# MATE: Benchmarking Multi-Agent Reinforcement Learning in Distributed Target Coverage Control

**Xuehai Pan**[1,2,5] `xuehaipan@pku.edu.cn`

**Mickel Liu**[1,2,5] `mickelliu@stu.pku.edu.cn`

**Fangwei Zhong**[3,5,†] `zfw@pku.edu.cn`

**Yaodong Yang**[3,4,5,†] `yaodong.yang@pku.edu.cn`

**Song-Chun Zhu**[3,4,5,6] `sczhu@bigai.ai`

**Yizhou Wang**[1,2,4] `yizhou.wang@pku.edu.cn`

[1] School of Computer Science, Peking University
[2] Center on Frontiers of Computing Studies, Peking University
[3] School of Intelligence Science and Technology, Peking University
[4] Institute for Artificial Intelligence, Peking University
[5] Beijing Institute for General Artificial Intelligence (BIGAI)
[6] Department of Automation, Tsinghua University

## Abstract

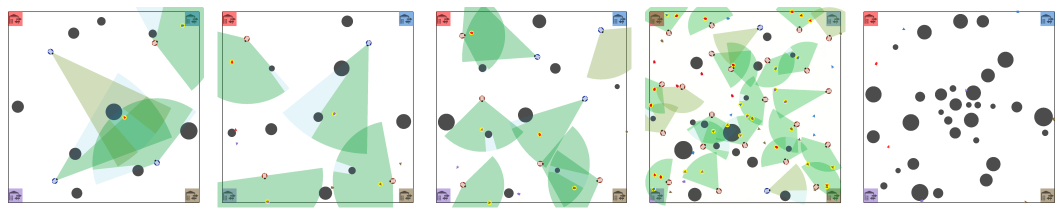
We introduce the Multi-Agent Tracking Environment (`MATE`), a novel multi-agent environment simulates the target coverage control problems in the real world. MATE hosts an asymmetric cooperative-competitive game consisting of two groups of learning agents—"cameras" and "targets"—with opposing interests. Specifically, "cameras", a group of directional sensors, are mandated to actively control the directional perception area to maximize the coverage rate of targets. On the other side, "targets" are mobile agents that aim to transport cargo between multiple randomly assigned warehouses while minimizing the exposure to the camera sensor networks. To showcase the practicality of MATE, we benchmark the multi-agent reinforcement learning (MARL) algorithms from different aspects, including cooperation, communication, scalability, robustness, and asymmetric self-play. We start by reporting results for cooperative tasks using MARL algorithms (MAPPO, IPPO, QMIX, MADDPG) and the results after augmenting with multi-agent communication protocols (TarMAC, I2C). We then evaluate the effectiveness of the popular self-play techniques (PSRO, fictitious self-play) in an asymmetric zero-sum competitive game. This process of co-evolution between cameras and targets helps to realize a less exploitable camera network. We also observe the emergence of different roles of the target agents while incorporating I2C into target-target communication. MATE is written purely in Python and integrated with OpenAI Gym API to enhance user-friendliness. Our project is released at `https://github.com/UnrealTracking/mate`.

## 1 Introduction

The target coverage problem studies the active control of the perception area of a group of agents to track the targets of interest, e.g., wireless sensor networks [1], surveillance camera networks [2, 3], and unmanned aerial vehicle (UAV) networks [4]. It has much real-life significance and received wide applications relating to social well-being, security, and entertainment. For example, smart camera networks can be used for anti-poaching [5, 6], anti-smuggling [7], border security [8] and—for more recreational uses—person-following [9, 10, 11] in filming and ball-tracking [12] in sports events, etc. However, it remains an open challenge to cooperatively control the cameras

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

[†]Equal Advising.



