# MULTI-AGENT HIERARCHICAL REINFORCEMENT LEARNING FOR HUMANOID NAVIGATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Multi-agent reinforcement learning is a particularly challenging problem. Current methods have made progress on cooperative and competitive environments with particle-based agents. Not enough progress has been made on solutions that could operate in the real world with interaction, dynamics, and humanoid navigation strategies. In this work, we make a significant step in multi-agent models on simulated humanoid navigation by combining Multi-Agent Reinforcement Learning with Hierarchical Reinforcement Learning. We build on top of foundational prior work in learning low-level physical controllers for locomotion and add a layer to learn decentralized policies for multi-agent goal-directed collision avoidance systems. Surprisingly, our results show that with this combination of methods common RL techniques can be used for finding heterogeneous policies. A video of our results on a multi-agent pursuit environment can be seen here[1].

## 1 INTRODUCTION

Multi-Agent Reinforcement Learning (MARL) is a very difficult problem space. Reinforcement learning methods rely heavily upon stationary environments wherein a learned policy can be improved by maximizing reward based on observations. With MARL, where agents learn in the shared environment, leads to agents making learning updates using data based on the previous dynamics of the environment. This is further complicated when the agents are not simple in their actions or representations. For example, physically enabled humanoid agents must learn balance control, navigation, and goal completion strategies. Yet if one goal of AI is to have agents that can operate in the real world they will need to reason over physical interaction amoung other agents.

Deep Reinforcement Learning (DRL) has been successful at solving complex planning tasks. Given a sizeable computational budget DRL has displayed superhuman performance on many games (Mnih et al., 2015; Silver et al., 2017). However, less progress has been made on the MARL problem space, possibly due to the non-stationary optimization of multiple changing policies which is not easily overcome by collecting more data (OpenAI, 2018). If the goal is to create agents that can operate in a dynamic multi-agent world, more stable methods with novel forms of communication are needed. The trend to make progress on MARL has been to simplify the optimization problem. For example, converting the multi-agent problem into a single agent centralized model results in large gains in performance but can increase the number of network parameters (Lowe et al., 2017). By using recurrent policies, significant compute, and constraints on the amount the policy is allowed to change, it is possible to beat the best humans at the multi-agent game of Dota (OpenAI, 2018). While these methods have shown promiss they have not yet displayed success in complex multi-agent environments. It has been possible to produce competitive behaviour using asymmetric self-play but this work is limited to few agents (Bansal et al., 2017). The driving question then is, how can we simplify the MARL optimization problem while maintaining the heterogeneous nature of multi-agent problems?

In this work, we propose the deep integration of MARL with Hierarchical Reinforcement Learning (HRL) to produce heterogeneous humanoid agents that can both navigate and interact in dynamic simulation. Specifically, we propose a method to reduce the difficulty in the MARL policy optimization problem. While previous methods have focused on re-framing the problem as a type

[1] https://sites.google.com/view/mahrl

of single-agent RL problem, our method preserves the important features of heterogeneous agent behaviour without adding more network parameters (Lowe et al., 2017). This use of HRL has many advantages over current methods. We use shared parameter methods but only for task-agnostic portions of the policy. Given the shared sub-policy, the optimization is simplified and allows us to learn complex multi-agent policies with significantly less data. This method represents, to the best of our knowledge, the first method for model-free multi-agent physical character control for locomotion and navigation.

## 2 Related Work

Simulated robot and physical character control is a rich area of research with many solutions and approaches. Neural models focused on training neural networks by receiving joint or body sensor feedback as input and producing appropriate joint angles as output (Geng et al., 2006; Kun & Miller III, 1996; Miller III, 1994). A biped character's movement controller set can also be manually composed using simple control strategies and feedback learning (Yin et al., 2007; Faloutsos et al., 2001). HRL has been proposed as a solution to handling many of the issues with current RL techniques that have trouble with long horizons and week signal. Many frameworks have been proposed for HRL but none seem to be the obvious choice for any particular problem (Sutton et al., 1999; Dayan & Hinton, 1993; Dietterich, 1999). One difficulty in HRL design is finding a reasonable communication representation to condition the lower level on. Some methods pretrain the lower level on a random distribution (Heess et al., 2016; Peng et al., 2017) and other learn a more constructive latent encoding (Nair et al., 2018; Eysenbach et al., 2019; Gupta et al., 2018). There is also the present challenge of learning multiple level of the hierarchy concurrently (Vezhnevets et al., 2017; Nachum et al., 2018; Levy et al., 2017). We use a goal based approach that uses footstep space representation that is randomly sampled to learn a task-agnostic lower level of control.

### 2.1 Multi-Agent Deep Reinforcement Learning

There are many types of multi-agent learning problems including cooperative, competitive, and with or without communication (Bu et al., 2008; Tan, 1993; Panait & Luke, 2005). Recent work converts the MARL problem to a single agent setting by using a single Q-function across all agents (Lowe et al., 2017). Additional work focuses on the problem of learning communication methods between agents (Foerster et al., 2016). While progress is being made, MARL is notoriously difficult due to the non-stationary optimization issue, even in the cooperative case (Claus & Boutilier, 1998). In this work, we apply a partial *parameter sharing* method assuming all agents are partly homogeneous at a tast-agnostic level and optimize similar goals (Gupta et al., 2017).

There exists few environments specifically created for MARL evaluation (Zheng et al., 2018; Suarez et al., 2019). The focus of these environments is often a type of strategy learning and cooperation (Tian et al., 2017; Vinyals et al., 2017). MARL is a growing area of research and as such will need increasingly complex environments to evaluate algorithms. In this work, we are interested in the overlapping problems of control and perception. To this end, we have created the first simulation of its type that affords multiple physics-based control tasks with variable numbers of agents. Recent work has begun to combine MARL and HRL but is limited to simple environments, uses additional methods to stabilize the optimization, and includes communication (Tang et al., 2018; Han et al., 2019).

Our work represents a new paradigm in learning within the multi-agent navigation domain–beyond simple navigation control strategies. We show compelling AI that learns navigation and gameplay strategies with fully articulated physical characters. This is achieved through a novel learning strategy that produces high value policies for a complicated control problem.

