JaxRobotarium: Training and Deploying Multi-Robot Policies in 10 Minutes

Shalin Anand Jain, Jiazhen Liu, Siva Kailas, Harish Ravichandar
Georgia Institute of Technology, United States

Abstract: Multi-agent reinforcement learning (MARL) has emerged as a promising solution for learning complex and scalable coordination behaviors in multi-robot systems. However, established MARL platforms (e.g., SMAC and MPE) lack robotics relevance and hardware deployment, leaving multi-robot learning researchers to develop bespoke environments and hardware testbeds dedicated to the development and evaluation of their individual contributions. The Multi-Agent RL Benchmark and Learning Environment for the Robotarium (MARBLER) is an exciting recent step in providing a standardized robotics-relevant platform for MARL, by bridging the Robotarium testbed with existing MARL software infrastructure. However, MARBLER lacks support for parallelization and GPU/TPU execution, making the platform prohibitively slow compared to modern MARL environments and hindering adoption. We contribute JaxRobotarium, a Jax-powered end-to-end simulation, learning, deployment, and benchmarking platform for the Robotarium. JaxRobotarium enables rapid training and deployment of multi-robot reinforcement learning (MRRL) policies with realistic robot dynamics and safety constraints, supporting both parallelization and hardware acceleration. Our generalizable learning interface provides an easy-to-use integration with SOTA MARL libraries (e.g., JaxMARL). In addition, JaxRobotarium includes eight standardized coordination scenarios, including four novel scenarios that bring established MARL benchmark tasks (e.g., RWARE and Level-Based Foraging) to a realistic robotics setting. We demonstrate that JaxRobotarium retains high simulation fidelity while achieving dramatic speedups over baseline (20x in training and 150x in simulation), and provides an open-access sim2real evaluation pipeline through the Robotarium testbed, accelerating and democratizing access to multi-robot learning research and evaluation. Our code is publicly available on `github`.

Keywords: Multi-Robot Learning, Sim2Real Deployment, Benchmarking, Jax

1 Introduction

Traditional approaches to multi-robot coordination have a rich history and offer robust solutions to standardized problems across a variety of domains [1, 2, 3, 4]. However, these approaches often rely on explicit specification of generalizable coordination mechanisms, which either requires considerable expertise or is virtually impossible in complex unstructured domains. Learning-based techniques, such as multi-robot reinforcement learning (MRRL), are emerging as promising alternatives that can offer a generalizable recipe to enable effective multi-robot coordination across various applications (e.g., scheduling [5], warehouse automation [6], and autonomous driving [7, 8]).

Despite its potential, learning-based algorithms are seldom deployed and evaluated on physical multi-robot platforms. Even when physical deployments are undertaken, they tend to rely on bespoke and closed platforms and testbeds, limiting reproducibility and benchmarking. In stark contrast, the multi-*agent* reinforcement learning (MARL) research community outside of robotics benefits from standardized and structured benchmarks, simulation, and learning environments (e.g.,

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

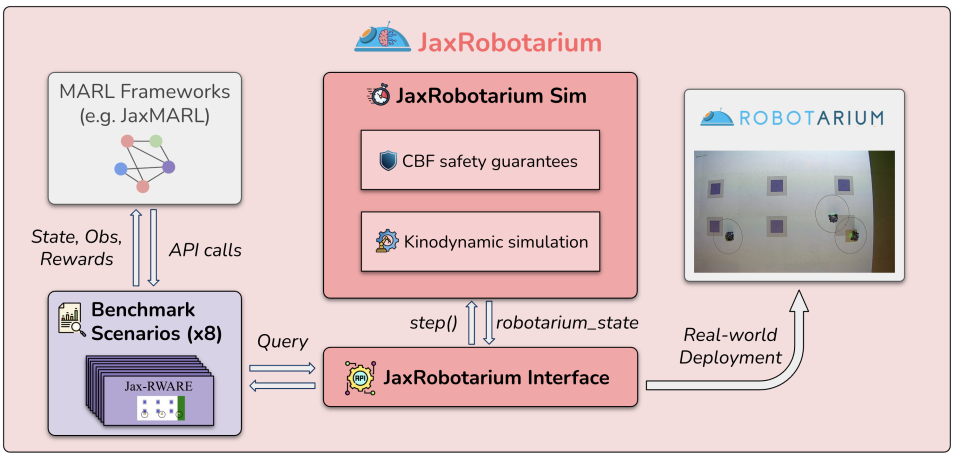

Figure 1: Overview of JaxRobotarium architecture. See Figure 5 in Appendix for additional details.

Multi-Agent Particle Environment (MPE) [9], StarCraft Multi-Agent Challenge (SMAC) [10], and JaxMARL [11]). However, these environments often abstract away robotics-relevant characteristics (e.g., robot dynamics and collision avoidance) and result in a considerable sim2real gap [8], limiting their ability to faithfully evaluate learning approaches tailored to multi-robot systems.

MARBLER [12] is a recent framework that attempts to meet this critical need. MARBLER acts as a bridge between the standardized infrastructure designed for MARL algorithms and the Robotarium [13] (an open and remotely-accessible multi-robot testbed). However, MARBLER does not support parallelization and execution on GPU/TPU, resulting in prohibitively slow training speeds that hinder adoption and comprehensive evaluations.

In this work, we contribute JaxRobotarium, an end-to-end open-source Jax-based platform designed to significantly accelerate the training and deployment of MRRL algorithms while maintaining ease-of-use and open access to hardware-based evaluations (see Fig. 1).

JaxRobotarium includes three key contributions. First, we contribute a new Jax-based *simulator* for the Robotarium that is designed from the ground-up and modeled after the existing Robotarium Python Simulator (RPS). Our new simulator supports parallelization and GPU/TPU execution, enabling dramatic improvements in efficiency by reducing the time and computational overhead typically associated with learning multi-robot coordination in realistic settings. Second, we contribute a new intuitive *interface* to facilitate integration with existing MARL libraries, such as JaxMARL [11], allowing researchers to leverage and build upon existing algorithmic implementations with minimal setup. Third, we contribute a new unified *benchmark* for MRRL that currently supports 8 representative multi-robot coordination scenarios, with flexible tools to rapidly create new ones or customize the existing ones. The unique benefits of JaxRobotarium include:

- Dramatic improvements in efficiency of training (up to 20x) and simulating (up to 150x) MRRL policies, supporting parallelization and execution on GPU/TPUs.
- Support for realistic simulation of robots, adhering to their dynamic constraints and safety guarantees obtained from barrier certificates.
- A standardized set of multi-robot coordination scenarios implemented in Jax with real-world counterparts for rapid benchmarking.
- An open-source, no-cost platform that provides any user in the world with access to training and real-world deployment of MRRL algorithms.

We validate the utility and benefits of JaxRobotarium through extensive efficiency evaluations, algorithm benchmarking, and comprehensive sim2real transfer experiments. We demonstrate that, compared to MARBLER [12], JaxRobotarium simulates robot trajectories up to 150 times faster and trains multi-robot policies up to 20 times faster. We benchmarked four MARL algorithms on the 8 scenarios provided by JaxRobotarium, revealing new insights into the algorithms' utility across

different multi-robot coordination scenarios. We systematically studied the sim2real gap of JaxRobotarium using data from over 200 real-world deployments of multi-robot policies that were entirely trained in simulation. We note that JaxRobotarium is not a panacea for the sim2real gap. Rather, JaxRobotarium helps study and reduce the sim2real gap by enabling rapid training and deployment. Our analysis reveals that while many task-algorithm combinations result in negligible performance differences when deployed on hardware, certain combinations of MARL algorithms and task characteristics exhibit noticeable performance differences. However, we show that such differences can be largely mitigated through simple randomization techniques such as noisy actions. We believe JaxRobotarium will serve as a powerful tool for the multi-robot systems community to advance the development, benchmarking, and sim2real transfer of learning algorithms.

2 Related Work

The platforms to facilitate learning multi-robot coordination strategies should have the following characteristics: (i) support efficient training and evaluation of policies [11, 14]; (ii) have realistic simulations of robots [8, 13], taking into account their dynamic constraints and the need for lower-level collision avoidance; (iii) have a convenient interface with common MARL algorithms and benchmarking tasks [11, 12, 15]. Centering around these key requirements, we evaluate and compare available MRRL and MARL frameworks in Table 1. The Multi-Agent Particle Environment (MPE) [9] is a widely used framework for evaluating MARL algorithms on a diverse suite of 2D tasks. While MPE was enhanced by successors including VMAS [14], BenchMARL [15], and JaxMARL [11], none of them realistically reflects the dynamics of real robots, impeding straightforward sim2real evaluation of MRRL policies. IsaacGym [16] and IsaacLab [17] offer high-fidelity simulation but are geared toward single-robot learning across varied embodiments. Multi-robot learning in these frameworks requires additional infrastructure for integration with MARL libraries, collision avoidance, benchmark tasks, and a sim2real pipeline, none of which are natively provided. Thus, to better evaluate MRRL policies, researchers must build task-specific test platforms [18, 19, 20] or reproduce previously proposed test beds, such as Cambridge RoboMaster [21]. As a contrastive line of efforts, the Robotarium [13] is a publicly available, multi-robot testbed that supports testing the coordination of up to 20 GRITSBots [22] robots. The Robotarium provides a simulator with implemented collision avoidance using control barrier functions [23] and offers sim2real evaluation through its hardware test bed. It is the only publicly available and free-to-use multi-robot hardware testbed in the world, and has been widely used for multi-robot research in areas including task allocation [24], heterogeneous coordination [25], motion planning [26], and safety [27]. To bridge the gap between MARL and a real multi-robot testbed, [12] contributes MARBLER, which is an open platform that promotes the comprehensive evaluation of MRRL algorithms on the Robotarium. Unfortunately, MARBLER only runs on the CPU without parallelization, resulting in prolonged training and tedious development cycles. JaxRobotarium builds upon the philosophy of MARBLER, marrying it with the computational efficiency offered by Jax and GPU parallelization.