(a) 4C vs. 2T (90) (b) 4C vs. 8T (90) (c) 8C vs. 8T (90) (d) 24C vs. 48T (160) (e) Navigation

Figure 1: Five snapshots of The Multi-Agent Tracking Environment at different scales. Note that here we abbreviate "camera" as "C", "target" as "T" and "obstacle" as "O".

in distributed networks. A few notable issues hinder the progress: the quantities of the cameras and targets that vary in real-time, the increasingly diverse and unpredictable trajectories of targets, the partial observability of the cameras, and the limited bandwidth of the communication networks. These factors contribute to the difficulties in current research regarding multi-camera cooperation.

Recent successes in multi-agent reinforcement learning (MARL) [13] have demonstrated the superior efficiency of multi-agent learning methods in tackling cooperative-competitive games at super-human levels, as shown in gaming AI [14, 15, 16, 17], robotic manipulation [18, 19], autonomous driving [20, 21], population biology [22], and etc. Assuredly, the research on target coverage problems would benefit significantly from multi-agent reinforcement learning. Unfortunately, we notice that the popular MARL algorithms, e.g., MADDPG [23], QMIX [24], MAPPO [25], HAPPO [26, 27] perform poorly in the target coverage problem [28], though they achieved great success in other existing benchmarks [23, 29, 30]. These benchmarks are either based on video games or some simplified scenarios, neglecting which features the real-world multi-agent applications desperately demand, e.g., heterogeneous agents, asymmetric games, the variable population of agents, simulating partial observation, and peer-to-peer communication. We have yet to see an open-source and standardized environment that benchmarks the MARL algorithms under such practical settings in the context of the target coverage problem.

Motivated by these findings, we build the *Multi-Agent Tracking Environment (MATE)* that advocates the proposal of a more practical multi-agent system. Our system has accounted for both aspects of fully-cooperative and fully-competitive games, the scalability and the robustness of agents, and the communication efficiency among agents. MATE is an open-source simulation that hosts an asymmetric two-team stochastic game between the "cameras" and the "targets". Inside MATE, the camera agents need to maximize the coverage rate on the "targets" while maintaining strong coordination within the team to minimize overlapping detection. On the other hand, the targets are tasked to maximize transport flow between randomly assigned warehouses while minimizing the time of being detected by the cameras. The game-theoretic theme in MATE stimulates the emergence of innovative strategies between two teams of asymmetric agents, thus facilitating the process of autocurriculum. As a design consideration to encourage multi-layer strategies and counterplay, MATE adds randomly spawned obstacles and transport tasks to offset the natural advantages of both types of agents. For example, cameras' mobility is restricted; therefore, targets may hide behind obstacles to temporarily avoid exposure. Still, the freight and bounties would drive the targets to stop hiding as soon as possible and carry on with their assigned tasks.

Worth noting that MATE is not anchored to entirely service the target coverage problem for training a better camera network, but it also stands out as a multi-purpose benchmark to aid the advancement of MARL algorithms. As discussed in the related work section, MATE has many essential features demanded in algorithm-related research in MARL. Besides mixed-motive games, MATE supports fully cooperative and fully competitive game types, which found popularity in the current landscape of theoretical analysis on MARL algorithms [31, 32, 33, 34, 35]. MATE also provides Peer-to-Peer communication channels, a topic that recently gained much interest in developing means beyond information broadcast in multi-agent communication [36, 37]. The competitive game hosted in MATE involves two-team of heterogeneous agents. The target agents differ in carrying capacity and moving speed, marking further heterogeneity in learning the controls of target agents or learning counter strategies against various types of targets as the camera agents.

We showcased a series of experiments to confirm the feasibility of training for both teams inside MATE. We reported the performance of four MARL algorithms (MAPPO [25], IPPO [38], MADDPG [23], QMIX [24]) on different environment configurations. We also showed the performance of our agents in the cooperative game with multi-agent communication add-ons. We observed the emergence of different roles (e.g., "distractors" and "running backs") while training

Table 1: Comparison between relevant MAL environments (viewing with colors is recommended).