## 3 Policy Representation and Learning

In this section, we outline key details of the general Reinforcement Learning (RL) framework. RL 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 actions $a_t \in A$, sampled from a policy $\pi(s_t, a_t)$ which results in a new state $s_{t+1} \in S$ according to the transition

probability function $T(s_{t+1}|a_t, s_t)$. Performing action $a_t$ in state $s_t$ produces a reward $r_t$ from the environment; the expected future discounted reward from executing a policy $\pi$ is:

$$J(\pi) = \mathbb{E}_{r_0,\ldots,r_T}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \tag{1}$$

where $T$ is the maximum time horizon, and $\gamma$ is the discount factor, indicating the planning horizon length. The agent's goal is to optimize its policy, $\pi(\cdot|\theta_\pi)$, by maximizing $J(\pi)$. The policies in the work are trained using the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017). The value function is trained using TD($\lambda$).

### 3.1 Hierarchical Reinforcement Learning

In HRL the original MDP is split into two different MDPs that should be easier to solve seperately. In practice, this is accomplished by learning two different policies. The lower level policy is trained first and is often conditioned on a latent variable or goal $g$. The lower level policy $\pi^{lo}(a|s, g)$ is constructed in a way to give it temporally correlated behaviour depending on the $g$. After the lower level policy is trained it is used to help solve the original MDP using a seperate policy $\pi^{hi}(g|s)$. This policy learns to provide goals to the lower policy to maxmize rewards.

## 4 Multi-Agent Reinforcement Learning

The extension to the MDP framework for MARL is a partially observable Markov game (Littman, 1994). A Markov game has a collection of $N$ agents, each with its own set of actions $A_0, \ldots, A_N$ and $O_0, \ldots, O_N$ that are partial observations of the full state space $S$. Each agent $i$ has its own policy $\pi(a|o_i, \theta_i)$ that models the probability of selecting an action. The environment transition function is a function of the full state and each agent's actions $T(S'|S, A_0, \ldots, A_N)$. Each agent $i$ receives a reward $r_i$ for taking a particular action given a partial observation $o_i$ and its objective is to maximize this reward over time $\sum_{t=0}^{T} \gamma^t r_i^t$ , where $\gamma$ is the discount factor and $T$ is the time horizon. The policy gradient can be computed for each agent as

$$\nabla_{\theta_i} J(\pi(\cdot|\theta_i)) = \int_{O_i} d_{\theta_i}(o_i) \int_{A_i} \nabla_{\theta_i} \log(\pi(a_i|o_i, \theta_i)) A_\pi(o_i, a_i)\, da_i\, do_i \tag{2}$$

where $d_\theta = \int_S \sum_{t=0}^{T} \gamma^t p_0(o_0)(o_0 \to o \mid t, \pi_0)\, do_0$ is the discounted state distribution, $p_0(o)$ represents the initial state distribution, and $p_0(o_0)(o_0 \to o \mid t, \pi_0)$ models the likelihood of reaching state $s$ by starting at state $o_0$ and following the policy $\pi(a, o|\theta_\pi)$ for $T$ steps (Silver et al., 2014). Here $A_\pi(o, a)$ represents the advantage function estimator GAE($\lambda$) in (Schulman et al., 2016).

The challenge in MARL is that each agent learns separately and this often results in a non-stationary learning problem. Each agent is learning how to estimate the dynamics and advantage of an environment that is affected by other evolving agents. Data collected and used for learning becomes inaccurate after any other agent's training update. We propose a method for reducing the inaccuracies between updates by using HRL.

### 4.1 Multi-Agent Hierarchical Reinforcement Learning

We construct a multi-agent learning structure that takes advantage of hierarchical design, or Multi-Agent Hierarchical Reinforcement Learning (MAHRL). Each agent has its own higher level policy (Multi-Agent Navigation Controller (NC)) $\pi(g|o, \theta_i^{hi})$ and a shared lower level policy (Locomotion Controller (LC)) $\pi(a|o, g, \theta^{lo})$. This method allows us to introduce more structure into the difficlut multi-agent optimization problem. We will use the notation $\pi(a_i|o_i, \theta_i^{hi}, \theta^{lo})$ to denote the policy induced by the pair of policies. While current research shows that it is challenging but possible to train a two-level hierarchy concurrently (Nachum et al., 2018; **?**), we instead pretrain the lower level policy and leave it fixed and shared across all agents. This reduces the MARL problem from learning the details of locomotion via joint torques for each agent to learning goal-based footstep plans for each agent.

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

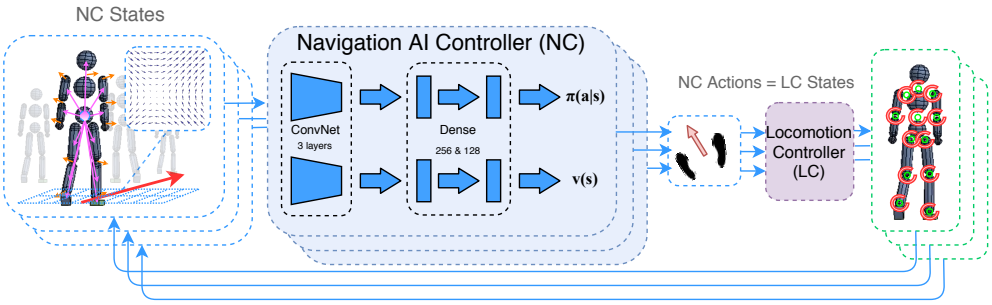



Figure 1: An overview of the MAHRL approach. From left to right: the NC state includes the relative goal position and distance, an egocentric velocity field capturing the relative velocity of obstacles and other agents, and the physical character link positions and tangent states; for each agent this state is input to a multi layer CNN, including two dense hidden layers, and outputs actions–the value function uses a similar network. These high-level actions are in the form of two-step plans dictating future foot placements; these footstep plans are consumed by the LC as $g$, which produces the angle-axis targets for per-joint PD controllers.

structure significantly reduces the non-stationarity. Not only is each agent sharing a portion of its network parameters with each other agent, this portion has been carefully constructed to provide structured exploration for the task–greatly reducing the number of network parameters that need to be learned and transforming the action space. We extend the analysis of this work by combining the PPO algorithm in a MARL *partial parameter sharing* setting. Details of the algorithm can be found in the supplementary material. This modification controls the way $d_{\theta_i}(o_i)$ can change for each agent making it easier for each agent to estimate other agents potential changes in behaviour.