Platform	**Train On GPU/TPU**	**Robot Dynamics**	**Safety Guarantee**	**Sim2Real Capabilities**	**MARL Support**	**Open Access Hardware**
MARL Platforms (MPE [9], SMAC [10], VMAS [14], BenchMARL [15], JaxMARL [11], Google Research Football [28])	✓	✗	✗	✗	✓	N/A
IsaacGym [16], IsaacLab [17]	✓	✓	✗	✓	✗	✗
Cambridge RoboMaster [21]	✓	✓	✗	✓	✓	✗
MuSHR [19]	✓	✓	✓	✓	✗	✗
Duckietown [20, 8]	✓	✓	✗	✓	✓	✗
Robotarium [13]	✗	✓	✓	✓	✗	✓
MARBLER [12]	✗	✓	✓	✓	✓	✓
JaxRobotarium (ours)	✓	✓	✓	✓	✓	✓

Table 1: JaxRobotarium versus existing frameworks for multi-robot/multi-agent learning.

3 JaxRobotarium Platform

JaxRobotarium is designed to achieve the training speeds of Multi-Agent RL in Multi-Robot RL while offering standardized, public, and free sim2real evaluation through the Robotarium. To this end, JaxRobotarium consists of the following key components,

1. A Jax-based simulator for the Robotarium, compatible with parallelization and GPU/TPU execution.
2. A platform for Multi-Robot RL, which we integrated with JaxMARL [11] to enable training with several SOTA MARL algorithms.
3. An implementation of all four MARBLER tasks, and four newly implemented tasks.
4. A direct deployment pipeline for sim2real evaluation on the Robotarium.

We provide a full overview of the architecture in the Appendix (Figure 5).

4 Jax-based Robotarium Simulator

4.1 Design

We contribute `Jax-RPS`, a Jax simulator for the Robotarium. `Jax-RPS` uses a unicycle dynamics model to faithfully simulate the behavior of the Robotarium's GRITSBots [29]. To control the GRITSBots, `Jax-RPS` implements single integrator and unicycle controllers to drive robots to desired positions/poses. To enforce the safety constraints often required in real-world deployment, `Jax-RPS` implements control barrier certificates for collision avoidance, which solve the quadratic program described in [30]. Finally, all functionality in `Jax-RPS` is designed to be compatible with Jax's `jit` compilation for accelerated execution and `vmap` for parallelization. Overall, this results in `Jax-RPS` being optimized for data-driven paradigms, such as Multi-Agent RL, while maintaining its faithfulness to real-world conditions and deployment considerations.

4.2 Evaluation

To validate the faithfulness and optimization of `Jax-RPS`, we construct the Random Waypoint Navigation scenario where four robots must each navigate to a randomly selected waypoint before being assigned a new one. We conduct the following experiments:

1. We measure faithfulness by simulating Random Waypoint Navigation for 10K timesteps and computing the difference in robot trajectories between `Jax-RPS` and `RPS` across unicycle position and pose controllers. We report mean wall time and trajectory mismatch with associated standard error averaged across 30 trials in Table 2.
2. We measure optimization performance by simulating Random Waypoint Navigation for 100K timesteps, where the timesteps are collected across multiple environments in parallel. We modified `RPS` to support multi-processing for parallelization. We report mean wall time across the number of environments with associated standard deviation averaged over 30 trials in Figure 2.

In all trials, we randomize the order of the controller-simulator conditions to avoid ordering biases. The above experiments were run on an M4 Pro CPU and 24 GB of RAM.

As seen in Table 2, `Jax-RPS` simulates trajectories that are identical to `RPS` with a 140- 150x speedup over `RPS` when run on CPU. Furthermore, in Figure 2 we observe `Jax-RPS` continues to benefit from increased parallelization, with its wall time to collect 100K timesteps monotonically decreasing as the number of environments increases. In contrast, `RPS` suffers slightly at higher degrees of parallelization, where the overhead of managing the multiple processes outweighs the benefit of executing the environments in parallel.

5 Benchmark Scenarios

We provide a diverse set of tasks to train on within JaxRobotarium. First, we re-implement the original MARBLER tasks in our framework, as described below.

- *Arctic Transport*: Two drone robots that observe their surrounding terrain must help guide two traversal robots, one faster on ice and slower on water, and vice versa. The team is successful if both traversal robots reach the goal zone.

Controller	Simulator	Wall Time (ms ↓)	Trajectory Error (↓)
Position	RPS	1425.28 ± 4.79	N/A
	Jax-RPS	**9.18 ± 0.10**	0.00 ± 0.00
Pose	RPS	1713.87 ± 5.39	N/A
	Jax-RPS	**11.42 ± 0.07**	0.00 ± 0.00

Table 2: Jax-RPS simulates the same trajectories significantly faster than RPS. Trajectory errors are computed for Jax-RPS against the trajectory for the identical scenario initial conditions simulated in RPS.

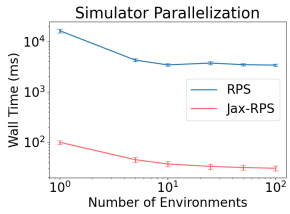

Figure 2: Jax-RPS maintains 100x or more speed up over RPS across increasing degrees of parallelization.

- *Predator Capture Prey → Discovery*: Sensing robots that sense landmarks within the environment collaborate with tagging robots that tag landmarks within the environment. Task performance is measured by the number of landmarks tagged. We note that this task has been renamed to Discovery for clarity.
- *Material Transport*: Robots with varying speeds and capacities must collaborate to unload two material depots, one further with less material, and one closer with more material. Task performance is measured by the amount of material remaining to be unloaded.
- *Warehouse*: Robots assigned to a zone type must collectively maximize the number of deliveries between their assigned zone type. Task performance is measured by the number of deliveries

In addition to the re-implementations above, we designed and implemented four new tasks:

- *Navigation (MAPF)*: Robots must navigate to independent goals within the environment. The team is successful if all robots are on their goals by the end of the rollout.
- *Foraging*: Inspired by grid world Level-Based Foraging [31], robots with varying levels of foraging ability must collaborate to forage all resources in the environment. Task performance is measured by the number of resources foraged.
- *Continuous-RWARE*: Inspired by the grid world RWARE [32], robots must collaborate to fulfill dropoffs of the requested shelves. Task performance is measured by the number of requested shelves dropped off.
- *Predator-Prey*: Robots must collaborate to tag a more agile prey, where the prey is controlled by the heuristic proposed in [33]. Task performance is measured by the number of tags on the prey.

Visualizations of all scenarios can be seen in Figure 4, and detailed descriptions are in Appendix B.

6 Jax-Robotarium Interface

6.1 Design

To enable the training and deployment of multi-robot policies with our simulator, we build a learning interface consisting of 5 main components:

- `RobotariumEnv` is the main interface structuring input to and output from Jax-RPS. It maintains a generalizable representation of the environment state for the Robotarium and provides visualization utilities to construct a consistent base visualization across all tasks.
- `HetMananger` provides an interface for injecting heterogeneity into multi-robot environments. It supports common heterogeneous representations: robot IDs, class IDs, capability vectors, and can be extended to include more. It manages the robot's awareness of heterogeneity, where the observation space can be configured to include any representation.
- `ControllerManager` handles translating policy actions into commands to the robots. If a policy is designed to output waypoints, these waypoints are processed by the configured controller into

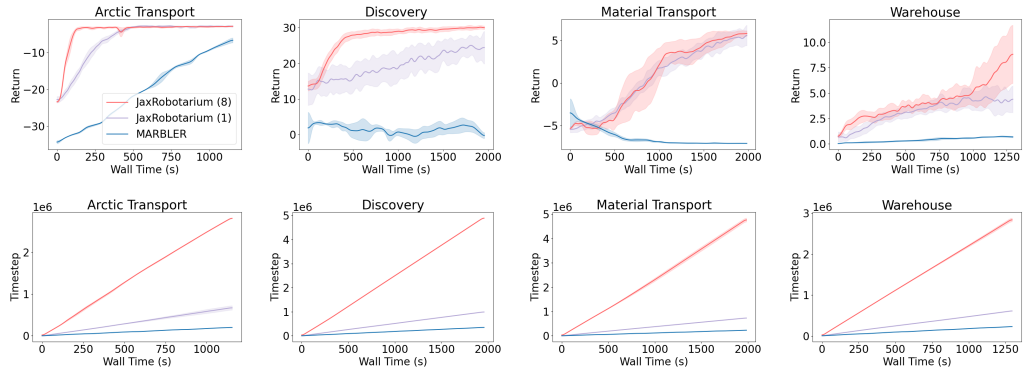

Figure 3: (Top) We plot the mean returns achieved in both platforms against the minimum training wall time. (Bottom) We plot the time taken to simulate timesteps across both platforms. We find that JaxRobotarium is significantly more efficient in training multi-robot policies than MARBLER. Results depict smoothed mean (solid line) with standard deviation (shaded).

unicycle velocity commands. If barrier functions are enabled, the unicycle velocity commands are then modified to guarantee collision avoidance.