| Environment | Game Type | Observations | Actions | Communication | Agent Type | Scalable |
|---|---|---|---|---|---|---|
| MPE (2017) [23] | Mixed-Motive | Continuous | Continuous & Discrete | Broadcast | Heterogeneous | Yes |
| MAgent (2018) [55] | Mixed-Motive | Discrete | Discrete | No | Homogeneous | Yes |
| Pommerman (2018) [56] | Fully-Competitive | Continuous | Discrete | Broadcast | Homogeneous | No |
| MARLÖ (2018) [57] | Mixed-Motive | Continuous + Pixels | Discrete | No | Homogeneous | Yes |
| Hanabi (2019) [58] | Fully-Cooperative | Discrete | Discrete | Broadcast | Homogeneous | No |
| SMAC (2019) [29] | Fully-Cooperative | Continuous | Discrete | No | Heterogeneous | No |
| Neural MMO (2019) [59] | Mixed-Motive | Discrete | Multi-Discrete | No | Homogeneous | Yes |
| GFootball (2019) [60] | Fully-Cooperative | Continuous | Discrete | No | Homogeneous | No |
| MAMuJoCo (2020) [61] | Fully-Cooperative | Continuous | Continuous | No | Heterogeneous | No |
| LBF (2020) [62] | Mixed-Motive | Discrete | Discrete | No | Homogeneous | Yes |
| RWARE (2020) [62] | Mixed-Motive | Discrete | Discrete | Broadcast | Homogeneous | Yes |
| DM Lab2D (2020) [63] | Mixed-Motive | Discrete | Discrete | No | Homogeneous | No |
| Flatland (2020) [64] | Fully-Cooperative | Continuous | Discrete | No | Homogeneous | Yes |
| SMART (2020) [20] | Mixed-Motive | Continuous & Discrete | Continuous & Discrete | No | Heterogeneous | Yes |
| **MATE (Ours)** | Fully-Coop. & Fully-Comp. | Continuous | Continuous & Discrete | Peer-to-Peer | Heterogeneous | Yes |

target agents with communications. Lastly, we employed the adversarial training algorithms (Policy Space Response Oracle (PSRO) [39], fictitious self-play (FSP) [40], and self-play (SP)) in our environment. The results suggest that training with such methods will decrease the exploitability of the trained policies and thus improve their robustness. Regarding the development aspects of MATE, it is extensively integrated with the OpenAI Gym API [41], which enables excellent compatibility with popular RL libraries such as RLlib [42], Tianshou [43], Stable-Baselines-3 [44], and other Gym compatible frameworks. The lightweight of the MATE attributes to efficient computations on CPUs. This property facilitates the opportunities for mass-scale parallelisms using high-throughput architectures like Ape-X [45] and IMPALA [46], etc. Also catering to various research needs, the setting of the environment can also be easily configured, including action space, observability, reward structure, population, and scene layouts. The development team will provide detailed documentation on various features and commit long-term support to this project.

## 2 Related Work

**Target Coverage Problem.** The target coverage problem is to find an optimal control strategy for sensors such that the time to monitor every interested target can be as long as possible [47]. It is a long-standing problem in directional sensor networks [48, 49], robotics [50, 51, 52, 53], and computer vision [3, 54]. Most previous algorithms are heuristically designed for a specific setting or application, lacking a general solution for this problem. Recently, Xu *et al.* [28] built a 2D environment, formulated the problem as a multi-agent cooperative game, and introduced a hierarchical multi-agent reinforcement learning approach to solve this game. However, compared with the real-world scenarios, the environment is over-simplified due to random-walking targets and a lack of obstacles. In MATE, we aim to build a more realistic simulator for benchmarking the off-the-shelf learning algorithms, e.g., account for occlusion caused by obstacles, the limited observing area of sensors, and controllable Field-of-View (FoV) of the Pan-Tilt-Zoom (PTZ) cameras. Besides, we reformulate the problem as a cooperative-competitive game and provide an interface to control the targets, i.e., the targets are controlled by adversaries to relentlessly challenge the camera policy with new strategies for the purpose of improving the robustness and generalizability of the trackers.

