Multi-Robot Collaboration through Reinforcement Learning and Abstract Simulation

Adam Labiosa[1] and Josiah P. Hanna[1]

***Abstract*— Teams of people coordinate to perform complex tasks by forming abstract mental models of world and agent dynamics. The use of abstract models contrasts with much recent work in robot learning that uses a high-fidelity simulator and reinforcement learning (RL) to obtain policies for physical robots. Motivated by this difference, we investigate the extent to which so-called *abstract simulators* can be used for multi-agent reinforcement learning (MARL) and the resulting policies successfully deployed on teams of physical robots. An abstract simulator models the robot's target task at a high-level of abstraction and discards many details of the world that could impact optimal decision-making. Policies are trained in an abstract simulator then transferred to the physical robot by making use of separately-obtained low-level perception and motion control modules. We identify three key categories of modifications to the abstract simulator that enable policy transfer to physical robots: simulation fidelity enhancements, training optimizations and simulation stochasticity. We then run an empirical study with extensive ablations to determine the value of each modification category for enabling policy transfer in cooperative robot soccer tasks. We also compare the performance of policies produced by our methodology with a well-tuned non-learning-based behavior architecture from the annual RoboCup competition and find that our approach leads to a similar level of performance. Broadly we show that MARL can be use to train cooperative physical robot behaviors using highly abstract models of the world.**

I. Introduction

Teamwork enables groups of agents (humans, animals, and robots) to accomplish complex tasks more efficiently than any individual could alone. For example, orca whales use teamwork to generate waves and push prey off of ice [1]. Likewise, autonomous robots can accomplish far more when they work as a team. For example, drones patrolling for wildfires can cover far more area if they coordinate the area they cover [2]. Multi-agent reinforcement learning (MARL) is a promising approach for obtaining such cooperative behaviors for autonomous robots given task interaction time and a reward function that defines success.

While MARL is a general approach to developing cooperative behaviors, current MARL algorithms are data inefficient. Many MARL success stories [3]–[8] have taken place exclusively in simulated environments where data is comparatively cheap to collect. Unfortunately, this scale of data collection is currently infeasible on physical robots.

To take advantage of simulation for physical robots, much recent work has gone toward developing realistic high-fidelity robotics simulators [9]–[11] to reduce the inevitable *reality gap* between simulation and reality. Complementing this research, in this work, we question if high-fidelity simulation is necessary for developing cooperative control policies for physical robots. We draw inspiration from human planning [12] and train in an *abstract* simulation which models the world at a low-fidelity, coarse level of detail (Figure 1b).

In addition to being a step toward imbuing robots with abstract reasoning capabilities, abstract simulators offer several benefits for learning cooperative robotic behaviors. First, in many domains, they will be easier to create and require much less domain knowledge. Second, abstract simulations simplify the world model and thus reduce the need for complex and slow physics calculations. Finally, many cooperative tasks require high-level reasoning, so extensive low-level modeling of the target domain may be unnecessary. With this motivation in mind, our work seeks to answer the question:

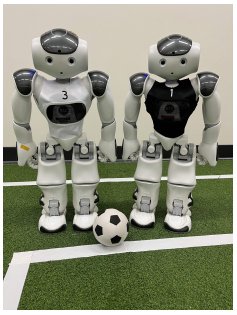
Can a team of robots learn cooperative behaviors using MARL within an abstract simulation of their task such that these behaviors transfer to physical robots?

In this paper, we answer this question affirmatively. In doing so, this work makes the following contributions:

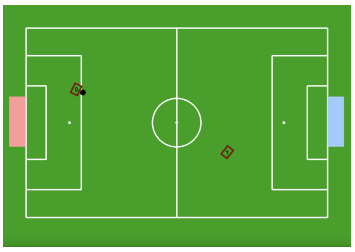
1) We demonstrate that multi-agent policies trained in an abstract simulator can successfully transfer to legged physical robots.
2) We identify and categorize key considerations for improving simulator realism and MARL training that enable zero-shot policy transfer to physical robots. A particularly surprising finding was that *decreasing* simulator fidelity sometimes *increased* sim2real transfer by promoting more effective MARL in simulation.
3) We conduct an extensive ablation study to determine the critical attributes that facilitate the use of abstract simulation to train multi-robot control policies.
4) We demonstrate that our approach produces policies for teams of two robots that perform on par with extensively tuned behaviors developed by recent champions of the RoboCup Standard Platform League competition.[2]

With these contributions, we provide a framework for developing transferable multi-agent policies using abstract simulation, potentially reducing the need for expensive and time-

[1]University of Wisconsin-Madison, Madison, WI, USA. Correspondence to labiosa@wisc.edu



(a) Two NAO humanoid robots used in physical experiments.



(b) Visualization of our abstract simulation used to train policies before deployment on robots. Robots are the small red boxes and the ball is the black circle.

Fig. 1: Comparison of the physical NAOv6 used in experiments to the abstract simulation we use to train the policies to control high-level cooperative behaviors.

consuming real-world training in multi-robot applications.

II. Related Work

In this section, we discuss prior work on abstraction in robotics and sim2real.

A. Abstraction for Robotic Reinforcement Learning

Abstraction is a common technique used to simplify the learning process for RL agents [13]. Prior to deep RL, abstraction was used for navigation tasks for various wheeled and snake-like robots [14]–[17]. Other works have extended abstraction for robotics into deep RL [18]–[21] but do not use abstract simulation to the extent we do in this work nor do they consider multi-agent domains. Truong et al. [22] uses an abstract simulation to accomplish navigation tasks but also does not consider the multi-agent case. Zhang et al. [23] uses abstract simulation for multi-agent drones but does not consider the more complex domain of legged robots.