- `Scenario` interface is a standard structure for defining new tasks within JaxRobotarium, aligned with the structures used in other learning frameworks such as VMAS [14], JaxMARL [11], and MARBLER [12].
- `Deploy` is a module that streamlines deployment to the Robotarium by compiling all the necessary files for an experiment into a folder, which can then be directly uploaded and run on the Robotarium test bed. Since the Robotarium test bed does not support Jax, the deployment module performs model conversion and file sanitation to ensure compatibility.

We choose JaxMARL as the MARL framework, given its SOTA performance and extensive baseline algorithm implementations. However, note that any Jax multi-agent learning framework designed for the output structure of the `Scenario` interface is compatible with JaxRobotarium.

6.2 Evaluation

To validate the utility of our framework, we compare the training performance and efficiency of JaxRobotarium with no parallelization (1 env) and parallelization (8 envs) against MARBLER on the four MARBLER scenarios: Arctic Transport, Discovery, Material Transport, and Warehouse. We report the episodic returns and timesteps simulated against the wall time of the fastest run, averaged across 3 seeds. All experiments are run on an Intel i7-12700KF and NVIDIA RTX 3070.

In Figure 3, we find that JaxRobotarium can train for up to 20x more timesteps and achieve consistently higher returns compared to MARBLER given the same amount of time. As expected, the support for GPU execution, parallelization, and just-in-time compilation significantly accelerates training across all tasks, bridging the gap between the Robotarium and modern MARL algorithms.

7 Benchmarking Learning Algorithms

7.1 Experiment setup

We demonstrate the benchmarking and sim2real evaluation capabilities of JaxRobotarium by training 5 policies in parallel for all tasks using the following SOTA MARL approaches:

- PQN [34]: a recent multi-agent Q-Learning approach designed to perform Q-updates efficiently by collecting non-correlated transitions in parallel across vectorized environments (we use 256), eliminating the need for a replay buffer.

- QMIX [35]: an established multi-agent Q-learning algorithm where decentralized Q-networks predict agent Q-values aggregated into a global Q-value by a hypernetwork-generated centralized critic conditioned on the global state.
- MAPPO [36]: an established multi-agent policy-gradient algorithm that learns decentralized action policies, regularized by a centralized critic conditioned on the global state.
- IPPO [36]: a multi-agent policy-gradient algorithm that learns decentralized action policies. Unlike MAPPO, IPPO uses a decentralized critic conditioned on the agent's local observation.

See Appendix A for further training details. We report mean episodic reward, collisions, and a task-performance metric with associated standard errors. Metrics are averaged across the 5 trained policies for 100 episodes each. For real-world experiments, we deploy the best-performing policy on the Robotarium hardware testbed and report metrics averaged across 5 unique instantiations, for a total of 160 experiments.

7.1.1 Results

Learning: In general, we do not find any single algorithm to consistently outperform all others across our 8 scenarios. Consistent with recent findings [34], we find that Q-Learning with PQN performs comparably or even outperforms Q-Learning with QMIX, with Material Transport being a notable exception (see Table 3). Given that PQN achieves this performance without maintaining a replay buffer, these results support PQN as a promising alternative to established Q-Learning methods for multi-robot settings, in which maintaining a large replay buffer could be intractable due to high-dimensional state spaces and combinatorial space complexity.

Scenario	Metric	Training-Seed Benchmark			
		PQN	**QMIX**	**MAPPO**	**IPPO**
Arctic Transport	Reward	-3.6 ± 0.04	-3.4 ± 0.02	-3.5 ± 0.04	-3.5 ± 0.04
	Success Rate	1.0 ± 0.00	1.0 ± 0.00	1.0 ± 0.00	1.0 ± 0.00
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Discovery	Reward	26.6 ± 0.20	29.8 ± 0.14	24.1 ± 0.32	30.0 ± 0.15
	Landmarks Tagged	5.1 ± 0.04	5.6 ± 0.03	4.5 ± 0.06	5.7 ± 0.03
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Material Transport	Reward	-5.0 ± 0.06	4.6 ± 0.07	1.3 ± 0.08	1.3 ± 0.09
	Material Left	65.1 ± 0.75	4.2 ± 0.31	15.7 ± 0.72	17.1 ± 0.71
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Warehouse	Reward	8.4 ± 0.20	6.1 ± 0.16	3.9 ± 0.02	8.0 ± 0.17
	Deliveries	1.7 ± 0.06	0.9 ± 0.05	0.0 ± 0.00	1.7 ± 0.05
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Navigation (MAPF)	Reward	-18.3 ± 0.39	-17.0 ± 0.17	-13.7 ± 0.00	-12.7 ± 0.16
	Success Rate	0.6 ± 0.39	0.8 ± 0.17	1.0 ± 0.00	1.0 ± 0.16
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Foraging	Reward	1.8 ± 0.12	4.5 ± 0.13	0.3 ± 0.06	7.1 ± 0.21
	Resources Foraged	0.3 ± 0.02	0.8 ± 0.03	0.1 ± 0.01	1.3 ± 0.01
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Predator Prey	Reward	58.8 ± 2.45	6.3 ± 0.65	11.0 ± 1.49	46.9 ± 3.15
	Tags	5.9 ± 0.25	0.6 ± 0.06	1.1 ± 0.15	4.7 ± 0.31
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Continuous RWARE	Reward	5.9 ± 0.78	3.6 ± 0.66	0.0 ± 0.59	1.1 ± 0.53
	Shelf Dropoffs	5.9 ± 0.02	3.6 ± 0.02	0.0 ± 0.00	1.1 ± 0.01
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00

Table 3: Performance metrics across all training seeds in simulation.

Further, we note that independent learning (IPPO) is surprisingly stronger in our multi-robot coordination scenarios compared to its centralized counterpart (MAPPO). We believe this is likely due to the limited partial observability in the configured benchmark scenarios (e.g., robots can generally observe their neighbors except in Navigation). This suggests that in settings without significant partial observability, independent learning is a simpler and effective alternative to centralized critic approaches, in line with prior results comparing MAPPO and IPPO on StarCraft tasks [36].

Sim2Real: As one would expect, we found reasonable gaps between the simulated and real world runs for most tasks across methods (see Table 4). Some of this gap could be attributed to only having limited real-world runs and the stochasticity of the real world compared to the deterministic simulator. However, two of our new tasks (Predator Prey and Continuous-RWARE) resulted in surprisingly large sim2real performance gaps, particularly for QMIX and PQN. Our qualitative analysis of the rollouts revealed that these tasks contain inflection points that can exacerbate modest differences into large differences in the environment and robot behavior over time. For instance, small positional changes to the robots in Predator-Prey can sometimes elicit qualitatively different prey behaviors which cascade into dramatic differences between the simulated and real-world rollouts. In contrast, existing MARBLER tasks are less sensitive to such variations, as robots mostly interact with a static environment. The observed gaps in the new tasks underscore the importance