**Multi-Agent Learning Environments.** Besides the context of MCMT, we have seen other multi-agent learning (MAL) environments that service different tasks. Table 1 summarizes the most relevant MAL environments based on our literature review. To our findings, MATE stands out as the environment that simultaneously offers fully-cooperative & fully-competitive game types, Peer-to-Peer communication support, and heterogeneous agents.

**Self-Play and Population-Based Training Regime.** MATE experimented with three training principles to promote camera-target competition in zero-sum games. Solving zero-sum games can be highly non-trivial due to the non-transitivity (e.g., Rock-Paper-Scissor) in the policy space [65]. Conventional self-play makes the agent continuously play against the latest copy of itself. Since the agents in MATE are heterogeneous, we adopt the asymmetric version [66, 67, 68] of the self-play training method. However, self-play may fail to converge due to the lack of policy diversity [69, 70], thereby trapped by the non-transitivity. Fictitious Self-Play (FSP) [40] is a population-based method that maintains a policy memory storing past versions of the policy and uniformly samples a policy from memory as the response against the opponent. Policy Space Response Oracle [39] with Nash

Equilibrium solver (PSRO-Nash) is also a population-based method that computes a meta-strategy distribution. Instead of a uniform distribution, the distribution computed by PSRO-Nash resembles that of a mixed-strategy Nash Equilibrium. Recently, many efforts have been spent on extending PSRO methods to diverse PSRO methods [71, 72], no-regret PSRO methods [70], and PSRO with meta-learning [73, 74]. In this paper, we conduct experiments to demonstrate the effectiveness of these training regimes for improving and evaluating the robustness of the tracking agents.

## 3 MATE: the Multi-Agent Tracking Environment

In this section, we will introduce various details about the MATE environment. There are 4 kinds of entities in this 2D mini-world (shown in Fig. 1): $N_{\mathcal{C}}$ proactive cameras $\mathcal{C} = \{c_i\}_{i=1}^{N_{\mathcal{C}}}$, $N_{\mathcal{T}}$ mobile targets $\mathcal{T} = \{t_i\}_{i=1}^{N_{\mathcal{T}}}$, $N_{\mathcal{O}}$ static obstacles, and $N_{\mathcal{W}}$ $(= 4)$ warehouses storing cargoes. The reward structure inside MATE resembles the "min-max" nature of a cooperative-competitive multi-agent game. Camera agents must maximize their coverage rate collaboratively while minimizing repeated detection on the same target. In the meantime, targets transport cargoes between warehouses as fast as possible while minimizing the surveillance from the cameras. The role of obstacles in the environment is to provide temporary shelter against camera surveillance, but at the same time can act as roadblocks on the path to the destination for the target agents. The warehouses are scattered at the four corners of the mini-world.

Plenty of cargo needed to be delivered by the targets between the warehouses, in which the cargoes are priced based on the delivery duration.

### 3.1 Entities and States

We define the state as the internal attributes of the entities, which may change continuously as the environment progresses. Every agent (or controllable entity) may obtain its own states (public + private) but can only observe the public states of other agents.