Another approach to abstraction is to use hierarchy where low-level and high-level control are trained separately. Stulp et al. [24] uses pretrained low-level motion primitives like we do in our work, but they do not use deep RL and it takes place in a simpler single-agent grasping domain. Ma et al. [25] also uses motion primitives for hierarchical RL and considers a multi-agent domain but does not use physical robots. Nachum et al. [26] uses hierarchy to train motion control and then high-level control in a multi-agent domain. Their work considers a simpler domain for pushing boxes with quadrupeds and uses high-fidelity simulation for both low- and high-level policy training.

Some works use low and high-fidelity simulation to balance fast training and real world performance [27]–[33]. Our work only uses low-fidelity training and explores the multi-agent space while these works are focused on single-agent training.

[2]We note that the RoboCup SPL competition presents a greater challenge than the specific evaluation scenarios we designed to test multi-agent cooperation and thus, we do not claim that the trained MARL policies alone would match the performance of an SPL team. In a separate and complementary submission to ICRA, we describe a system for using single-agent RL in the RoboCup SPL competition.

B. Sim2real Reinforcement Learning

Reinforcement learning methods generally require large amounts of data to exhibit desirable behaviors. For example, learning to play Dota 2 required 10 months of training on thousands of CPUs and hundreds of GPUs [7]. This computational scale is infeasible for physical robots due to the prohibitive monetary and time costs. Therefore, many works adopt the sim2real paradigm – training policies in simulation and transferring the fixed policies to physical robots [34], [35].

Sim2real transfer has produced intelligent robot behavior across a wide variety of domains [36]–[45]. In the domain of robot soccer, high-fidelity sim2real has achieved vision-to-motor control, complex multi-agent behaviors and multi-robot teamwork [46]–[49]. Despite the successes of high-fidelity sim2real, the challenge of the reality gap remains. Popular methods to bridge this gap include domain randomization [50], and precise simulation-to-reality matching [51]–[53].

Many works utilize sim2real transfer to enable multi-agent behaviors on real robots [54]–[56]. Some works [57], [58] employ sim2real transfer to deploy multi-agent policies on real robots, but they focus on end-to-end deployment which generally requires a high-fidelity simulation.

Sim2real RL can be considered as an instance of model-based RL. Model-based single-agent RL has been applied to learn on physical robots by learning a world model and training agents in that model [44], [59]–[68]. Model-based RL has also been applied to multi-agent tasks for increased multi-agent performance [69]–[73]. These works focused on learning world models as opposed to using a crafted simulation and do not evaluate on physical robots.

III. Background

We formalize the MARL problem as solving a stochastic game, also known as a Markov game. A stochastic game is defined by the tuple $(\mathcal{S}, \mathcal{A}_1, \ldots, \mathcal{A}_n, p, r_1, \ldots, r_n)$, where n is the number of agents; $\mathcal{S}$ is the set of all possible states of the environment; and $\mathcal{A}_i$ is the set of actions available to agent i. The joint action space $\mathcal{A}$ is defined as $\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_n$, representing all possible combinations of actions taken by the agents. $p(s'|s, a_1, \ldots, a_n)$ is the probability of transitioning to state s' given the current state s and the joint action $(a_1, \ldots, a_n)$ taken by all agents. $r_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function for agent i, which depends on the current state and the joint action. In a stochastic game, the goal of each agent i is to learn a policy $\pi_i : \mathcal{S} \to \mathcal{A}_i$ that maximizes its expected cumulative reward. This work focuses on a cooperative multi-agent setting. In cooperative games all agents share the same reward function and aim to maximize this shared cumulative reward.

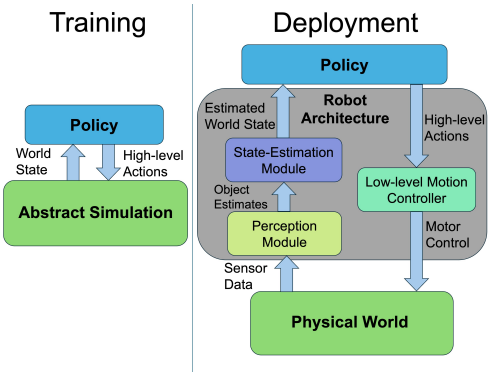


Fig. 2: Visualization of our training and deployment interaction models. Exact world state is given during training and world state is estimated by the robot architecture modules during deployment. Similarly, in simulation the policy exactly controls high-level actions while during deployment they are passed to a motion controller which controls motor outputs.

IV. Robot Hardware and Simulation

Before presenting our methodology for crossing the abstract simulator reality gap, we describe the robot hardware and simulation set-up that we will use. We present these platforms to ground the presentation of our methodology in Section V.

A. Robot Hardware and Behavior Architecture

In this work, we use the NAO v6 humanoid robot (Figure 1a). The NAO has 25 degrees of freedom for movement with 2 cameras, foot pressure sensors, and IMUs for perception. All computation runs onboard the robot using a 4-core Intel Atom processor.

Our work focuses on enabling MARL to learn high-level decision-making for the NAO in cooperative tasks. To facilitate the use of high-level policies, we make use of separately implemented low-level perception and motion capabilities by training without an end-to-end approach and build upon the NAO behavior architecture developed by the University of Bremen's RoboCup team, B-Human [74]. This behavior architecture provides the full capabilities required to participate in the RoboCup Standard Platform League competition including low-level modules that handle real-time object detection, localization, and motor control and high-level modules for team and individual robot behaviors (Figure 2). Our focus in this work is to learn a policy that replaces the high-level behavior modules that maps state estimates to high-level decisions.