Scenario	Metric	Simulated Experiments				Real-World Experiments			
		PQN	QMIX	MAPPO	IPPO	PQN	QMIX	MAPPO	IPPO
Arctic Transport	Reward	-3.5 ± 0.22	-3.5 ± 0.18	-3.7 ± 0.22	-3.4 ± 0.18	-2.6 ± 0.13	-3.0 ± 0.13	-3.8 ± 1.16	-2.5 ± 0.09
	Success Rate	1.0 ± 0.00	1.0 ± 0.00	1.0 ± 0.00	1.0 ± 0.00	1.0 ± 0.00	1.0 ± 0.00	0.8 ± 0.18	1.0 ± 0.00
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Discovery	Reward	26.1 ± 1.88	25.2 ± 1.16	23.6 ± 2.68	29.3 ± 1.25	24.7 ± 1.07	26.1 ± 1.48	23.9 ± 2.55	29.3 ± 1.25
	Landmarks Tagged	5.0 ± 0.40	4.8 ± 0.18	4.6 ± 0.54	5.6 ± 0.22	4.6 ± 0.22	5.0 ± 0.27	4.6 ± 0.45	5.6 ± 0.22
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Material Transport	Reward	-6.1 ± 0.04	3.8 ± 0.49	0.4 ± 0.85	0.4 ± 0.89	-6.1 ± 0.04	4.2 ± 0.72	-0.3 ± 0.89	1.6 ± 0.67
	Material Left	68.3 ± 2.64	10.9 ± 2.28	37.2 ± 9.88	26.7 ± 6.44	66.3 ± 2.64	5.5 ± 2.64	35.7 ± 10.96	12.4 ± 3.22
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Warehouse	Reward	6.8 ± 0.58	10.0 ± 1.48	4.0 ± 0.00	5.6 ± 1.25	8.2 ± 1.30	9.2 ± 1.12	3.8 ± 0.18	4.6 ± 1.03
	Deliveries	1.2 ± 0.18	2.2 ± 0.45	0.0 ± 0.00	1.2 ± 0.31	1.6 ± 0.36	1.8 ± 0.31	0.0 ± 0.00	0.8 ± 0.31
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Navigation (MAPF)	Reward	-19.5 ± 2.33	-14.8 ± 2.06	-14.3 ± 2.64	-12.8 ± 2.10	-17.8 ± 2.06	-14.7 ± 2.01	-13.7 ± 2.59	-12.2 ± 1.92
	Success Rate	0.0 ± 0.00	1.0 ± 0.00	1.0 ± 0.00	1.0 ± 0.00	0.2 ± 0.18	1.0 ± 0.00	1.0 ± 0.00	1.0 ± 0.00
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Foraging	Reward	6.2 ± 1.07	6.2 ± 1.07	1.0 ± 0.89	6.6 ± 2.41	5.2 ± 0.18	5.2 ± 0.18	2.0 ± 1.07	11.0 ± 0.00
	Resources Foraged	1.2 ± 0.18	1.2 ± 0.18	0.2 ± 0.18	1.2 ± 0.45	1.0 ± 0.0	1.0 ± 0.00	0.4 ± 0.22	2.0 ± 0.00
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Predator Prey	Reward	44.0 ± 17.13	4.0 ± 2.19	14.0 ± 8.77	52.0 ± 16.37	14.0 ± 5.37	36.0 ± 16.64	6.0 ± 3.58	28.1 ± 13.64
	Tags	4.4 ± 1.70	0.4 ± 0.22	1.4 ± 0.89	5.2 ± 1.65	1.4 ± 0.54	3.6 ± 1.65	0.6 ± 0.36	3.8 ± 1.21
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Continuous RWARE	Reward	1.0 ± 0.67	0.8 ± 0.54	0.0 ± 0.00	0.0 ± 0.00	0.8 ± 0.31	6.2 ± 2.68	0.0 ± 0.00	0.0 ± 0.00
	Shelf Dropoffs	1.0 ± 0.67	0.8 ± 0.54	0.0 ± 0.00	0.0 ± 0.00	0.8 ± 0.31	6.2 ± 2.68	0.0 ± 0.00	0.0 ± 0.00
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00

Table 4: Performance metrics across scenarios for simulated and real-world experiments.

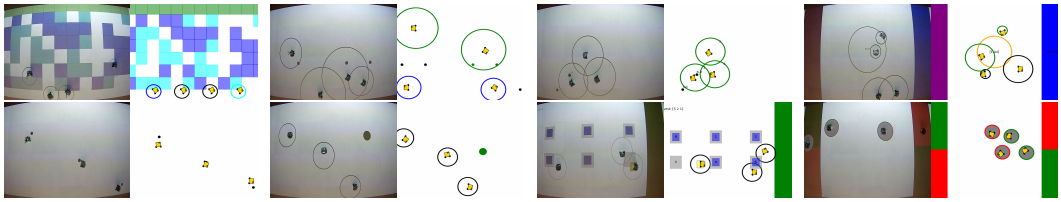

Figure 4: Scenarios (real on left and sim on right) from left to right and top to bottom as follows: Arctic Transport, Discovery, Foraging, Material Transport, Navigation, Predator Prey, RWARE, Warehouse. For detailed descriptions of each scenario, please see Appendix B.

of hardware evaluation, especially in multi-robot scenarios where task outcomes are closely tied to interaction dynamics. The ability to accurately simulate single robots does not necessarily imply accurate multi-robot simulations.

Domain Randomization: Inspired by the above observations, we applied simple domain randomization through action noise to learn a more robust policy. We found that training with action noise can reduce the sim2real gap in Predator Prey for both QMIX and PQN and improve task performance in simulation (see Table 5).

Scenario	Metric	Simulated Experiments		Real-World Experiments	
		PQN	QMIX	PQN	QMIX
Predator Prey	Returns	30.0 ± 6.33	3.3 ± 1.21	16.0 ± 5.06	26.0 ± 11.33
	Tags	3.0 ± 0.62	0.3 ± 0.13	1.6 ± 0.52	2.6 ± 1.14
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00
Predator Prey w/ DR	Returns	66.7 ± 3.23	16.7 ± 6.09	46.7 ± 9.66	18.0 ± 4.60
	Tags	6.7 ± 0.34	1.7 ± 0.62	4.7 ± 0.96	1.8 ± 0.46
	Collisions	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00	0.0 ± 0.00

Table 5: Predator Prey performance with and without domain randomization (3 initial conditions, 5 trials per initial condition).

8 Conclusion

We contribute JaxRobotarium, an accelerated benchmark platform for MRRL with free, publicly available, and standardized sim2real evaluation through the Robotarium hardware test bed. Our benchmark experiments demonstrate up to a 150x speed up in simulation, and a 20x speed up in training multi-robot policies compared to baseline. We demonstrate the utility of our platform by evaluating 4 MARL algorithms on 8 multi-robot coordination scenarios, with a large-scale sim2real evaluation consisting of over 200 real world deployments. We hope JaxRobotarium will be a valuable resource for accelerating the development, benchmarking, and sim2real evaluation of MRRL algorithms and architectures.

9 Limitations

We design our platform for sim2real evaluation through the Robotarium testbed, so we only simulate the Robotarium GRITSBots and we do not simulate observations from more complex perception modules, such as images or LiDAR scans.

The decision to utilize the Robotarium introduces additional limitations. Since the robots are homogeneous, heterogeneity has to be artificially injected, such as limiting a robot's velocity or waypoint step size, or through introducing scenario-specific heterogeneity that is recognized by the scenario's step function, such as obeying the foraging level of robots when deciding if a resource is successfully foraged. Furthermore, the Robotarium hardware test bed does not support Jax libraries, so the quadratic problem solver used for barrier certificates in deployment and the solver used in Jax simulation (chosen differently for compatibility with `jit` and `vmap`) are not aligned. We observe that underlying differences in the solvers can result in different proposed solutions to the same quadratic program.

We note that our platform cannot be considered a replacement for training in high-fidelity robotic simulation environments such as IsaacGym [16] and IsaacLab [17]. Rather, JaxRobotarium is designed to be a turn-key platform for simulation-training-deployment tailored for MRRL research, offering native integrations (e.g., JaxMARL), standardized benchmarks (e.g., RWARE adaptations), and a ready-to-use zero-cost sim2real pipeline.

Finally, in our training evaluations, we only use an RNN-based policy architecture since it is widely implemented to overcome the challenges of partial observability and long-horizon reasoning [35, 36, 11, 37, 12], and do not evaluate on MLP architectures [14, 15], GNN architectures [38, 39, 40, 41], or more recent Hyper-Network architectures [42, 43]. Additionally, we train with limited partial observability in most scenarios, see Appendix B for exact details. Future work could benchmark these policy architectures within JaxRobotarium with stronger partial observability.

References

[1] N. Baras and M. Dasygenis. An algorithm for routing heterogeneous vehicles in robotized warehouses. In *2019 Panhellenic Conference on Electronics & Telecommunications (PACET)*, pages 1–4, 2019. doi:10.1109/PACET48583.2019.8956244.

[2] J. J. Roldán, P. Garcia-Aunon, M. Garzón, J. De León, J. Del Cerro, and A. Barrientos. Heterogeneous multi-robot system for mapping environmental variables of greenhouses. *Sensors*, 16(7):1018, 2016.

[3] B. Rao, H. F. Durrant-Whyte, and J. Sheen. A fully decentralized multi-sensor system for tracking and surveillance. *The International Journal of Robotics Research*, 12(1):20–44, 1993.

[4] J. P. Queralta, J. Taipalmaa, B. C. Pullinen, V. K. Sarker, T. N. Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund. Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision. *Ieee Access*, 8:191617–191643, 2020.

[5] Z. Wang, C. Liu, and M. Gombolay. Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints. *Autonomous Robots*, 46(1):249–268, 2022.

[6] A. Krnjaic, R. D. Steleac, J. D. Thomas, G. Papoudakis, L. Schäfer, A. W. Keung To, K.-H. Lao, M. Cubuktepe, M. Haley, P. Börsting, and S. V. Albrecht. Scalable multi-agent reinforcement learning for warehouse logistics with robotic and human co-workers. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 677–684, 2024. doi:10.1109/IROS58592.2024.10802813.