## 5 Learning Hierarchical Multi-Agent Navigation Control

To solve the hierarchical learning problem, we train in a bottom-up fashion sharing the LC policy among heterogenous NC policies. To do this, the levels of the hierarchy communicate through shared actions and state in a feedback loop that is built to reflect biological biomechanical locomotion system in humans. The NC's objective is to provide footstep placement goals $a_H = g_L$ for the LC as seen in the right hand side of Figure 1. These footstep goals are produced as two-step plans. Each step is parameterized with its root relative placement, angle on the ground, centre of mass heading vector, and a time signature. The NC is queried for a new footstep action every 0.5 s. The NC decides what action to take based on the egocentric velocity field $E$ in front of the agent, its pose $\mathbf{s}_{char}$ and the NC goal $g_H$, $C = \{E, \mathbf{s}_{char}, g_H\}$, seen in the left box of Figure 1.

### 5.1 State Space

In several studies, it has been shown that optic flow, an inherently egocentric phenomenon of mapping velocities to regions of vision, is key to sighted locomotion (Bruggeman et al., 2007; Warren Jr et al., 2001). Additionally, the field of vision directly impacts walking ability and locomotion stability (Jansen et al., 2011). Taken together, vision's role in locomotion forms an egocentric velocity field where perceived distance and movement play different roles in locomotion control (Turano et al., 2005). This evidence has been used previously to define multi-agent navigation models, both by constructing a discretized egocentric field (Kapadia et al., 2009) and by learning the discretization of an egocentric field (Long et al., 2017).

The NC uses as input an egocentric velocity field relative to the agent's location and rotation. This egocentric velocity field $E$ is $32 \times 32$ over a space of 5x5 m, starting $0.5$ m behind the agent and extending $4.5$ m in front, shown in Figure 1(a). The velocity field consist of two image channels in the x and y direction of a square area directly in front of the agent, where each sample is calculated as the velocity relative to the agent (Bruggeman et al., 2007; Warren Jr et al., 2001). 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 distance to the current spatial goal location. Both the egocentric velocity field sampling and the goal location state features can be seen in the left box of Figure 1.

### 5.2 REWARD

Navigation combines the desire to move towards goals, while avoiding collisions with other objects, in an energy efficient way. The methods purpose is to implicitly learn low effort local turning behaviour while avoiding collisions without the need to explicitly describe its operation. To elicit this behaviour, a combination of reward signals are used. Primarily, a dense reward is used to encourage the agent to walk in the direction of its current goal.

$$r_{Hd} = \exp(-(\min(0, (u_{tar} * v_{com}) - v_{com}))^2) \tag{3}$$

where $v_{com}$ is the agent's velocity and $u_{tar}^T$ is a normalized vector in the direction of the goal. For this work, a desired speed of $v_{com} = 1.0m/s$ is used.

A directional reward is not enough to encourage the agent to proactively reach its goal. To reinforce the importance of goal reaching behaviour, a large reward $r_{goal}$ for reaching the goal is added–being within $\sqrt{2}$ m. In this work, the value $r_{goal} = 20$ is used and comes from $horizon = 1/(1-\gamma)$ and the maximum reward the agent can otherwise receive is $max(r_d) = 1$, a reward that will be more important than travelling to the goal should be at least $horizon * max(r_d)$. This reward greatly increases the goal reaching behaviour. However, it can have the unintended effect of making the agent seek its goal aggressively, by trampling other agents. To reduce this behaviour, a repulsive cost was added when the agent is within $\sqrt{3}$ m other agents

$$r_{Ha}(a) = \sum_{a' \in \{A-a\}} -(r_s + (l - dist(a, a')) \tag{4}$$

where $A$ is the set of agents in the simulation, and $dist(a, a')$ computes the Euclidean distance between the Centre of Mass (COM) of agents $a$ and $a'$. We empirically found $r_s = 2.5$ and $l = 3.0$ work well for defining a distance dependent penalty in this case. This is intended to roughly approximate the power law of pedestrian interactions (Karamouzas et al., 2014). A similar repulsive cost is applied between the agent $a$ and obstacles $OB$ in the scene $r_{Hb}(a) = \sum_{ob \in OB} -2.5$ for each obstacle within $1$ m of $a$. The high reward for reaching the goal makes the agent very single-minded. The obstacle penalty is introduced to prevent the agents from using obstacles as affordances to regain balance.

Many RL simulation environments use a flag to indicate the episode or simulation end. This indicates that either the agent has reach its time limit or the agent has entered into an unrecoverable area of the state space, such as a fall. With multiple agents, the likelihood of a fall is high. It is not clear what to do when there are multiple agents being simulated. Terminating early, when one agent has fallen, is sub-optimal for other agents doing well, and waiting for every agent to fall wastes compute resources while most agents are collecting unhelpful data. We chose to reset the simulation when more than half the agents have fallen. However, fallen agents continue to act and need to be heavily penalized, so they receive a fixed reward of $-5$–lower than any other reward.

The NC's final reward function is a combination of the task rewards and behavioural costs:

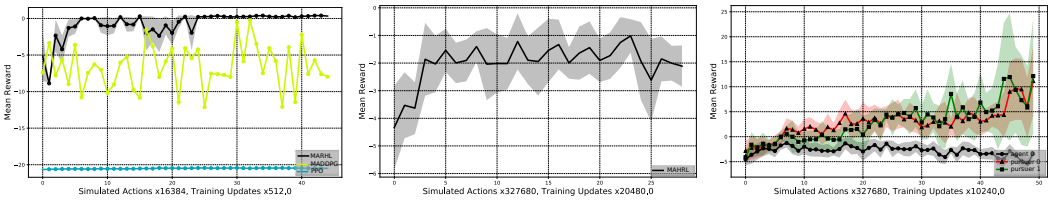
$$r_H(a) = \begin{cases} -5 & \text{if } fallen \\ r_{Hd} + r_{Ha}(a) + r_{Hb}(a) & \text{otherwise.} \end{cases} \tag{5}$$