The state and action spaces are designed to maximize the benefits of the abstract simulation and the existing robot architecture. The state space $\mathcal{S}$ includes egocentric representations of objects around the field represented by $\Delta X, \Delta Y, \cos\theta, \sin\theta$. The agents action space $\mathcal{A}$ is given as longitudinal movement, lateral movement, angular rotation, and optionally kick or stand.

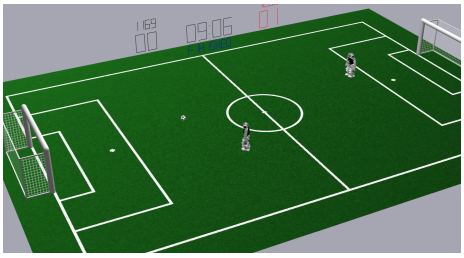


Fig. 3: High-fidelity simulation developed by the B-Human RoboCup Team. Used for simulation experiments. Physics are based on the Open Dynamics Engine [76].

B. Abstract Simulation

Since we focus on investigating the use of high-level abstract simulators for MARL, in this section, we introduce the abstract simulator (AbstractSim – Figure 1b) that we used in this study to ground the following sections. We note that AbstractSim is not part of our contribution and instead it was developed independently for single-agent RL use in the RoboCup competition [75]. We use it here as a case study to evaluate the role of abstract simulations in complex, multi-agent tasks. AbstractSim models the state and transition dynamics of the world without explicitly modeling how the physical robot estimates the state or carries out actions. Object representations are also simplified with the bipedal robots represented as rectangles with no agent-to-agent collision dynamics. Both of these design decisions impact the dynamics of both the robot and soccer ball we use in our experiments. At every timestep the robot moves precisely in the direction indicated by the policy, abstracting away complexities such as momentum and gait and ignoring all dynamics of legged movement. Ball dynamics are also simplified. The ball moves at a consistent speed and direction upon contact and decelerates at a fixed rate.

For MARL, AbstractSim offers two key benefits. First, the simplifications facilitate faster training which makes it easier to iterate on the MARL training approach (e.g., tuning the reward function and training set-up). In comparison to our high-fidelity simulation (Figure 3), AbstractSim runs 30x as fast. Second, it reduces the need for accurate robot and environment models. The key challenge – that we address next – is that AbstractSim has a potentially large reality gap which hinders direct policy transfer. The reality gap arises from the significance difference between the simplified dynamics of AbstractSim and the complex dynamics of the real world. Concretely, with our initial version of the simulator, agents failed to learn effective cooperative behaviors or even simple single-agent behaviors. In the next section, we describe the key techniques that enable the use of MARL and our AbstractSim to train cooperative policies for the physical robots.

V. Ingredients for Multi-agent Abstract Sim2Real

To successfully apply MARL in an abstract simulator, we identify three categories of changes to the simulator and

	Full MARL (F)	Large Displacement (E)	Realistic Agent Size (T)	No Ball Noise (N)
BS 1	$\mathbf{0.90 \pm 0.2}$	0.70 ± 0.3	0.20 ± 0.3	0.10 ± 0.2
BS 2	$\mathbf{1.00 \pm 0.0}$	0.70 ± 0.4	0.50 ± 0.4	0.90 ± 0.2
BS 3	$\mathbf{0.80 \pm 0.3}$	0.60 ± 0.4	0.30 ± 0.4	0.30 ± 0.4
D 1	$\mathbf{0.50 \pm 0.4}$	0.00 ± 0.0	0.40 ± 0.4	0.10 ± 0.2
D 2	$\mathbf{1.00 \pm 0.0}$	0.80 ± 0.3	$\mathbf{1.00 \pm 0.0}$	0.30 ± 0.4
D 3	0.80 ± 0.3	$\mathbf{0.90 \pm 0.2}$	0.70 ± 0.4	0.00 ± 0.0

(a) Success Rates for Basic and Defender Physical Robot Experiments. Higher is better.

	Full MARL (F)	Large Displacement (E)	Realistic Agent Size (T)	No Ball Noise (N)
BS 1	45.78 ± 8.1	43.14 ± 4.9	$\mathbf{39.50 \pm 1.0}$	60.00 ± 0.0
BS 2	$\mathbf{24.30 \pm 8.8}$	24.86 ± 6.4	37.60 ± 8.9	39.44 ± 5.8
BS 3	43.75 ± 5.6	$\mathbf{37.00 \pm 5.4}$	55.00 ± 3.0	49.67 ± 9.2
D 1	$\mathbf{40.60 \pm 12.3}$	N/A	54.25 ± 4.8	56.00 ± 0.0
D 2	23.50 ± 3.0	$\mathbf{22.00 \pm 2.7}$	35.50 ± 5.2	45.67 ± 20.3
D 3	39.00 ± 8.5	$\mathbf{32.22 \pm 3.5}$	$\mathbf{32.22 \pm 3.5}$	N/A

(b) Time to Score for Basic and Defender Physical Robot Experiments. Lower is better.

Fig. 4: Physical robot experiment results. F are full methods. E are fidelity enhancements. T are training optimizations. N are noise. Our full method is Full MARL. Each column provides an experiment location for either the Basic Soccer (BS) task or the Defender (D) task. Number of trials is 10. 95% confidence intervals computed using the Student t-distribution.

training procedure that enable successful policy transfer.

A. Simulation Fidelity Enhancements

Some degree of realism is required to enable policy transfer even when the simulation is an intentionally abstract model of the world so we first consider modifications that improve the realism of world dynamics. We found that increasing the realism of the simulation is beneficial only to the extend that it is necessary for training cooperative behaviors. Overly complex simulations can increase learning difficulty without improving policy transfer. Therefore, we identify a small set of modifications that make key details more realistic.