[7] S. Bhalla, S. Ganapathi Subramanian, and M. Crowley. Deep multi agent reinforcement learning for autonomous driving. In *Canadian Conference on Artificial Intelligence*, pages 67–78. Springer, 2020.

[8] 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)*, pages 8814–8820. IEEE, 2022.

[9] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[10] B. Ellis, J. Cook, S. Moalla, M. Samvelyan, M. Sun, A. Mahajan, J. Foerster, and S. Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[11] A. Rutherford, B. Ellis, M. Gallici, J. Cook, A. Lupu, G. Ingvarsson, T. Willi, A. Khan, C. S. de Witt, A. Souly, S. Bandyopadhyay, M. Samvelyan, M. Jiang, R. T. Lange, S. Whiteson, B. Lacerda, N. Hawes, T. Rocktaschel, C. Lu, and J. N. Foerster. Jaxmarl: Multi-agent rl environments in jax. (arXiv:2311.10090), Dec. 2023. doi:10.48550/arXiv.2311.10090. URL http://arxiv.org/abs/2311.10090. arXiv:2311.10090 [cs].

[12] R. Torbati, S. Lohiya, S. Singh, M. S. Nigam, and H. Ravichandar. Marbler: An open platform for standardized evaluation of multi-robot reinforcement learning algorithms, 2023.

[13] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt. The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems. *IEEE Control Systems Magazine*, 40(1): 26–44, 2020. doi:10.1109/MCS.2019.2949973.

[14] M. Bettini, R. Kortvelesy, J. Blumenkamp, and A. Prorok. Vmas: A vectorized multi-agent simulator for collective robot learning. In *International Symposium on Distributed Autonomous Robotic Systems*, pages 42–56. Springer, 2022.

[15] M. Bettini, A. Prorok, and V. Moens. Benchmarl: Benchmarking multi-agent reinforcement learning. *Journal of Machine Learning Research*, 25(217):1–10, 2024. URL http://jmlr.org/papers/v25/23-1612.html.

[16] 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.

[17] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi:10.1109/LRA.2023.3270034.

[18] J. Chen, F. Deng, Y. Gao, J. Hu, X. Guo, G. Liang, and T. L. Lam. Multirobolearn: An open-source framework for multi-robot deep reinforcement learning. In *2023 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1–6. IEEE, 2023.

[19] S. S. Srinivasa, P. Lancaster, J. Michalove, M. Schmittle, C. Summers, M. Rockett, R. Scalise, J. R. Smith, S. Choudhury, C. Mavrogiannis, et al. Mushr: A low-cost, open-source robotic racecar for education and research. *arXiv preprint arXiv:1908.08031*, 2019.

[20] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang, D. Hoehener, S.-Y. Liu, M. Novitzky, I. F. Okuyama, J. Pazis, G. Rosman, V. Varricchio, H.-C. Wang, D. Yershov, H. Zhao, M. Benjamin, C. Carr, M. Zuber, S. Karaman, E. Frazzoli, D. Del Vecchio, D. Rus, J. How, J. Leonard, and A. Censi. Duckietown: An open, inexpensive and flexible platform for autonomy education and research. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1497–1504, 2017. doi:10.1109/ICRA.2017.7989179.

[21] J. Blumenkamp, A. Shankar, M. Bettini, J. Bird, and A. Prorok. The cambridge robomaster: An agile multi-robot research platform. *arXiv preprint arXiv:2405.02198*, 2024.

[22] D. Pickem, M. Lee, and M. Egerstedt. The gritsbot in its natural habitat - a multi-robot testbed. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4062–4067, 2015. doi:10.1109/ICRA.2015.7139767.

[23] L. Wang, A. D. Ames, and M. Egerstedt. Safety barrier certificates for collisions-free multi-robot systems. *IEEE Transactions on Robotics*, 33(3):661–674, 2017. doi:10.1109/TRO.2017.2659727.

[24] G. Notomista, S. Mayya, S. Hutchinson, and M. Egerstedt. An optimal task allocation strategy for heterogeneous multi-robot systems. In *2019 18th European Control Conference (ECC)*, pages 2071–2076, 2019. doi:10.23919/ECC.2019.8795895.

[25] E. Seraj, R. Paleja, L. Pimentel, K. M. Lee, Z. Wang, D. Martin, M. Sklar, J. Zhang, Z. Kakish, and M. Gombolay. Heterogeneous policy networks for composite robot team communication and coordination. *IEEE Transactions on Robotics*, 2024.

[26] D. Sun, J. Chen, S. Mitra, and C. Fan. Multi-agent motion planning from signal temporal logic specifications. *IEEE Robotics and Automation Letters*, 7(2):3451–3458, 2022.

[27] Y. Chen, A. Singletary, and A. D. Ames. Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach. *IEEE Control Systems Letters*, 5(1):127–132, 2020.

[28] K. Kurach, A. Raichuk, P. Stańczyk, M. Zając, O. Bachem, L. Espeholt, C. Riquelme, D. Vincent, M. Michalski, O. Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 4501–4510, 2020.

[29] D. Pickem, M. Lee, and M. Egerstedt. The gritsbot in its natural habitat-a multi-robot testbed. In *2015 IEEE International conference on robotics and automation (ICRA)*, pages 4062–4067. IEEE, 2015.

[30] Y. Emam, P. Glotfelter, and M. Egerstedt. Robust barrier functions for a fully autonomous, remotely accessible swarm-robotics testbed. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 3984–3990. IEEE, 2019.

[31] F. Christianos, L. Schäfer, and S. V. Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[32] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021. URL `http://arxiv.org/abs/2006.07869`.

[33] B. Peng, T. Rashid, C. S. de Witt, P.-A. Kamienny, P. Torr, W. Böhmer, and S. Whiteson. Facmac: Factored multi-agent centralised policy gradients. In *Advances in Neural Information Processing Systems*, 2021.

[34] M. Gallici, M. Fellows, B. Ellis, B. Pou, I. Masmitja, J. N. Foerster, and M. Martin. Simplifying deep temporal difference learning. *arXiv preprint arXiv:2407.04811*, 2024.

[35] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

[36] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. WU. The surprising effectiveness of ppo in cooperative multi-agent games. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24611–24624. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9c1535a02f0ce079433344e14d910597-Paper-Datasets_and_Benchmarks.pdf.

[37] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021. URL http://arxiv.org/abs/2006.07869.

[38] Q. Li, W. Lin, Z. Liu, and A. Prorok. Message-aware graph attention networks for large-scale multi-robot path planning. *IEEE Robotics and Automation Letters*, 6(3):5533–5540, 2021. doi:10.1109/LRA.2021.3077863.

[39] P. Howell, M. Rudolph, R. Torbati, K. Fu, and H. Ravichandar. Generalization of heterogeneous multi-robot policies via awareness and communication of capabilities. *arXiv preprint arXiv:2401.13127*, 2024.

[40] M. Bettini, A. Shankar, and A. Prorok. Heterogeneous multi-robot reinforcement learning. *arXiv preprint arXiv:2301.07137*, 2023.

[41] S. Kailas, S. Jain, and H. Ravichandar. Evaluating and improving graph-based explanation methods for multi-agent coordination. *arXiv preprint arXiv:2502.09889*, 2025.

[42] K. Fu, S. A. Jain, P. Howell, and H. Ravichandar. Capability-aware shared hypernetworks for flexible heterogeneous multi-robot coordination. *arXiv preprint arXiv:2501.06058*, 2025.

[43] K. ab Abebe Tessera, A. Rahman, and S. V. Albrecht. Hypermarl: Adaptive hypernetworks for multi-agent rl, 2025. URL https://arxiv.org/abs/2412.04233.

[44] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. URL https://arxiv.org/abs/1406.1078.

[45] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search*, volume 10, pages 151–158, 2019.

[46] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial intelligence*, 219:40–66, 2015.