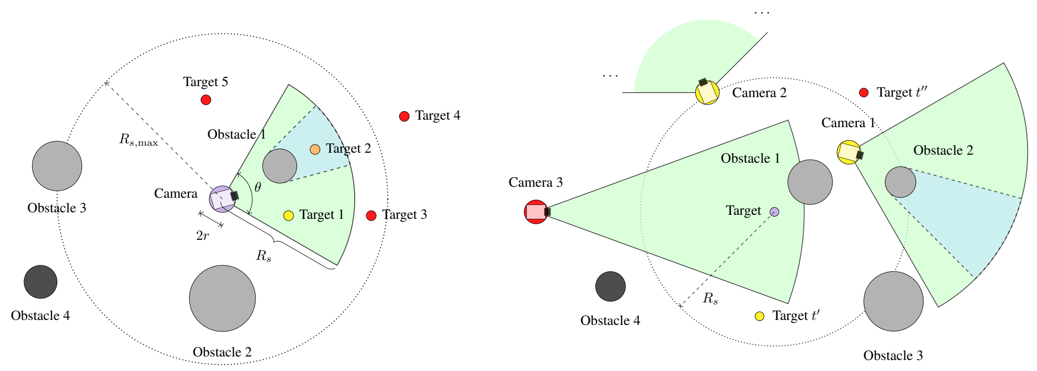
**Camera** is an in-place, zoomable, directional sensor with a pie-shape field of view. The *publicly accessible state* of the camera $\boldsymbol{s}_c^{\text{pub}} = [x, y, r, R_s, \phi, \theta]$ contains the self-location data in the world coordinate system, the physical radius $r$, the visible line of sight $R_s$, the viewing direction angle $\phi$, and field-of-view angle $\theta$. In addition to these, the camera's *privately accessible state* $\boldsymbol{s}_c^{\text{pvt}} = \left[\boldsymbol{s}_c^{\text{pub}}, R_{s,\max}, \Delta\phi_{\max}, \Delta\theta_{\max}\right]$ are constants indicating the maximum possible values for these parameters, with $\Delta\phi_{\max}$ being the camera's maximum rotation speed and $\Delta\theta_{\max}$ being the maximum zooming speed.

**Target** is a mobile vehicle equipped with an advanced omnidirectional sensor for which obstacles would not block the sensing field. The *publicly accessible state* $\boldsymbol{s}_t^{\text{pub}} = [x, y, R_s, \mathbb{I}[\text{loaded}]]$ consists of the self-location data, the sensible range $R_s$, and an indicating variable $\mathbb{I}[\text{loaded}]$ that indicates whether the target is loaded with payloads. The *privately accessible state* $\boldsymbol{s}_t^{\text{pvt}} = \left[\boldsymbol{s}_t^{\text{pub}}, v_{\max}, W^{(1)}, \dots, W^{(N_{\mathcal{W}})}, E^{(1)}, \dots, E^{(N_{\mathcal{W}})}\right]$ are the maximum movement speed $v_{\max}$, a one-hot-like vector $\boldsymbol{W}$ to indicate the payload destination, and a bit array $\boldsymbol{E}$ to "memorize" if the previously visited warehouse is empty. There are two kinds of vehicles for targets, one with high speed and small carrying capacity, and the other with low speed and large carrying capacity. The former is twice as fast as the latter but has a halved carrying capacity.

**The Obstacle** is a circular-shape static object that randomly spawns (controlled by a distribution) in the environment. Targets may use obstacles to stay hidden from camera surveillance. We added a transmittance attribute to the obstacles, so the cameras with a particular chance can detect the target hidden behind the obstacle at each timestep. This design feature prevents the target agents from over-reliance on this shortsighted strategy. The state of an obstacle includes the location and the radius, i.e., $\boldsymbol{s}_o = \boldsymbol{s}_o^{\text{pub}} = [x, y, r]$.

### 3.2 Observations

Observation acquired by an agent is a partial representation of the true states of the environment. By default, the agents in MATE have partial observability over the environment. An agent could only observe the entities within its field of view and obtain the publicly accessible states of these observable entities.



(a) Camera Observation (within the green sector) (b) Target Observation (within the dotted circle)

Figure 2: Schematic diagram of agent sight ranges. The agent (at the center, colored in purple) can obtain its own privately accessible state, and other agents' and obstacles' publicly accessible states within the sight range (colored in yellow and gray). The camera can perceive the target (colored in orange) behind an obstacle with the probability value of the obstacle's transmittance.

**Camera observations** The observation of a camera $c \in \mathcal{C}$ can be divided into five parts, the preserved data $\boldsymbol{s}^{\text{psrv}}$, its own privately accessible state $\boldsymbol{s}_c^{\text{pvt}}$, and the publicly accessible states of targets and obstacles and other cameras within its field of view. The preserved data $\boldsymbol{s}^{\text{psrv}}$ contains global information about the environment, such as the number of each category of entities, the current agent ID in the team, etc. Fig. 2a shows an example of the camera's perception in the green-shaded area.