### 5.3 SIMULATION ENVIRONMENT & TRAINING

We construct a collection of physics-based simulation tasks to evaluate the method on. 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 are randomly generated in locations that are at least $1$ m away from any obstacle. Each agent is randomly placed in the scene such that it does not intersect with any other agent or obstacle. The number and density of agents in the simulation vary depending on the task. For training, we found that starting with 3 agents in the environment is a good trade-off between computation cost and the generalization ability of the resulting learned policy. Environment specifics are given in Table 1. We consider the simulation and training environment to be, to the best of our knowledge, another novel contribution. While some simulators exist that support physics-based simulation for robots (Brockman et al., 2016; Tassa et al., 2018), few support more than one agent, with at most 2. Other libraries focus on supporting different kinds of MARL configurations for particle-based

| Name | agent count | obstacle count | size | agent direction | obs location |
|---|---|---|---|---|---|
| waypoint | 1 | $[0, 10]$ | $10 \times 10$ m | random | random |
| oncoming | $[2, 10]$ | $[0, 2]$ | $8 \times 8$ m | towards circle centre $+\mathcal{N}(0, 0.15)$ | circle centre |
| mall | $[3, 5]$ | $[0, 10]$ | $10 \times 10$ m | random | random |
| bottleneck | $[3, 5]$ | 4 | $10 \times 20$ m | right $+\mathcal{N}(0, 0.15)$ | around |

Table 1: Scenarios and their main parameters.




(a) Baseline Comparison (b) Humanoid3d on Mall (c) Pursuit

Figure 2: Comparative study of the learning curves of (a) PPO and MAHRL and (b) each agent in the 5 agent *mall* environment. Comparative study of the learning curves of each agent using MAHRL in the 3 agent pursuit enviroment.

agents (Lowe et al., 2017). Our proposed approach represents the first physics-based simulation of its kind that supports MARL.

The NC uses convolutional layers followed by dense layers. The particular network used is as follows: 16 convolutional filters of size $6 \times 6$ and stride $2 \times 2$, 16 convolutional filters of size $3 \times 3$ and stride $1 \times 1$, 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 are standardized with respect to all states observed so far. The rewards are scaled by a running variance. 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 0.000 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 (Kingma & Ba, 2014) to train the Artificial Neural Network (ANN) parameters.

## 6 Results

In this section, we demonstrate the efficacy of the proposed method. We separate our evaluation into two sections. First, we examine the performance of the model in terms of training and learned policies. Second, we examine the performance of our arpproach in terms of computation cost and generalizability with respect to the number of agents in the environment.

### 6.1 Learning

We evaluate MAHRL in complex multi-agent environments by first examining the performance of the reward function over training episodes. We show that MAHRL performs much better than the basic PPO algorithm without any hierarchical structure in the 5-agent *mall* in Figure 2(a). After a lengthy training session the basic PPO is not able to even produce a standing behaviour. We find that MAHRL performs well and is able to learn navigation behaviour after a short amount of training. We find that the hierarchical design provides a significant improvement in this case. We also compare MADDPG and find that it does better than PPO but strugles to produce good behaviour. We beleive this is related to the large Q-network that needs to be created for MADDPG. In Figure 2(b) we show the learning curves training MAHRL with a full *humanoid* model. Even with this increased control complexity out method is able to learn succesful navigation strategies shown in Figure 5. Each agent in the scenario quickly develops strong navigation behaviours that become more conservative

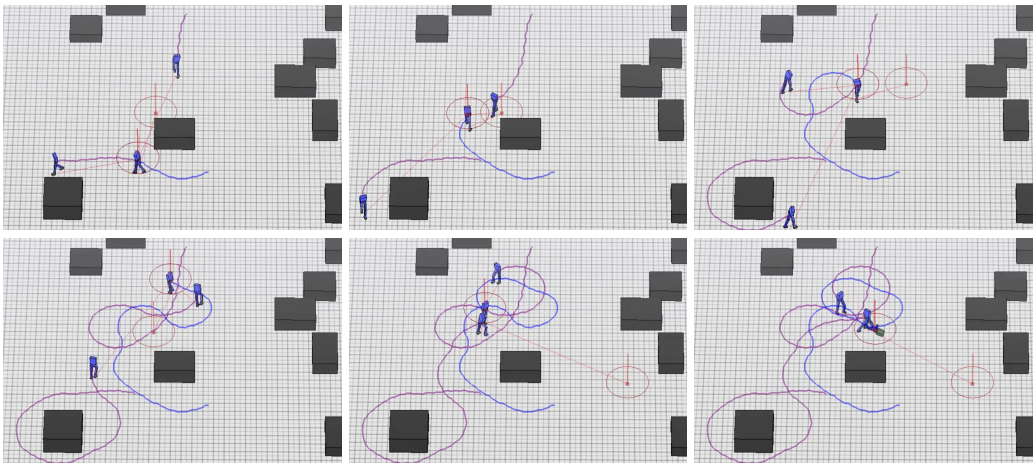


Figure 4: Rasterized images from the pursuit environment, where the pursuer agents learn to work together to corner and tackle the navigating agent (middle agent). A video for this example can be found here.

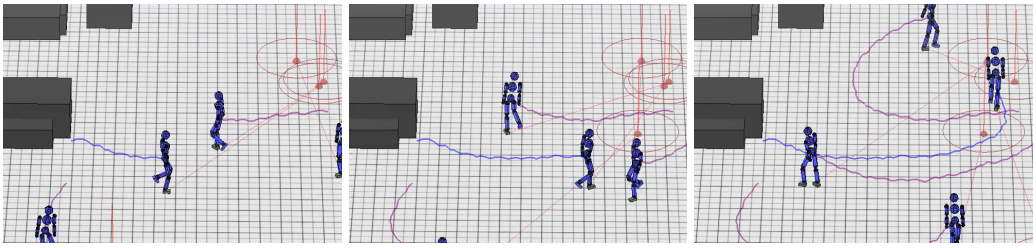
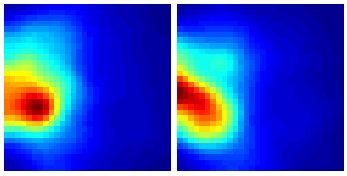


Figure 5: Rasterized images from the mall environment with humanoid agents navigating and avoiding each other while seeking goals. A video for this example can be found here.

over time as agents value avoiding collisions. This can result in agents taking longer paths in the environment.