Concretely, we modify the simulated agents' speed to match the physical robots' speed. From the initial AbstractSim, we lowered the robot's angular and translational velocity movements. These modifications aim to reduce the reality gap between the abstract simulator and reality by improving the accuracy of the dynamics to the real world.

Contrary to the assumption that increased realism always enhances real-world performance, we observed that in some cases, decreasing certain elements of realism in the simulation actually improved the physical robots' performance. We term these counterintuitive simulation adjustments *Training Optimizations*.

B. Training Optimizations

The objective of our method is to maximize performance on real robots, which requires both effective MARL in simulation and effective sim2real transfer. A counterintuitive observation is that some changes – designed to increase realism and promote transfer – led to MARL failing in AbstractSim. Consequently, the policies we obtained were not performant on the physical robot. We observe that too much realism can make the simulation overly complex or challenging, hindering the learning process and leading to overfitting in simulation. Therefore, it is sometimes beneficial to make changes that *decrease* realism so as to balance trainability with transferability.

Concretely, we modified three aspects of our simulator to less closely match the physical world: robot size, goal size and the time to kick. First, we increased the robot size to about four times the size of a real NAO which increased the ball-robot contact distance. Second, we decreased the goal size to force precision when kicking into the goal. Finally, we removed any time delay during kicking. It takes about one second for the physical robots to kick the ball, but during training, we remove this delay as it hinders learning. These modifications, while seemingly counterintuitive, seek to address the reality gap by encouraging the learning of more robust policies that are less sensitive to the dynamics of the simulation environment.

C. Simulation Stochasticity

Even when it is beneficial to make an abstract simulator more realistic, there are limits to how realistic we can make the simulation without significantly increasing its complexity. Accurately modeling complex real-world dynamics like robot-to-ball contact would require details models of the robots and world kinematics which would increase complexity and possibly hinder learning. Instead of increasing realism, we propose to increase robustness by adding noise into hard to model dynamics. This type of randomization is sometimes used as part of domain randomization for sim2real.

Concretely, we considered two uses of noise with our platforms. First, during training, we add noise to robot-ball contact during a push contact but not a kick, as our robots have precise kicking capabilities. We add uniform noise as we observe highly stochastic motion following contacts on the physical robots. We also explore the use of noise to encourage robustness to error in localization on the physical robot. We do so by adding noise to the observations of the robot during training.

By introducing noise into the simulation, we increase the variability of the environment and encourage the learning of more robust policies that are less sensitive to the uncertainties of the real-world.

VI. Experimental Analysis

We now present an empirical study designed to answer the questions:

1) Can multi-agent robot control policies trained in an abstract simulator transfer directly to physical collaboration tasks?

	Full MARL (F)	BHuman (F)	Large Displacement (E)	Small Displacement (E)	Large Angle Displacement (E)	Realistic Agent Size (T)	Realistic Goals (T)	Kicking Time (T)	No Ball Noise (N)	With Observation Noise (N)
BS 1	$\mathbf{0.88 \pm 0.1}$	0.87 ± 0.1	0.66 ± 0.1	0.62 ± 0.1	0.75 ± 0.1	0.71 ± 0.1	0.83 ± 0.1	0.54 ± 0.1	0.39 ± 0.1	0.74 ± 0.1
BS 2	$\mathbf{1.00 \pm 0.0}$	$\mathbf{1.00 \pm 0.0}$	0.89 ± 0.1	0.74 ± 0.1	0.99 ± 0.0	0.82 ± 0.1	0.98 ± 0.0	0.97 ± 0.0	0.96 ± 0.0	0.88 ± 0.1
BS 3	0.97 ± 0.0	$\mathbf{0.99 \pm 0.0}$	0.80 ± 0.1	0.51 ± 0.1	0.71 ± 0.1	0.76 ± 0.1	0.92 ± 0.1	0.11 ± 0.1	0.50 ± 0.1	0.80 ± 0.1
D 1	$\mathbf{0.84 \pm 0.1}$	0.69 ± 0.1	0.44 ± 0.1	0.00 ± 0.0	0.01 ± 0.0	0.05 ± 0.0	0.13 ± 0.1	0.33 ± 0.1	0.02 ± 0.0	0.12 ± 0.1
D 2	0.79 ± 0.1	$\mathbf{0.93 \pm 0.1}$	0.21 ± 0.1	0.04 ± 0.0	0.81 ± 0.1	0.68 ± 0.1	0.20 ± 0.1	0.82 ± 0.1	0.89 ± 0.1	0.13 ± 0.1
D 3	0.80 ± 0.1	$\mathbf{0.98 \pm 0.0}$	0.66 ± 0.1	0.00 ± 0.0	0.71 ± 0.1	0.46 ± 0.1	0.53 ± 0.1	0.65 ± 0.1	0.18 ± 0.1	0.46 ± 0.1

(a) Success Rates for Basic and Defender Simulation Experiments. Higher is better.