**Target observations** The observation of the target $t \in \mathcal{T}$ is also composed of multiple parts, including the preserved data $\boldsymbol{s}^{\text{psrv}}$, its own privately accessible state $\boldsymbol{s}_t^{\text{pvt}}$; and the publicly accessible states of cameras, obstacles and other targets. Fig. 2b helps to demonstrate an example of the target's sight range.

Please refer to the supplementary for the mathematical formulation of the observations of both camera and target agents.

## 3.3 Actions

**Camera** is an in-place directional sensor with two types of continuous actions: rotation and zooming. These two action parameters can be adjusted simultaneously. Together they determine the shape of the camera's perception zone, though the area of perception remains unchanged. This design consideration balances the expected values of the number of perceivable targets under all possible action parameters.

**Target** is a mobile agent that can move freely inside this mini-world. Target's action space consists of the displacement vector $\boldsymbol{v} = (\Delta x, \Delta y)$ in Cartesian coordinates.

## 3.4 Reward Structure and Types of Stochastic Games

MATE can switch between three reward mechanisms corresponding to the typical three types of stochastic games. But first and foremost, we will motivate the use of Mean Coverage Rate as a reward function, which is an important metric used for evaluating MCMT tasks.

**Mean Coverage Rate** One typical method in RL for evaluating different models' performances on an equal basis is using the mean reward of every episode. However, we argue that mean coverage rate, while strongly correlating to episode mean reward, is comparatively a more intuitive measure for evaluating the performance of camera agents on tracking tasks. The mean coverage rate is also a normalized measure that eases the problem of comparing results from different environment configurations. For an episode with length $L$, it is:

$$\text{Mean Coverage Rate} = \frac{1}{L} \sum_{k=1}^{L} \frac{\text{Number of Detected Targets at step } k}{\text{Total Number of Targets}}. \tag{1}$$

In an intra-team *fully-cooperative game*, all agents in the same team will receive team rewards based on the team performance and would not differ between individuals. MATE, by default, uses Mean

Coverage Rate as the team reward for the camera agents. For the target agents, we propose the *transport reward* to incentivize the transport of goods with the awareness of avoiding detection. Formally, the target team reward is $r^{(\mathcal{T})} = F + B$ and empirically we keep $F$ and $B$ roughly equal. $F$ stands for "freight", a fixed-value sparse reward received upon every successful delivery of the assigned cargoes. $B$ is short for "bounty" on every cargo. The value of the bounty will depreciate per time-step and further decrease if the cameras have detected the target that carries this cargo. Whereas in the setting of inter-team *fully-competitive game*, we let the camera team receives the opposite value of the team reward of the targets, i.e., $r^{(\mathcal{C})} = -r^{(\mathcal{T})}$. The two teams are kept to play a zero-sum game in the environment. In addition to the built-in team rewards mentioned above, users can customize the reward functions with wrappers. Depending on the user configuration, the game setting may transform into a general *mixed-motive game*.

## 4 Core Features of MATE

**Sample Efficient and Easy to Use** MATE is a multi-agent environment based on numerical simulation and implemented in pure Python with minimal dependencies[1]. Users can install MATE with a single shell command. Without parallelization, a single-thread program can sample around 300 steps per second on a modern CPU[2] in the default configuration (4 cameras, 8 targets, 9 obstacles). Besides, MATE ships with various custom wrappers and built-in rule-based agents, and the existing algorithms can run on MATE with few modifications. The source code is released under the MIT Open Source License with detailed documentation. The MATE environment is out-of-the-box compatible with OpenAI Gym API [41]. We represent the sample code at the following that runs random action agents on our environment.

```python
import mate
env = mate.make('MultiAgentTracking-v0')  # or gym.make
env.reset()
done = False
while not done:
    camera_joint_action, target_joint_action = env.action_space.sample()
    (
        (camera_joint_observation, target_joint_observation),
        (camera_team_reward, target_team_reward),
        done,
        (camera_infos, target_infos)
    ) = env.step((camera_joint_action, target_joint_action))
```