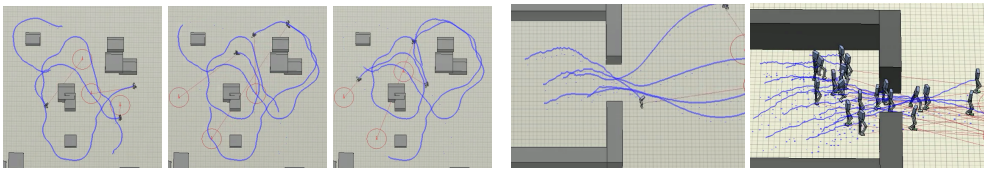
By extracting the gradients on the input features for the value function in the learned model we can examine some artifacts of what is learned. Recall, that for input we include in the state of the NC a simple model of an egocentric perceptual field–a square region in front of the agent. We show that our models learns two important aspects of navigation with respect to this field. The magnitude of the velocity field gradients in our learned model reveal that the learning process has developed an egocentric velocity field attenuated with distance. This can be seen in Figure 3a. Interestingly, this field is biased toward the rightward direction (down in the figures). This bias supports reciprocal collision avoidance in counter flows.



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

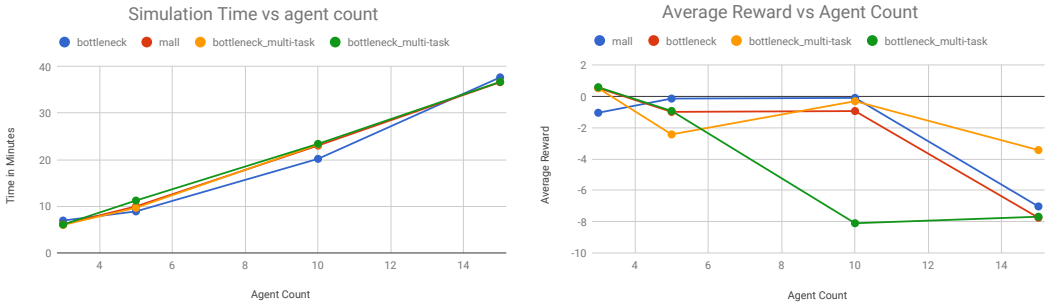
To further evaluate MAHRL, it is applied to a *pursuit* environment. In this environment there is one agent (agent 0) with the same navigation goal as in previous environments. As well, 2 additional agents (pursuer 0,1) that have the goal of reaching agent 0. This is accomplished by setting a high-level goal $g_H$ for the pursuers to the location of agent 0. In Figure 2(c), the three agents all begin to increase their average reward via their navigation objective, as learning progresses the pursuing agents outperform agent 0 and as they get better agent 0 has an increasingly difficult time reaching its own navigation targets. An example of this behaviour is shown in Figure 4.

To evaluate the NC qualitatively we show that agents learn to navigate arbitrary environments while avoiding collisions with both obstacles and other agents. First, we show a rasterized version of an example episode from the *pursuit* environment in Figure 4 where agents learn to corner and tackle. Then, we show that agents can successfully and continuously navigate complicated environments of



(a) 5 Agent Mall (b) BottleNeck

Figure 6: (a) Agents reaching series of targets in arbitrary environments (images in raster order). (b) Egress scenarios with a group of (left) 5 and (right) 21 agents. The density of the second group results in a physics-based bumping, pushing, and trampling.




(a) (b)

Figure 7: The performance of the method from two quantitative perspectives, (a) the computational performance with respect to agent count and (b) the generalization performance with respect to average reward value.

forced interactions as seen in Figure 6a (Kapadia et al., 2011). Finally, robust clogging and trampling is shown in both low and high density bottleneck egress scenarios respectively Figure 6b.

### 6.2 Computation and Generalization

The deep integration of physical character control and distributed multi-agent navigation comes with a cost directly dependent on the number of active agents. In this section, we show two results in the same experiment, the computational costs of increasing the number of agents and the model generalization to more difficult scenarios. For two scenarios, mall and bottleneck, the number of agents is increased, and we record the average reward and computation time from the simulation. The agent-computation curve in Figure 7(a) indicates a linear trend in computational cost. While at agents counts in the 20s the simulation is not real-time, the most computational expensive part is not the learning system but the physics simulation. Computationally efficient articulated body simulation is an active area of research (Erez et al., 2015). For accurate and stable simulation we use a physics time-step of $0.0005$ s.

To evaluate the learned policies ability to generalize with group size, we vary the number of agents for select tasks after a homogenious policy has been trained. The longer the policy can maintain a high average reward the better the generalization. In addition, the average reward for two different types of policy training styles are compared. The first method trains on a single task at a time, the second method uses *multi-task* learning in hopes that a more generalizable task-independent structure is aquired. The *multi-task* method, often preferring to optimize easier tasks, does not appear to learn more robust policies compared to the scenario-space based method (Kapadia et al., 2011). All generalizaiton results can be seen in Figure 7(b). However, generalization remains a known and open issue of DRL methods (Zhang et al., 2018)

## 7 Conclusion

In this paper we present a novel method where, for the first time, multi-agent navigation and physical character control are integrated. Multiple heterogenious interacting agents can experience physical interactions, handle physical perturbations, and produce robust physical phenomenon like falling

and slipping. To achieve this, we developed an integrated model of MARL and HRL with *partial parameter sharing*. The evaluation of this approach shows how valuable it is for addressing the non-stationary learning problem of MARL in complex multi-agent scenarios. In particular, the heterogeneous pursuer scenario learns complex routing and tackling behaviours akin to state-of-the-art competitive self-play approaches.

While our method produces promising results, the work is limited by the fixed LC partial parameter sharing. There is room for research in the area of training the LC and NC concurrently. Still, the proposed method benefits greatly from this stability. For the NC, we introduced a set of reward functions to encourage human-like behaviour while navigating with other agents. These rewards are motivated from the literature but balancing them is its own challenge. In the future, it may be beneficial to use additional data-driven imitation terms to encourage human-like paths. Finally, great effort was made to create an agent that is robust to the number of agents in the simulation. However, robust generalization remains an open problem.

## REFERENCES

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017. URL http://arxiv.org/abs/1710.03748.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

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

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

Bullet. Bullet physics library, 2015. http://bulletphysics.org.

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

Peter Dayan and Geoffrey E Hinton. *Feudal Reinforcement Learning*, pp. 271278. Morgan-Kaufmann, 1993. URL http://papers.nips.cc/paper/714-feudal-reinforcement-learning.pdf.

Thomas G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *arXiv:cs/9905014*, May 1999. URL http://arxiv.org/abs/cs/9905014. arXiv: cs/9905014.

Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 4397–4404. IEEE, 2015.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

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

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2137–2145, 2016.

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

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 5302–5311. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7776-meta-reinforcement-learning-of-structured-exploration-strategies.pdf.

Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In Gita Sukthankar and Juan A. Rodriguez-Aguilar (eds.), *Autonomous Agents and Multiagent Systems*, pp. 66–83, Cham, 2017. Springer International Publishing. ISBN 978-3-319-71682-4.

Dongge Han, Wendelin Boehmer, Michael Wooldridge, and Alex Rogers. Multi-agent hierarchical reinforcement learning with dynamic termination. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2006–2008. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Sander EM Jansen, Alexander Toet, and Peter J Werkhoven. Human locomotion through a multiple obstacle environment: strategy changes as a result of visual field limitation. *Experimental brain research*, 212(3):449–456, 2011.

Mubbasir Kapadia, Shawn Singh, William Hewlett, and Petros Faloutsos. Egocentric affordance fields in pedestrian steering. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pp. 215–223. ACM, 2009.

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

Ioannis Karamouzas, Brian Skinner, and Stephen J Guy. Universal power law governing pedestrian interactions. *Physical review letters*, 113(23):238701, 2014.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. Learning multi-level hierarchies with hindsight. *arXiv:1712.00948 [cs]*, Dec 2017. URL http://arxiv.org/abs/1712.00948. arXiv:1712.00948.

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

Pinxin Long, Wenxi Liu, and Jia Pan. Deep-learned collision avoidance policy for distributed multiagent navigation. *IEEE Robotics and Automation Letters*, 2(2):656–663, 2017.

Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6379–6390. Curran Associates, Inc., 2017.

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

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. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *arXiv:1805.08296 [cs, stat]*, May 2018. URL http://arxiv.org/abs/1805.08296. arXiv: 1805.08296.

Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *arXiv:1807.04742 [cs, stat]*, Jul 2018. URL http://arxiv.org/abs/1807.04742. arXiv: 1807.04742.

OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.

Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

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

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *ArXiv e-prints*, July 2017.

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

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

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, and et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354359, Oct 2017. ISSN 1476-4687. doi: 10.1038/nature24270.

Joseph Suarez, Yilun Du, Phillip Isola, and Igor Mordatch. Neural MMO: A massively multiagent game environment for training and evaluating intelligent agents. *CoRR*, abs/1903.00784, 2019.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181211, Aug 1999. ISSN 0004-3702. doi: 10.1016/S0004-3702(99)00052-1.

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

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

Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller. DeepMind Control Suite. *ArXiv e-prints*, January 2018.

Yuandong Tian, Qucheng Gong, Wenling Shang, Yuxin Wu, and C. Lawrence Zitnick. Elf: An extensive, lightweight and flexible research platform for real-time strategy games. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2659–2669. Curran Associates, Inc., 2017.

Kathleen A Turano, Dylan Yu, Lei Hao, and John C Hicks. Optic-flow and egocentric-direction strategies in walking: Central vs peripheral visual field. *Vision research*, 45(25-26):3117–3132, 2005.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *arXiv:1703.01161 [cs]*, Mar 2017. URL http://arxiv.org/abs/1703.01161. arXiv: 1703.01161.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy P. Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. Starcraft II: A new challenge for reinforcement learning. *CoRR*, abs/1708.04782, 2017.

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

KangKang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics*, 26(3):Article 105, 2007.

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

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

Lianmin Zheng, Jiacheng Yang, Han Cai, Ming Zhou, Weinan Zhang, Jun Wang, and Yong Yu. Magent: A many-agent reinforcement learning platform for artificial collective intelligence. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

# 8 Supplementary Material

## 8.1 Locomotion Controller

The Low-Level Controller (LLC) is designed to solve the problem of coordinating joint torques over time to mimic selected motion capture data and facilitate robust balance and locomotion. The LLC is queried at a frequency of $30$ Hz to produce joint target angles for per-joint PD controllers. The details of this approach are outlined in the following sections.

## 8.2 Reward

We found that a combination of imitation and end effector rewards leads to a robust and reactive motion. The goal is to reward joint angle targets which result in mimicking source motions. The combination of these competing rewards leads to a complex balancing problem of many objectives–trading off imitation behaviour for footstep matching.

### 8.2.1 Imitation

The imitation method is predicated on time indexed $0.5$ s long reference motions $q_i(t)$ created from motion capture data and augmented with footstep parameterized labels. The difference between the agent and the desired pose to imitate is computed as the weighted difference between the time indexed pose and the agent $r_{Lp} = \exp(-0.5 \sum_i ||\mathbf{s}^i_{char} - q^i(t)||^2)$. The motion database contains $7$ s of segmented motion that coarsely covers the space of possible footstep positions and the motions necessary to realize those footsteps. The accuracy and diversity of the LLC's motion can be increased or made more robust with additional motion if desired, including stylized, disordered, running, and more.

### 8.2.2 Footstep goal

The footstep goals $g_L = \{\hat{p}_0, \hat{p}_1, \hat{\theta}_{root}\}$ in Figure 1(c), 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 Zaytsev et al. (2015).

The LLC is designed to learn to place its feet as accurately as it can to match these step goals using the reward $r_{Lg} = \exp(-0.2||\mathbf{s}^g_{char} - g_L||^2)$. The better the LLC learns this behaviour, the stronger

the communication channel between it and the High-Level Controller (HLC) will be. However, if the agent matches these footstep goals too well, the smoothness of the motion will degrade. A balance is struck between learning robust footstep placement strategies and perfect mimicry–the reference motions and foot placements guide naturalness and should not be over-fitted.

#### 8.2.3 END EFFECTOR

An additional reward is given to encourage the policy to match the end-effector positions in the imitation motion $r_{Le} = \exp(-0.15 \sum_e ||\mathbf{s}^e_{char} - q^e(t)||^2)$. Reward is also given for how well the agent matches the imitated motion's COM $r_{Lcom} = \exp(-0.2 \sum ||\mathbf{s}^{com}_{char} - q^{com}(t)||^2)$

#### 8.2.4 TORQUE PENALTY

To smooth the actions generated from the control policy, a reward for maximizing the above rewards with minimal torque cost is used $r_{L\tau} = \exp(-0.2 \sum_i \tau(\mathbf{s}^i_{char}))$. These torques are normalized by specified torque limits that keep the agent from displaying unrealistic strength.