	Full MARL (F)	BHuman (F)	Large Displacement (E)	Small Displacement (E)	Large Angle Displacement (E)	Realistic Agent Size (T)	Realistic Goals (T)	Kicking Time (T)	No Ball Noise (N)	With Observation Noise (N)
BS 1	36.50 ± 1.7	$\mathbf{31.42 \pm 1.7}$	42.04 ± 1.5	41.62 ± 2.0	42.66 ± 1.8	32.65 ± 1.6	36.55 ± 1.7	45.48 ± 2.3	49.15 ± 2.1	39.45 ± 1.9
BS 2	16.42 ± 1.3	$\mathbf{11.28 \pm 0.7}$	18.30 ± 1.3	20.96 ± 1.5	17.48 ± 0.6	21.45 ± 1.6	19.78 ± 1.1	23.65 ± 1.8	24.07 ± 1.4	19.47 ± 1.5
BS 3	33.56 ± 1.2	$\mathbf{21.34 \pm 1.1}$	38.94 ± 2.3	37.22 ± 2.8	42.65 ± 2.2	41.05 ± 1.7	46.62 ± 1.1	52.45 ± 4.3	51.96 ± 1.6	34.00 ± 1.5
D 1	34.14 ± 1.7	$\mathbf{29.79 \pm 2.1}$	45.06 ± 2.9	37.22 ± 2.8	57.00 ± 0.0	52.80 ± 5.4	43.69 ± 4.1	45.58 ± 3.4	54.00 ± 2.0	43.58 ± 3.9
D 2	34.74 ± 2.5	$\mathbf{14.23 \pm 1.4}$	32.81 ± 4.0	59.00 ± 1.1	30.99 ± 2.0	52.01 ± 1.6	36.95 ± 4.4	43.22 ± 2.0	46.27 ± 1.9	36.54 ± 6.0
D 3	33.23 ± 1.9	$\mathbf{20.40 \pm 1.0}$	40.66 ± 3.1	37.22 ± 2.8	34.22 ± 2.1	47.93 ± 2.7	29.85 ± 2.0	47.22 ± 2.1	49.56 ± 3.8	28.39 ± 2.0

(b) Time to Score for Basic and Defender Simulation Experiments. Lower is better.

Fig. 5: Simulation experiment results. F are full methods. E are fidelity enhancements. T are training optimizations. N are noise. Our full method is Full MARL. Each column provides an experiment location for either the Basic Soccer (BS) task or the Defender (D) task. Number of trials is 100. 95% confidence intervals computed using the Student t-distribution.

2) Which parts of our methodology are most critical for enabling transfer?

A. Empirical Setup

To answer our empirical questions, we train policies in abstract simulation and then evaluate them on both physical robots and using a high-fidelity simulation as a surrogate for real world evaluation. In this subsection we detail the training setup and provide empirical results.

We consider our fullMARL training method to be: tuned agent displacement, large training agents, small training goals, no kicking time and ball-robot contact noise. To understand the impact of each component of our fullMARL method, we conduct an ablation study where we remove one component while keeping the other components unchanged.

For testing, we construct two cooperative tasks based on the robot soccer domain and measure performance based on success rate and time to score:

1) **Basic Soccer.** Our first task, which we call basic soccer, has two robots and a ball on an empty field. During evaluation, we configure the start location of the robots such that performance is higher when they cooperate.
2) **Static Defender.** The second task adds a non-moving defender robot that is set either between a robot and the ball or a robot and the goal. The addition of this robot introduces an obstacle that makes it more difficult to learn cooperative behaviors.

In both tasks, we consider three initial configurations for evaluation which emphasize cooperative behaviors (Figure 6).

On the physical robots we run the our suite of experiments with our fullMARL method along with one method from each ingredient category (Section V). Physical robot experiments are costly to run, so we supplement them with high-fidelity simulation experiments. The high-fidelity simulation, developed by the B-Human team, closely matches the real-world performance of our policies (Figure 3). In simulation we also compare our results to a RoboCup-winning architecture developed by B-Human, the same group that created the underlying localization and movement modules. We do note that while our policies perform at the level of the B-Human behavior in our testing, our evaluation is just one step toward robust behaviors that handle the complexities of robot soccer as effectively as their code does.

For policy training, we use the StableBaselines3 [77] implementation of Proximal Policy Optimization (PPO) [78] and use the SuperSuit [79] library to enable MARL training. We use independent learning as the MARL approach in the abstract simulator and policies are trained with shared neural network weights which speeds up training time [80].

B. Experimental Results

Here we analyze our real robot and simulation experiment results.

1) Simulation Fidelity Enhancement Analysis: In both simulations and physical robots, we observe that an imprecise training walk speed negatively impacts the success rate. Specifically, the large training displacement caused the robot to score 0% of the time in the static defender analysis setting since the robot kicked the ball out of bounds on every run. In the static defender simulation, both small training displacement and large angle displacement performed 40% worse than large training displacement, achieving about a 0% success rate. Overall, all three changes to the displacement lowered the success rate by at least 15% and at most 80%. These findings show the importance of careful tuning of parameters to ensure the simulated motion closely matches the real-world motion dynamics.

2) Training Optimization Analysis: In our training optimization analysis, the realistic agent size performed significantly better in the static defender task, likely because the

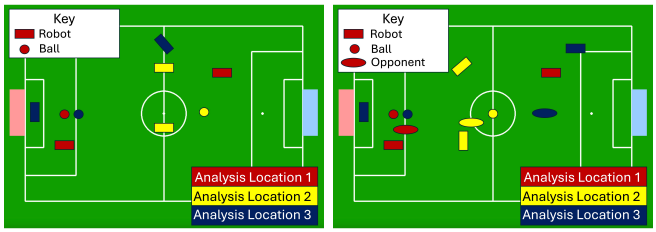


Fig. 6: Analysis locations for basic soccer task and defender task. (Left) Basic soccer task. (Right) Static defender task.

large agents in simulation encouraged the robot to maintain a farther distance to other robots. In simulation, the realistic agents, realistic goal sizes performed well on the basic soccer task but very poorly on the static defender task. This suggests that these modifications help especially in more challenging scenarios. Lastly, for the no kicking time modification, we see less than a 50% success rate in both the basic soccer and static defender tasks. Qualitatively, this is due to the difficulty in learning to kick the ball and instead pushing it due to the kicking delay. These findings show the need for careful consideration of training optimizations due to their potential large impact for on-robot performance.