**Communicative Agents** MATE implements an intra-team communication channel for each team that supports both broadcast and Peer-to-Peer communication. Unlike the widely used MPE environment [23], we explicitly isolate messages from agent observations so the user may customize the message format, such as vectors or texts. Communication facilitates strategic coordination among agents, preventing unnecessary exploration and repeated efforts, especially in a partially

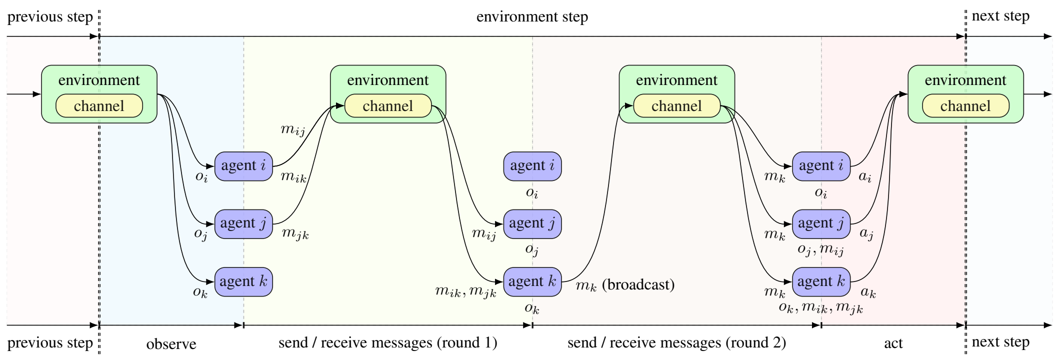


Figure 3: Illumination of the multi-round communication mechanism in MATE.

[1] Only NumPy, SciPy, Gym, and their dependencies are required.

[2] Tested using a single-threaded program with an Intel® i7 8700 @ 3.20GHz CPU. The agents take random action in the environment.

Table 2: Overview of the baseline algorithms. The TarMAC and I2C algorithms are communication add-ons that can utilize with other multi-agent reinforcement learning algorithms.

| Algorithm | Category | Centralized Training | On/Off-Policy | Action Space | Communication |
|---|---|---|---|---|---|
| QMIX [24] | Value-based | Yes | Off | Discrete | No |
| MADDPG [23] | Policy-based | Yes | Off | Continuous | No |
| IPPO [38] | Policy-based | No | On | Discrete / Continuous | No |
| MAPPO [25] | Policy-based | Yes | On | Discrete / Continuous | No |
| TarMAC [78] | Communication | Yes | On / Off | Discrete / Continuous | Broadcast |
| I2C [36] | Communication | Yes | On / Off | Discrete / Continuous | Peer-to-Peer |

observable environment. Users may add custom wrappers to simulate random signal noise, distance-based delays, restricted communication ranges, limited bandwidths, etc. As Fig. 3 illustrates, MATE supports multi-round communications, allowing agents to exchange several rounds of information within the same environment step. This mechanism can facilitate more multi-agent research regarding negotiations [75, 76] and conversations [77].

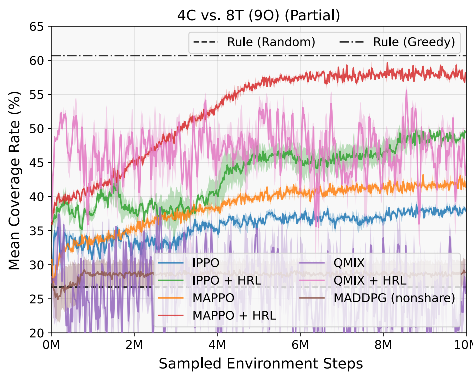
**Heterogeneous and Asymmetric** The heterogeneity of the MATE environment exhibits two aspects: inter-team and intra-team. First, the agents in two opposite teams are completely distinct regarding their dynamics and tasks. Second, agents within the same team may have different abilities, e.g., the vehicles for targets with varying carrying capacities and movement speeds. These heterogeneities reflect a realistic theme and will allow agents to emerge with diversified strategies and individuality in a complex environment. The MATE environment hosts an asymmetric competitive game with two teams of heterogeneous agents. Asymmetry is not only reflected in the heterogeneity of agents but also in the variable quantities of players in the teams. Under different game setups, the equilibrium of the game change relative to the strength of both teams, and their corresponding strategy should also adapt accordingly.