### 8.3 TRAINING & SIMULATION ENVIRONMENT

The LLC state includes the proprioceptive-like joint information. In particular, the components of $\mathbf{s}_{char}$ consist of the COM relative distances and velocities of each links as well as the rotation and angular velocities of each link as shown in Figure 1(d). The trained policy actions are in the form of angle-axis targets for per-joint Proportional Derivative (PD) controllers. In this way, the action is used to set the desired position of each joint of the agent. This can be seen in Figure 1(e). The method is also able to learn policies for agents of different types. In Figure **??** we additionally show the method using a full humanoid LLC character.

The agent is simulated with the physics-based rigid body simulation environment Bullet Bullet (2015). The policy is trained in an on-policy fashion. At the beginning of each episode a goal $g_L$ is sampled from the simulation and updated every $0.5$ s. Episodes end when either a time limit $T$ is reached or the agent falls. Between training rounds $4096$ transition tuples are collected by simulating episodes in parallel. After enough tuples have been collected $16$ minibatch updates are performed with a batch size of $256$. The value function is updated $64$ with a minibatch size of $64$. For the policy and value function. The policy and value function are both modeled using a ANN with $2$ hidden layers of size $512$ and $256$. The network uses ReLU activations through except for after the last layer where a *tanh* activation that outputs values between $[-1, 1]$ is used. The Adam stochastic gradient optimization method is used to train the ANN parameters Kingma & Ba (2014). The learning rates for the policy and value function are $0.000$ and $0.001$ respectively. Details related to the learning algorithm can be found in the supplementary material.

Many previous methods have created robust controllers via imitation. These methods were not intended to be used in crowds or dynamic environments with potentially random disturbances from other interacting agents in the simulation. Given that we intend to use this agent in a physical multi-agent simulation where the agent may bump into other agents or obstacles when trained in a crowded setting, we apply additional methods to simulate pushes that may be encountered. That is, random pushes between $50 - 150$ N are applied every $3 - 5$ s for a duration of $0.1 - 0.3$ s to increase the robustness of the controller. Similarly, the motions in the motion database start from the same facing direction. To make the agent more robust to distributed crowd scenarios, the agent's initial facing direction is randomized during training.

### 8.4 LEARNING DETAILS

The modification to this algorithm for the MARL method is to change line $7-8$ to execute an action for each agent in the simulation in parallel.

The Inputs are standardized wrt to all states seen so far. The rewards are divided by the variance. The variance is computed from a running computation during training that is updated after every data collection step. The advantage is batch normed.

**Algorithm 1** Goal-Based Learning Algorithm

Randomly initialize model parameters $\theta_\pi$ and $\theta_v$
**while** not done **do**
  **for** $i \in \{0, \dots N\}$ **do**
    $\tau_i \leftarrow \{\}$
    **for** $t \in \{0, \dots, T\}$ **do**
      $a_t \leftarrow \pi(\cdot|s_t, \theta_\pi)$
      $s_{t+1}, r_t \leftarrow$ execute $a_t$ in environment
      $\tau_{i,t} \leftarrow < s_t, a_t, r_t >$
      $s_t \leftarrow s_{t+1}$
    **end for**
  **end for**
  Update value function $V_\pi(\cdot)$ parameters $\theta_v$ using $\{\tau_0, \dots, \tau_N\}$
  Update policy parameters $\theta_\pi$ using $\{\tau_0, \dots, \tau_N\}$
**end while**

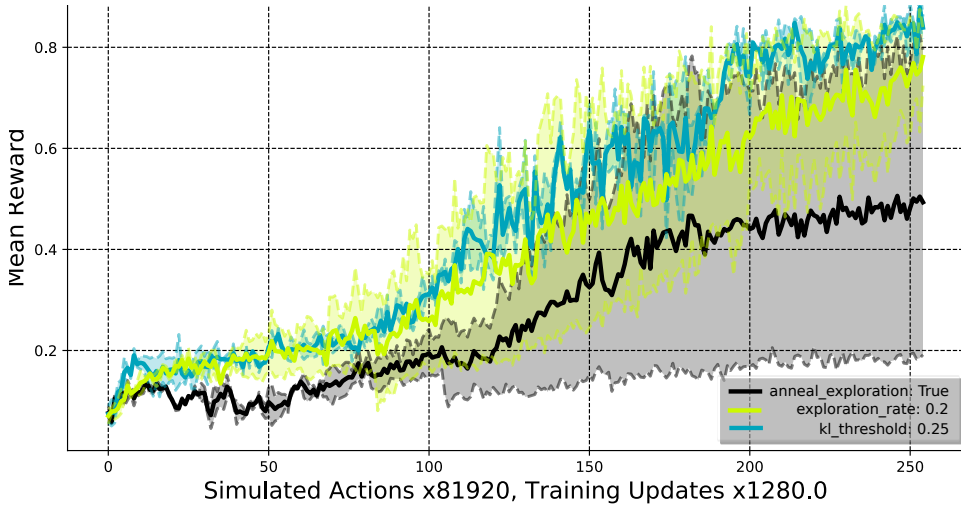



Figure 8: Hyperparameter exploration learning curves for the LC training.

**Hyper Parameter Exploration** Parameter exploration is a key process in acheiving the best results for DRL methods. In this work we explored different network architectures, policy variances, annealing the variance, learning rates and activation types. The results with the best performance are reported in the paper.

## 8.5 Results Details

### 8.5.1 LC Hyperparameter Exploration

The LC hyperparameter exploration learning curves in Figure 8 illustrate the process of finding optimal training parameters. It was found that annealing the policy variance over time did not increase the learning efficiency. As well, the best settings for the policy variance and *kl_threshold* are $0.2$ and $0.25$ respectively.