3) Simulation Stochasticity Enhancement Analysis: We consider two types of noise to add stochasticity to our simulation. The first and more impactful of the two is ball contact noise. Quantitatively, this method failed 9 out of 10 times on both physical robot tasks. Qualitatively, it only kicked the ball once throughout all the trials. When observing in AbstractSim, the agents rarely kick which suggests that without contact noise, the agents overfit to the optimized dynamics of AbstractSim and learn that pushing the ball is a consistent method of scoring goals. This result holds in both simulation domains, as the method does not achieve above a 40% success rate. Observation noise is less impactful but does hurt performance. This suggests that our underlying localization module is robust enough that adding noise during training hurts understanding of the ball location.

These results demonstrate that multi-agent robot control policies can be successfully trained in an abstract simulator and transferred to physical robot collaboration tasks but only when the environment is correctly designed. The ablation study shows the critical importance of accurate simulation for motion and the correct selection of training optimizations and simulation stochasticity.

VII. Discussion and Future Work

Our empirical study demonstrates the feasibility of using highly abstract simulations for MARL training of cooperative control policies for physical robots. In order to cross the reality gap from the abstract simulator, we identified three categories of simulator and training modifications that enable transfer from abstract simulation to reality. Our physical robot experiments revealed that ball noise was the most impactful change. The use of ball-robot contact noise likely helped the agent avoid overfitting to the simplified abstract dynamics and underscores the critical importance of introducing simulation stochasticity to coarsely model contact dynamics.

While all three categories contributed substantially to successful sim-to-real transfer, of particular interest are the Training Optimizations, which yielded some of the most intriguing results. These optimizations, counterintuitively, involved reducing certain aspects of simulation realism to enhance real-world performance. This approach not only improved transfer success but also challenged our assumptions about the relationship between simulation fidelity and real-world effectiveness.

While a promising initial step toward the use of abstract simulators to develop multi-robot systems, we identify several limitations and directions for future work. First, the scenarios we used involved two robots cooperating and only required basic levels of teamwork. In the future, we plan to extend the use of abstract simulation for MARL training of teams of 5 robots to play against 5 other robots as used in the RoboCup SPL competition. This direction will require the study of sim2real in the presence of adversarial agents. Second, our method relies on existing lower level perception, localization, and control modules. The success of the decision making is contingent on the accuracy of the localization and movement control of the agent. This limitation is seen in the failure cases of our Defender task. The robot is unable to detect the defending robot and this results in a failed run. This limitation is increasingly less of an issue as many robotics systems ship with competent movement and perception code. Third, in this paper we manually selected the fidelity and training improvements. In the future, we aim to automate their selection. Finally, future work should consider the theoretical foundation of what creates a training or fidelity improvement. This foundation would provide deeper insight into the understanding of why some simulation aspects improve performance with less accurate world-modeling.

VIII. Conclusion

In this work we presented a method to enable teamwork on physical robots using MARL and abstract simulation. Starting with a base abstract simulator, we identified key simulator adjustments that enabled effective MARL and subsequent sim2real transfer. We conducted an extensive ablation study to identify the key components that significantly impact the success of our teamwork in the domain of robot soccer. These components fall into three categories: fidelity enhancement, training optimizations, and simulation stochasticity. Our findings demonstrate that our method successfully enables teamwork on physical robots in the robot soccer domains and compares favorably to a state-of-the-art behavior architecture designed for the RoboCup competition. These findings show that MARL can be an effective tool for developing cooperative robot behaviors even without the use of high-fidelity simulation.

REFERENCES

[1] R. L. Pitman and J. W. Durban, "Cooperative hunting behavior, prey selectivity and prey handling by pack ice killer whales (orcinus orca), type b, in antarctic peninsula waters," *Marine Mammal Science*, vol. 28, no. 1, pp. 16–36, 2012.

[2] M. Momeni, H. Soleimani, S. Shahparvari, and B. Afshar-Nadjafi, "A multi-agency coordination resource allocation and routing decision-making problem: A coordinated truck-and-drone dss for improved wildfire detection coverage," *International journal of disaster risk reduction*, vol. 97, p. 104027, 2023.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[5] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[6] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[7] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[8] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," *arXiv preprint arXiv:1909.07528*, 2019.

[9] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[10] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[11] R. Serban, A. Tasora, and D. Negrut, "Chrono: An open-source multi-physics simulation package," in *presentado en The 5th Joint International Conference on Multibody System Dynamics, Lisboa, Portugal*, 2018.

[12] M. K. Ho, D. Abel, C. G. Correa, M. L. Littman, J. D. Cohen, and T. L. Griffiths, "People construct simplified mental representations to plan," *Nature*, vol. 606, no. 7912, pp. 129–136, 2022.

[13] L. Li, T. J. Walsh, and M. L. Littman, "Towards a unified theory of state abstraction for mdps." *AI&M*, vol. 1, no. 2, p. 3, 2006.

[14] J. Provost, B. J. Kuipers, and R. Miikkulainen, "Self-organizing perceptual and temporal abstraction for robot reinforcement learning," in *AAAI Workshop on Learning and Planning in Markov Processes*, 2004, pp. 79–84.

[15] K. Shibata, "Learning of deterministic exploration and temporal abstraction in reinforcement learning," in *2006 SICE-ICASE International Joint Conference*. IEEE, 2006, pp. 4569–4574.