[47] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

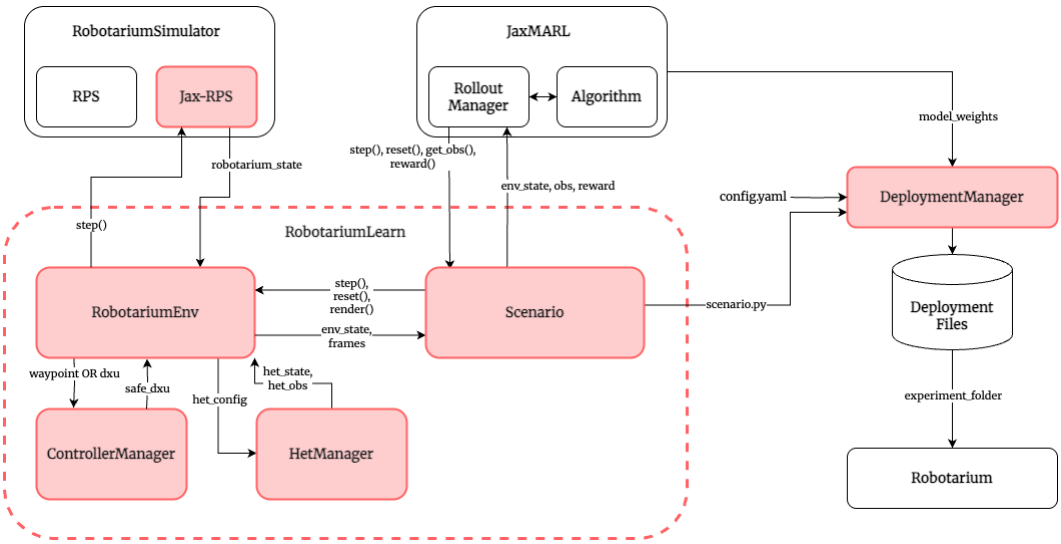

Figure 5: JaxRobotarium architecture, colored components are novel.

A Training Details

A.1 Policy Architecture

We use an RNN-based policy architecture across all of our training experiments. The specific implementations are all adopted from JaxMARL's baseline implementations [11] and use a Gated Recurrent Unit (GRU) [44].

- PQN: A 2 layer MLP with Layer Normalization before ReLU activations, followed by a GRU, followed by a linear layer.
- QMIX: A 1 layer MLP with ReLU activation, followed by a GRU, followed by a linear layer.
- MAPPO: A single layer MLP with ReLU activation, followed by a GRU, followed by a single layer MLP with ReLU activation, followed by a linear layer.
- IPPO: A single layer MLP with ReLU activation, followed by a GRU, followed by a single layer MLP with ReLU activation, followed by a linear layer.

For all algorithms, we train 5 policies in parallel across 5 unique seeds. Exact widths for the policies' hidden layers were chosen by ablating over the widths 128, 256, and 512 trained on a single seed. The best-performing width was then chosen for the final training runs. We report all policy widths in Table 6.

Scenario	PQN	QMIX	MAPPO	IPPO
Arctic Transport	256	256	128	128
Discovery	256	256	128	256
Material Transport	512	128	128	512
Warehouse	256	512	128	256
Navigation	256	512	128	512
Foraging	128	128	512	256
Predator Prey	128	512	256	512
Continuous RWARE	256	256	128	256

Table 6: Policy hidden width across scenarios and algorithms.

A.2 PQN

Parameter	Value
NUM_ENVS	256
NORM_INPUTS	False
NORM_TYPE	layer_norm
EPS_START	1.0
EPS_FINISH	0.1
EPS_DECAY	0.1
MAX_GRAD_NORM	1
NUM_MINIBATCHES	16
NUM_EPOCHS	4
LR	0.0005
LR_LINEAR_DECAY	False
GAMMA	0.9
LAMBDA	0.3

Table 7: PQN hyperparameters

We train with the JaxMARL tuned hyperparameters listed in Table 7.

A.3 QMIX

Parameter	Value
NUM_ENVS	8
BUFFER_SIZE	5000
BUFFER_BATCH_SIZE	32
MIXER_EMBEDDING_DIM	32
MIXER_HYPERNET_HIDDEN_DIM	128
MIXER_INIT_SCALE	0.001
EPS_START	1.0
EPS_FINISH	0.05
EPS_DECAY	0.1
MAX_GRAD_NORM	25
TARGET_UPDATE_INTERVAL	200
TAU	1.0
NUM_EPOCHS	1
LR	0.005
LEARNING_STARTS	10000
LR_LINEAR_DECAY	True
GAMMA	0.9

Table 8: QMIX hyperparameters

We train with the JaxMARL tuned hyperparameters listed in Table 8.

A.4 MAPPO

We train with the JaxMARL tuned hyperparameters listed in Table 9.

A.5 IPPO

We train with the JaxMARL tuned hyperparameters listed in Table 10.

B Scenario Details

As a general note for all scenarios, if heterogeneity awareness is configured, robots will additionally observe the configured heterogeneity representation. In our training experiments, the action space

Parameter	Value
LR	0.002
NUM_ENVS	16
HIDDEN_SIZE	128
UPDATE_EPOCHS	4
NUM_MINIBATCHES	4
GAMMA	0.99
GAE_LAMBDA	0.95
CLIP_EPS	0.2
SCALE_CLIP_EPS	False
ENT_COEF	0.01
VF_COEF	0.5
MAX_GRAD_NORM	0.5
ANNEAL_LR	True

Table 9: MAPPO hyperparameters

Parameter	Value
LR	0.0005
NUM_ENVS	16
UPDATE_EPOCHS	4
NUM_MINIBATCHES	2
GAMMA	0.99
GAE_LAMBDA	0.95
CLIP_EPS	0.3
SCALE_CLIP_EPS	False
ENT_COEF	0.01
VF_COEF	1.0
MAX_GRAD_NORM	0.5
ANNEAL_LR	True

Table 10: IPPO hyperparameters

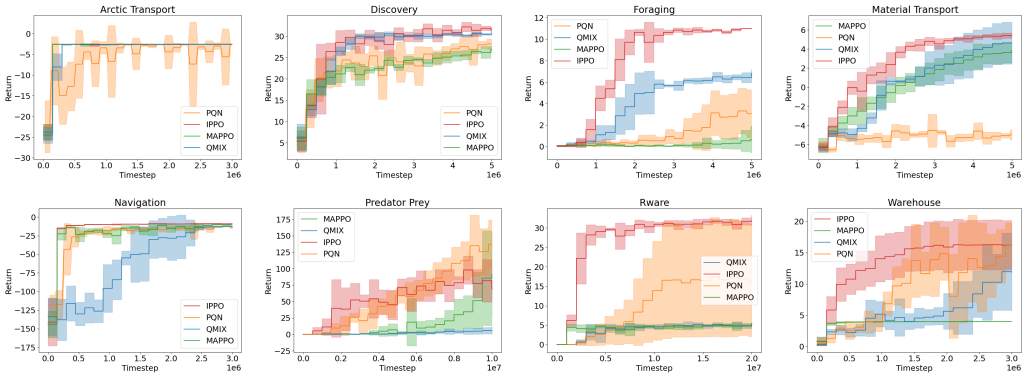

Figure 6: We plot the greedy returns throughout training, where all algorithms are trained for the same number of timesteps for each scenario

for all robots is waypoints that are followed by a unicycle position controller with barrier functions enabled for collision avoidance.

B.1 Arctic Transport

B.1.1 Description

Adopted from MARBLER, in this scenario, the ice and water robots must cross the terrain consisting of ice tiles (light blue), water tiles (dark blue), and ground tiles (white) to reach the goal zone (green).

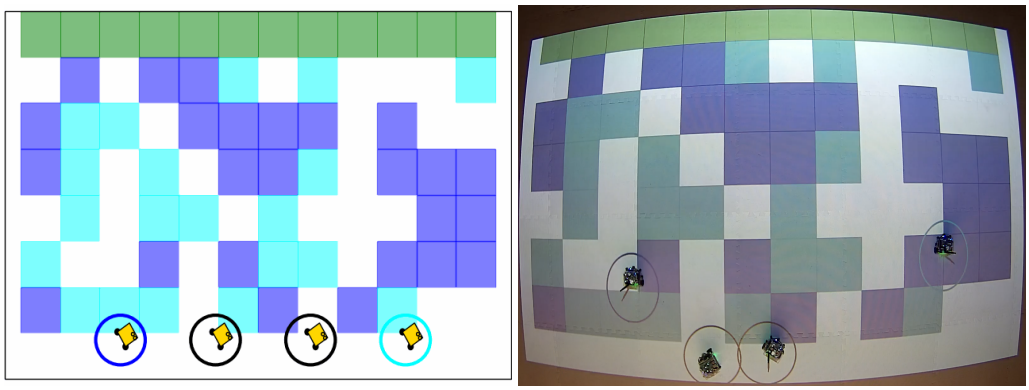

Figure 7: Arctic Transport in simulation (left) and real-world (right).

Ice and water robots move at the same speed across ground tiles, slow over tiles not corresponding to their type, and fast over tiles corresponding to their types. Ice and water robots observe their position, the position of other robots, and the current tile type they are occupying. Drone robots observe a 3x3 grid centered around their current position, and move at a uniform speed across all tiles. Their observations are directly communicated to the ice and water robots, such that they can help guide the ice and water robots. The team is rewarded for minimizing the distance of the ice and water robots to the goal zone, and penalized for every step taken where both robots are not in the goal zone.

