytsev, S Javad Hasaneini, and Andy Ruina. 2015. Two steps is enough: no need to plan far ahead for walking balance. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 6295–6300.

[55] Amy Zhang, Nicolas Ballas, and Joelle Pineau. 2018. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint arXiv:1806.07937* (2018).