[16] K. Ito, Y. Fukumori, and A. Takayama, "Autonomous control of real snake-like robot using reinforcement learning; abstraction of state-action space using properties of real world," in *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*. IEEE, 2007, pp. 389–394.

[17] L. Frommberger and D. Wolter, "Structural knowledge transfer by spatial abstraction for reinforcement learning agents," *Adaptive Behavior*, vol. 18, no. 6, pp. 507–525, 2010.

[18] K. Ito and Y. Takeuchi, "Reinforcement learning in dynamic environment: abstraction of state-action space utilizing properties of the robot body and environment," *Artificial Life and Robotics*, vol. 21, pp. 11–17, 2016.

[19] R. Veerapaneni, J. D. Co-Reyes, M. Chang, M. Janner, C. Finn, J. Wu, J. Tenenbaum, and S. Levine, "Entity abstraction in visual model-based reinforcement learning," in *Conference on Robot Learning*. PMLR, 2020, pp. 1439–1456.

[20] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on robot learning*. PMLR, 2023, pp. 403–415.

[21] I. Geles, L. Bauersfeld, A. Romero, J. Xing, and D. Scaramuzza, "Demonstrating agile flight from pixels without state estimation," *arXiv preprint arXiv:2406.12505*, 2024.

[22] J. Truong, M. Rudolph, N. H. Yokoyama, S. Chernova, D. Batra, and A. Rai, "Rethinking sim2real: Lower fidelity simulation leads to higher sim2real transfer in navigation," in *Conference on Robot Learning*. PMLR, 2023, pp. 859–870.

[23] Y. Zhang, Y. Hu, Y. Song, D. Zou, and W. Lin, "Back to newton's laws: Learning vision-based agile flight via differentiable physics," *arXiv preprint arXiv:2407.10648*, 2024.

[24] F. Stulp and S. Schaal, "Hierarchical reinforcement learning with movement primitives," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2011, pp. 231–238.

[25] A. Ma, M. Ouimet, and J. Cortés, "Hierarchical reinforcement learning via dynamic subspace search for multi-agent planning," *Autonomous Robots*, vol. 44, no. 3, pp. 485–503, 2020.

[26] O. Nachum, M. Ahn, H. Ponte, S. Gu, and V. Kumar, "Multi-agent manipulation via locomotion using hierarchical sim2real," *arXiv preprint arXiv:1908.05224*, 2019.

[27] M. Cutler, T. J. Walsh, and J. P. How, "Real-world reinforcement learning via multifidelity simulators," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 655–671, 2015.

[28] J. Qiu, C. Yu, W. Liu, T. Yang, J. Yu, Y. Wang, and H. Yang, "Low-cost multi-agent navigation via reinforcement learning with multi-fidelity simulator," *IEEE Access*, vol. 9, pp. 84 773–84 782, 2021.

[29] J. J. Beard and A. Baheri, "Black-box safety validation of autonomous systems: A multi-fidelity reinforcement learning approach," *arXiv preprint arXiv:2203.03451*, 2022.

[30] A. Agrawal and C. McComb, "Reinforcement learning for efficient design space exploration with variable fidelity analysis models," *Journal of Computing and Information Science in Engineering*, vol. 23, no. 4, p. 041004, 2023.

[31] S. Bhola, S. Pawar, P. Balaprakash, and R. Maulik, "Multi-fidelity reinforcement learning framework for shape optimization," *Journal of Computational Physics*, vol. 482, p. 112018, 2023.

[32] D. F. Leguizamo, H.-J. Yang, X. Y. Lee, and S. Sarkar, "Deep reinforcement learning for robotic control with multi-fidelity models," *IFAC-PapersOnLine*, vol. 55, no. 37, pp. 193–198, 2022.

[33] J. Truong, D. Yarats, T. Li, F. Meier, S. Chernova, D. Batra, and A. Rai, "Learning navigation skills for legged robots with learned robot embeddings," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 484–491.

[34] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[35] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, "Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning," *IEEE Access*, vol. 9, pp. 153 171–153 187, 2021.

[36] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[37] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[38] X. Cao, C. Sun, and M. Yan, "Target search control of auv in underwater environment with deep reinforcement learning," *IEEE Access*, vol. 7, pp. 96 549–96 559, 2019.

[39] I. Carlucho, M. De Paula, S. Wang, Y. Petillot, and G. G. Acosta, "Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning," *Robotics and Autonomous Systems*, vol. 107, pp. 71–86, 2018.

[40] T. Hester and P. Stone, "Texplore: real-time sample-efficient reinforcement learning for robots," *Machine learning*, vol. 90, pp. 385–429, 2013.

[41] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," in *Conference on robot learning*. PMLR, 2018, pp. 561–591.

[42] P. Falco, A. Attawia, M. Saveriano, and D. Lee, "On policy learning robust to irreversible events: An application to robotic in-hand ma-

nipulation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1482–1489, 2018.

[43] M. Hazara and V. Kyrki, "Transferring generalizable motor primitives from simulation to real world," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2172–2179, 2019.

[44] G. Chalvatzaki, X. S. Papageorgiou, P. Maragos, and C. S. Tzafestas, "Learn to adapt to human walking: A model-based reinforcement learning approach for a robotic assistant rollator," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3774–3781, 2019.

[45] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[46] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, J. Humplik, M. Wulfmeier, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner *et al.*, "Learning agile soccer skills for a bipedal robot with deep reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi8022, 2024.

[47] D. Tirumala, M. Wulfmeier, B. Moran, S. Huang, J. Humplik, G. Lever, T. Haarnoja, L. Hasenclever, A. Byravan, N. Batchelor *et al.*, "Learning robot soccer from egocentric vision with deep reinforcement learning," *arXiv preprint arXiv:2405.02425*, 2024.