B.1.2 Experiment Parameters

- Number of Robots: 4
- Max steps: 100
- Training timesteps: 3e6
- Controller
  - Controller: unicycle Position
  - Barrier Fn: enabled
- Normal Step: 0.2
- Fast Step: 0.3
- Slow Step: 0.1
- Heterogeneity
  - Type: class
  - Obs Type: class (robots observe their own class)
  - Values: [[1, 0, 0], [1, 0, 0], [0, 1, 0], [0, 0, 1]] (each entry indicates the type [drone, water, ice])
- Distance Reward: -0.05 $\times$ summed shortest distances of ice and water robots to the goal zone
- Step Reward: -0.10 $\times$ 0 if all robots are on their goal, otherwise 1
- Violation Reward: 0 $\times$ number of collisions

B.2 Discovery

B.2.1 Description

Renamed from Predator-Capture-Prey in MARBLER to better reflect the scenario objective, in this scenario, N sensing robots must collaborate with M tagging robots to tag K landmarks (black) in the environment. Sensing robots and tagging robots have configurable sensing radii. Robots observe their own location and the location of other robots. Initially, robot observations also contain a position outside the map for all landmarks, indicating they have not been sensed. When a landmark is within range of a sensing robot, it is considered sensed, the black marker is updated to green,

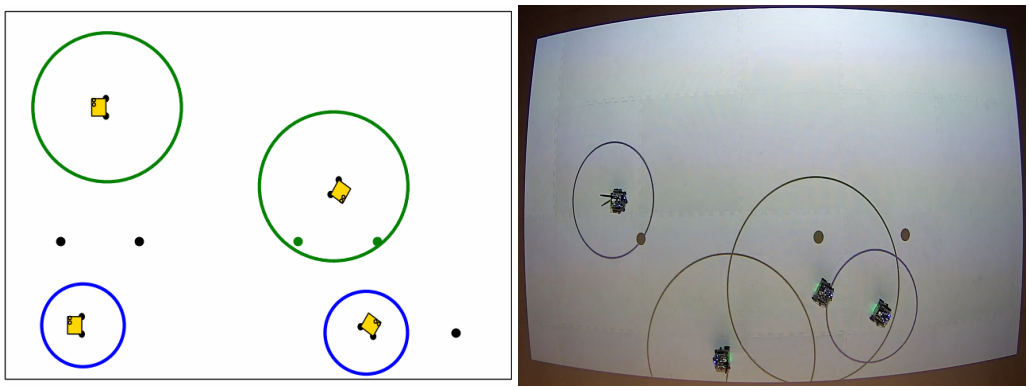

Figure 8: Discovery in simulation (left) and real-world (right).

and the location becomes available within the observation. When a landmark is within range of a tagging robot, the landmark is considered tagged and disappears from the visualization. The team is rewarded for landmarks sensed and tagged, and penalized for every step taken where all landmarks are not tagged. This scenario can scale to an arbitrary number of robots and landmarks.

B.2.2 Experiment Parameters

- Number of Robots: 4
- Sensing Robots: 2
- Tagging Robots: 2
- Max Steps: 100
- Training timesteps: 5e6
- Controller
  - Controller: unicycle position
  - Barrier Fn: enabled
- Heterogeneity
  - Type: capability set
  - Obs type: full capability set (robots observe the team's capability values)
  - Values: [[0.45, 0], [0.45, 0], [0, 0.25], [0, 0.25]] (each entry is [sensing radius, tag radius])
- Sensing Reward: 1 $\times$ landmarks sensed
- Tagging Reward: 5 $\times$ landmarks tagged
- Step Reward: -0.05 $\times$ 0 if all landmarks tagged, otherwise 1
- Violation Reward: 0 $\times$ number of collisions

B.3 Foraging

B.3.1 Description

Inspired by grid-world level-based foraging in [31], in this scenario, N robots with varying levels must collaborate to forage M resources (black markers) within the environment. Labels indicate the levels of robots and resources. Robots observe their position, the position of other robots, the location of resources, and their corresponding levels. For a resource to be considered foraged, it must be within the foraging radius of M robots, where the summed level of the M robots is greater than or equal to the level of the resource. When resources are foraged, they disappear from the visualization to indicate they have been collected. The team is rewarded for each resource foraged, scaled by the level of that resource. This scenario can scale to an arbitrary number of robots and resources.

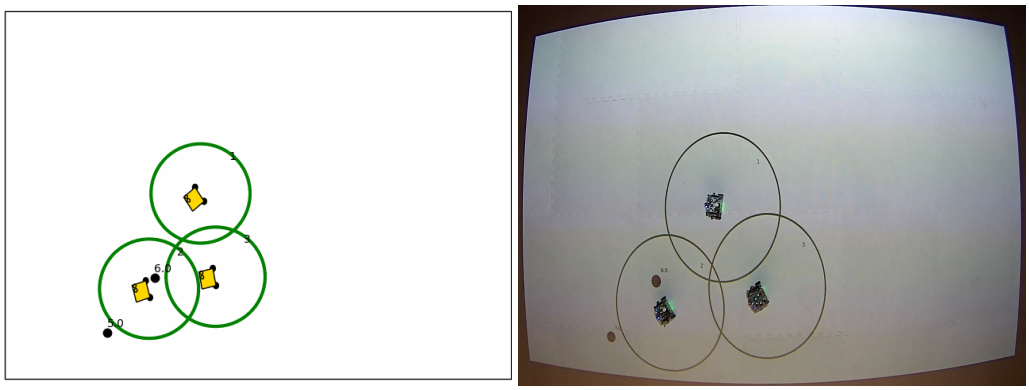

Figure 9: Foraging in simulation (left) and real-world (right).

B.3.2 Experiment Parameters

- Number of Robots: 3
- Number of Resources: 2
- Max steps: 100
- Training timesteps: 5e6
- Controller
  - Controller: unicycle position
  - Barrier Fn: enabled
- Heterogeneity
  - Type: capability set
  - Obs type: full capability set (robots observe the team's capability values)
  - Values: [[1], [2], [3]] (each entry is the robot's foraging level)
- Foraging Reward: $1 \times$ level of resources foraged
- Violation Reward: $0 \times$ number of collisions

B.4 Material Transport

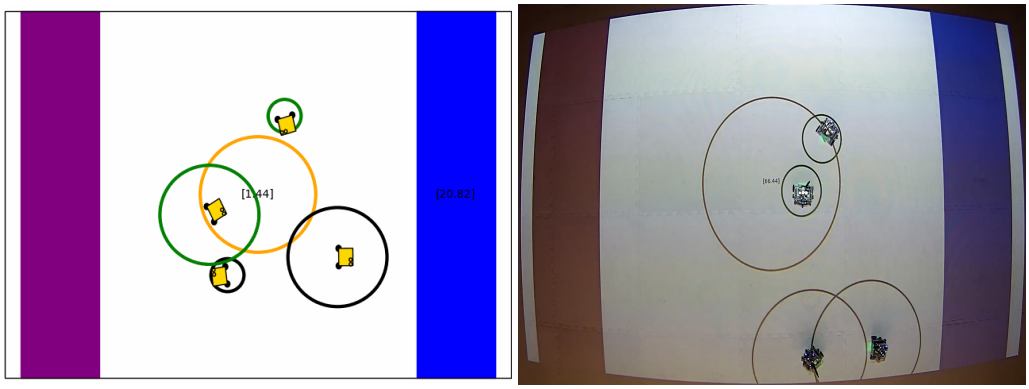

Figure 10: Material Transport in simulation (left) and real-world (right).

B.4.1 Description

Adopted from MARBLER, in this scenario, N robots with varying speeds and carrying capacities must collaborate to unload two loading zones (green circle and green rectangle) into the drop-off zone (purple rectangle). Both zones are initialized with an amount of material sampled from a

zone-specific distribution. We assume both distributions are Gaussian with pre-specified means and variances. Robots observe their position, the position of other robots, and the remaining material to unload in both zones. Robots can only load material by entering the loading zones, can only carry the amount of material specified to their carrying capacity, and must drop off their loaded material into the drop-off zone before loading more material. The team is rewarded for the amount of material loaded and the amount of material dropped off, and it is penalized for each step taken where there is material remaining in the loading zones. This scenario can scale to an arbitrary number of robots.