### 8.5.2 Multi-agent Navigation Unit Tasks

In this section, we show several multi-agent navigation tasks and their qualitative performance using our method. These are rendered as birds-eye views of scenarios with each agents trajectory over time rendered. Each agents goal is rendered as a red point within a red circle and the shortest linear path to that from the agent's current position is rendered as a red line. We show that agent's learn to navigate towards arbitrary goals repeatedly hitting its mark with each new goal in Figure 9. Predictive reciprocal collision avoidance is important in multi-agent navigation and lends naturalness to

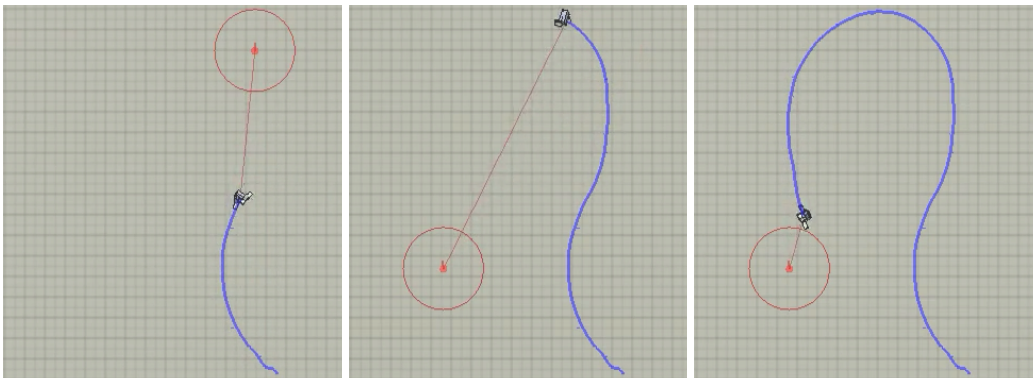
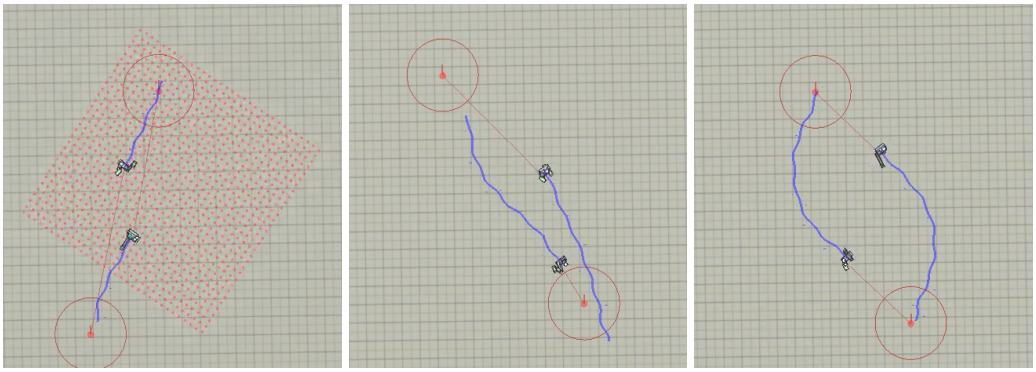


Figure 9: The agent moves towards and reaches goals repeatedly (top-left to bottom-right). The learned model produces smooth goal seeking trajectories show in in blue.



(a) (b) (c)

Figure 10: Three examples demonstrating reciprocal collision avoidance. The size and position of the egocentric state sampling field relative of the agent is shown in (a).

qualitative results. We show our method learns high value reipcrocal collision avoidane strategies, and we overlay the sample points for the velocity state space to illustrate how this is acheived in Figure 10. We show that this generalizes to the introduction of obstacles in both Figure 11 & 12.

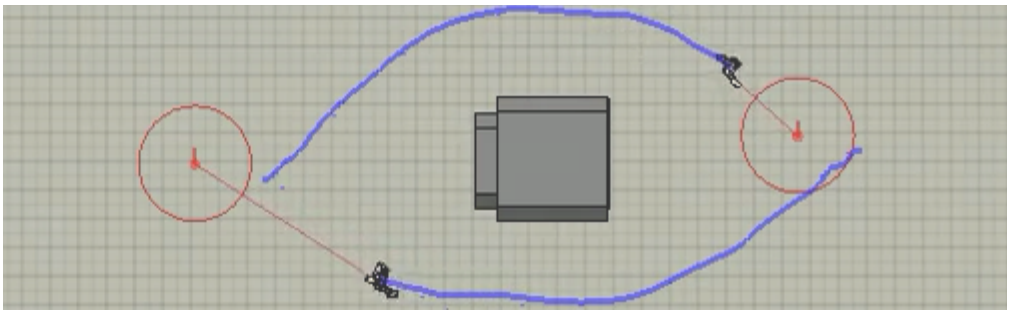
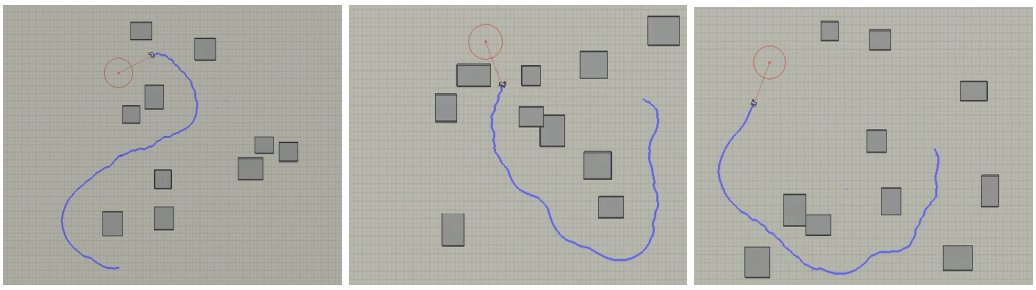


Figure 11: Reciprocal collision avoidance with obstacles. Each agent's initial position is the target location of the other agent.



(a) (b) (c)

Figure 12: An agent reaching goals in arbitrary complex environments. The red circle indicates the final goal.

## 8.6 Additional Discussion

### 8.6.1 LC Agility

While often in the crowd simulation literature it is common to have agents that can turn on point or or produce holonomic motions, this is unrealistic for articulated agents. Interestingly, the agent learned stopping and in-place turning behaviour which was not contained in the imitation data the LC was trained on, indicating the system can generate behaviours beyond its design. However, the agents in this work do not make unrealistic sharp turns. This is in part related to the NC being able to avoid falls, however there are many other factors: The LC motion capture data, $g_L$ selection during LC training and that the LC was not trained with other agents. Progress in any of these areas can improve the responsiveness of the LC and is left for future work.

While the LC is goal driven, this goal is only based on foot placement, a fully interactive agent can have many other types of short term goals, including where to put one's hands to manipulate items in the environment, like doors.