[48] Y. Duan, B. X. Cui, and X. H. Xu, "A multi-agent reinforcement learning approach to robot soccer," *Artificial Intelligence Review*, vol. 38, pp. 193–211, 2012.

[49] K.-H. Park, Y.-J. Kim, and J.-H. Kim, "Modular q-learning based multi-agent cooperation for robot soccer," *Robotics and Autonomous systems*, vol. 35, no. 2, pp. 109–122, 2001.

[50] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[51] M. Kaspar, J. D. M. Osorio, and J. Bock, "Sim2real transfer for reinforcement learning without dynamics randomization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4383–4388.

[52] J. P. Hanna, S. Desai, H. Karnan, G. Warnell, and P. Stone, "Grounded action transformation for sim-to-real reinforcement learning," *Machine Learning*, vol. 110, no. 9, pp. 2469–2499, 2021.

[53] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[54] J. Blumenkamp, S. Morad, J. Gielis, Q. Li, and A. Prorok, "A framework for real-world multi-robot systems running decentralized gnn-based policies," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8772–8778.

[55] C. S. de Witt, B. Peng, P.-A. Kamienny, P. Torr, W. Böhmer, and S. Whiteson, "Deep multi-agent reinforcement learning for decentralized continuous cooperative control," *arXiv preprint arXiv:2003.06709*, vol. 19, 2020.

[56] Z. Cui, C. Yang, and X. Cao, "A distributed simulation-to-reality transfer framework for multi-agent reinforcement learning," in *2024 36th Chinese Control and Decision Conference (CCDC)*. IEEE, 2024, pp. 3807–3812.

[57] E. Candela, L. Parada, L. Marques, T.-A. Georgescu, Y. Demiris, and P. Angeloudis, "Transferring multi-agent reinforcement learning policies for autonomous driving using sim-to-real," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 8814–8820.

[58] W. Zhao, J. P. Queralta, L. Qingqing, and T. Westerlund, "Towards closing the sim-to-real gap in collaborative multi-robot deep reinforcement learning," in *2020 5th International conference on robotics and automation engineering (ICRAE)*. IEEE, 2020, pp. 7–12.

[59] F. Djeumou, C. Neary, and U. Topcu, "How to learn and generalize from three minutes of data: Physics-constrained and uncertainty-aware neural stochastic differential equations," *arXiv preprint arXiv:2306.06335*, 2023.

[60] J. Huang, Y. Zhang, F. Giardina, and A. Rosendo, "Trade-off on sim2real learning: Real-world learning faster than simulations," in *2022 8th International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, 2022, pp. 95–100.

[61] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg, "Daydreamer: World models for physical robot learning," in *Conference on Robot Learning*. PMLR, 2023, pp. 2226–2240.

[62] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 124–134, 2018.

[63] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine, "Solar: Deep structured representations for model-based reinforcement learning," in *International conference on machine learning*. PMLR, 2019, pp. 7444–7453.

[64] T. Hester, M. Quinlan, and P. Stone, "Rtmba: A real-time model-based reinforcement learning architecture for robot control," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 85–90.

[65] P. Becker-Ehmck, M. Karl, J. Peters, and P. van der Smagt, "Learning to fly via deep model-based reinforcement learning," *arXiv preprint arXiv:2003.08876*, 2020.

[66] D. Martínez, G. Alenya, and C. Torras, "Safe robot execution in model-based reinforcement learning," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 6422–6427.

[67] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, and K. S. Pister, "Low-level control of a quadrotor with deep model-based reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4224–4230, 2019.

[68] N. Imanberdiyev, C. Fu, E. Kayacan, and I.-M. Chen, "Autonomous navigation of uav by using real-time model-based reinforcement learning," in *2016 14th international conference on control, automation, robotics and vision (ICARCV)*. IEEE, 2016, pp. 1–6.

[69] P. Van Der Vaart, A. Mahajan, and S. Whiteson, "Model based multi-agent reinforcement learning with tensor decompositions," *arXiv preprint arXiv:2110.14524*, 2021.

[70] K. Zhang, S. Kakade, T. Basar, and L. Yang, "Model-based multi-agent rl in zero-sum markov games with near-optimal sample complexity," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1166–1178, 2020.

[71] B. Pasztor, I. Bogunovic, and A. Krause, "Efficient model-based multi-agent mean-field reinforcement learning," *arXiv preprint arXiv:2107.04050*, 2021.

[72] P. G. Sessa, M. Kamgarpour, and A. Krause, "Efficient model-based multi-agent reinforcement learning via optimistic equilibrium computation," in *International Conference on Machine Learning*. PMLR, 2022, pp. 19 580–19 597.

[73] V. Egorov and A. Shpilman, "Scalable multi-agent model-based reinforcement learning," *arXiv preprint arXiv:2205.15023*, 2022.

[74] B.-H. Team, "B-Human Code Release," https://wiki.b-human.de/coderelease2023/, October 2023. [Online]. Available: https://wiki.b-human.de/coderelease2023/

[75] "GitHub - Badger-RL/BadgerRLSystemCodeRelease2023: A Robocup CodeRelease forked From BHumanCodeRelease2022 — github.com," https://github.com/Badger-RL/BadgerRLSystemCodeRelease2023, [Accessed 16-09-2024].

[76] R. Smith *et al.*, "Open dynamics engine," 2005.

[77] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

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

[79] J. K. Terry, B. Black, and A. Hari, "Supersuit: Simple microwrappers for reinforcement learning environments," *arXiv preprint arXiv:2008.08932*, 2020.

[80] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 29, 2016.