B.4.2 Experiment Parameters

- Number of Robots: 4
- Max steps: 70
- Training timesteps: 5e6
- Controller
  - Controller: unicycle position
  - Barrier Fn: enabled
- Heterogeneity
  - Type: capability set
  - Obs type: full capability set (robots observe the team's capability values)
  - Values: [[0.45, 5], [0.45, 5], [0.15, 15], [0.15, 15]] (each entry is [step size, carrying capacity])
- Gaussian distribution parameters of materials in circular loading zone
  - Mean: 75
  - Variance: 10
- Gaussian distribution parameters of materials in rectangular loading zone
  - Mean: 15
  - Variance: 4
- Load Reward: 0.25 $\times$ material loaded by robots
- Drop Off Reward: 0.75 $\times$ material dropped off by robots
- Step Reward: -0.1 $\times$ 0 if no material remaining in loading zones, otherwise 1
- Violation Reward: 0 $\times$ number of collisions

B.5 Navigation (MAPF)

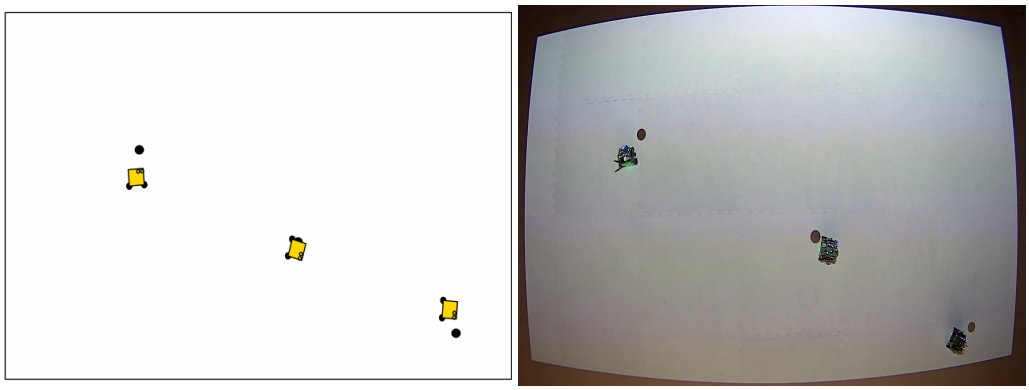

Figure 11: Navigation (MAPF) in simulation (left) and real-world (right).

B.5.1 Description

MAPF is a classic problem in multi-robot/multi-agent systems [45, 46]. In this scenario, N robots must navigate to individually specified goals (black markers). Robots observe their position and

the vector to their goal location. The team is rewarded for robots minimizing their distance to their respective goals. This scenario can scale to an arbitrary number of robots.

B.5.2 Experiment Parameters

- Number of Robots: 3
- Max steps: 100
- Training timesteps: 3e6
- Controller
  - Controller: unicycle position
  - Barrier Fn: enabled
- Distance Reward: -1 $\times$ euclidean distance of robots to their respective goals
- Violation Reward: 0 $\times$ number of collisions

B.6 Predator Prey

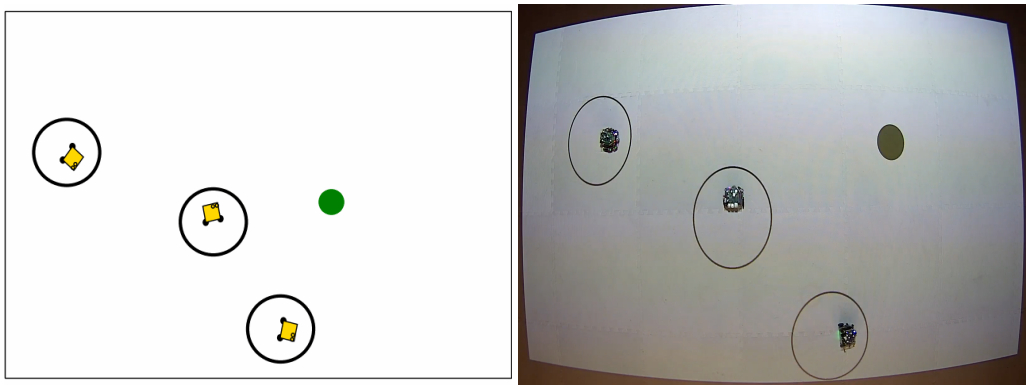

Figure 12: Predator Prey in simulation (left) and real-world (right).

B.6.1 Description

Based on a variant of the classic predator-prey scenario introduced in [47], in this scenario, N tagging robots must collaborate to tag a more agile prey (green marker). The prey is controlled by the heuristic proposed in [33], where at each step the prey maximizes its distance from the closest tagging robot. Robots observe their position, the position of other robots, and the position of the prey. Robots tag the prey by getting the prey within their tagging radius, and the prey marker will briefly turn red to indicate a successful tag. The team is rewarded for each successful tag on the prey. This scenario can scale to an arbitrary number of robots.

B.6.2 Experiment Parameters

- Number of Robots: 3
- Max steps: 100
- Training timesteps: 10e6
- Controller
  - Controller: unicycle position
  - Barrier Fn: enabled
- Tag Reward: 10 $\times$ 1 if a tag occurred, 0 otherwise
- Violation Reward: 0 $\times$ number of collisions

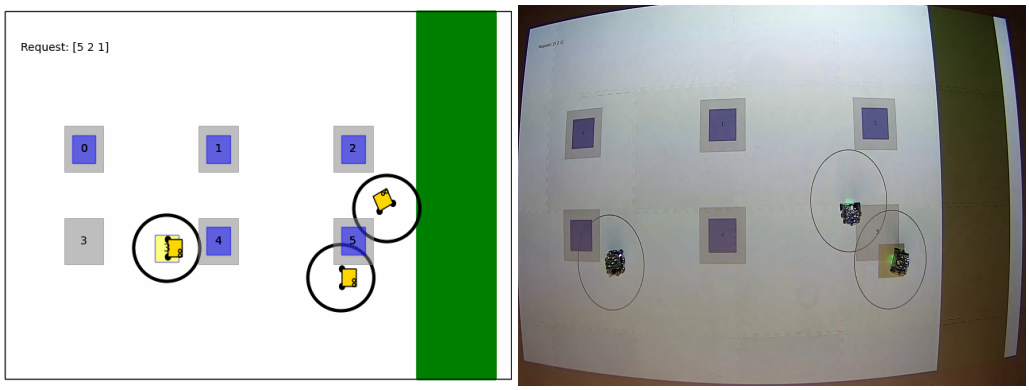

Figure 13: RWARE in simulation (left) and real-world (right).

B.7 Continuous-RWARE

B.7.1 Description

Inspired by grid-world RWARE [32], in this scenario, robots must collaborate to deliver the requested shelves (indicated by the top left text) to the dropoff zone (purple), where the request is updated upon each successful shelf delivery. Shelves are individually labeled and begin in staging zones (gray) arranged in a grid. Robots observe their position, the position of other robots, the position of each staging zone and the shelf occupying it, and the requested set of shelves. Robots can pickup or return shelves by entering the gray staging zone. If a robot is not carrying a shelf, it can freely move underneath shelves and through staging zones. Once a robot is carrying a shelf, it cannot move through other shelves, and shelves can only be returned at an unoccupied staging zone. A robot successfully completes a drop-off when it enters the purple zone carrying a shelf that is currently requested, at which point the corresponding entry in the request set is randomly updated to any non-requested shelf. Robots are sparsely rewarded for each successful drop-off. This scenario scales to an arbitrary number of robots and shelves.

B.7.2 Experiment Parameters

- Number of Robots: 3
- Number of Shelves: 6
- Max steps: 100
- Training timesteps: 20e6
- Controller
  - Controller: unicycle position
  - Barrier Fn: enabled
- Drop Off Reward: 1 $\times$ number of successful shelf drop-offs
- Violation Reward: 0 $\times$ number of collisions

B.8 Warehouse

B.8.1 Description

Adopted from MARBLER, in this scenario, N red robots and M green robots must navigate to the zone with matched color on the right to pick up packages, and then deliver them to the zone with corresponding colors on the left. Robots observe their position and the position of other robots. The team is rewarded for packages loaded and packages delivered. This scenario scales to an arbitrary number of robots.

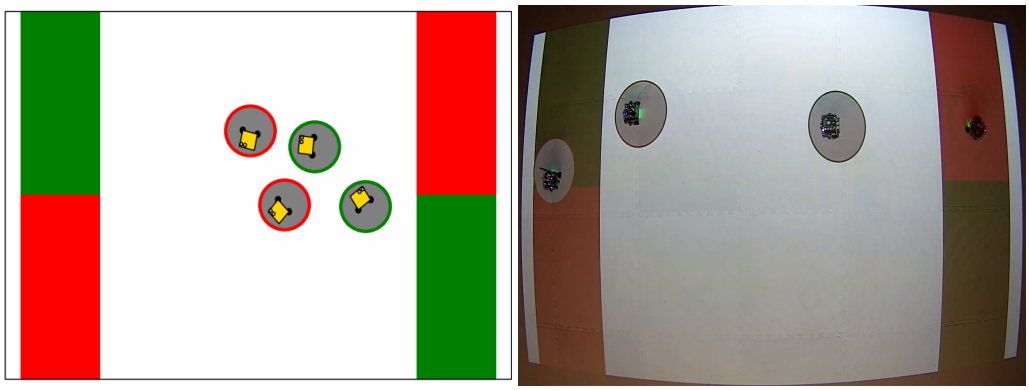

Figure 14: Warehouse in simulation (left) and real-world (right).

B.8.2 Experiment Parameters

- Number of Robots: 4
- Max steps: 70
- Training timesteps: 3e6
- Controller
  - Controller: unicycle position
  - Barrier Fn: enabled
- Heterogeneity
  - Type: class
  - Obs type: class (robots observe their own class)
  - Values: [[1, 0], [1, 0], [0, 1], [0, 1]] (each entry indicates the type [green, red])
- Load Reward: 1 $\times$ packages loaded
- Delivery Reward: 3 $\times$ packages delivered
- Violation Reward: 0 $\times$ number of collisions