**Variety, Flexibility, and Scalability** In our default configuration, two groups of agents perform two tasks – the target coverage task for the camera agents and the transport task for the target agents. But with our user-friendly framework, researchers may extend this environment to suit more missions for the agents, such as deciding camera placement, trajectory prediction, resource collection, etc. The default setup of the environment reflects the most complex setting in which the following features are enabled: 1) *mixed cooperation-competition*, 2) *continuous action*, 3) *partially observable*, 4) *communicative*, and 5) *team reward only*. MATE is highly modularized so that the users can convert to different environments to suit their particular needs with our provided wrappers (presented in the supplementary). Users can train a target or camera network curriculum by dynamically adjusting the difficulty levels and transferring these policies across different settings. The number of entities in the environment is also configurable. MATE may support simultaneous interactions between two to hundreds of agents. As the quantity of agents varies, the complexity and difficulty of the environment also change accordingly, which allows the emergence of diverse strategies. Simply by varying the number of agents on both sides, the users can test the robustness of the newly-developed Multi-Agent Learning (MAL) algorithms.

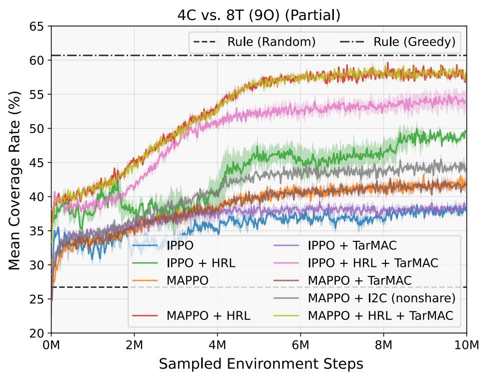
## 5 Experiments

In this section, we will present the results for **(1)** collaborative game where training one team of agents (either cameras or targets) against rule-based opponents **(2)** additionally incorporates multi-agent communications into the collaboration games **(3)** competitive game where training two teams of agents using asymmetric self-play or Population Based Training (PBT).

For fair comparisons in the cooperative games, we ran each experiment between different algorithms for 10 million environment steps. The model performance was averaged across experiments from three random seeds to reflect statistical properties. We use and extend the RLlib [42] to implement QMIX [24], MADDPG [23], IPPO [38], MAPPO [25], TarMAC [78], and I2C [36] algorithms in all of our experiments. Table 2 lists the properties of the baseline algorithms.



(a) MARL baselines



(b) MARL with multi-agent communication

Figure 4: The learning curves for the camera agents in the cooperative game. The intervals (shaded region) indicate one standard deviation over three separate runs. The target agents are (greedy) rule-based controlled.

## 5.1 Training Cameras in Fully-cooperative Game

We conduct feasibility checks on the train-ability of camera agents using MARL algorithms on MATE and accordingly report the training performance of the camera's policies on the `4C` vs. `8T` (`9O`).[3] setting. We presented a hierarchical RL (HRL) model with a selection-based low-level policy, which is controlled by a rule-based executor. The MARL algorithms only learn the high-level policy. Moreover, we added a 5-steps frame-skip to the high-level policy, in which the low-level policy would receive a signal from the high-level policy once in 5 timesteps.

The results in Fig. 4a demonstrate the superior performance of the HRL methods and the steady convergence of the PPO-based methods. Fig. 4b shows the effect of adding multi-agent communication modules to various MARL algorithms on the Mean Coverage Rate. Enabling communication for the MAPPO algorithm with the hierarchical agent structure hardly shows improvement in convergence. We believe it is because of the strong inductive bias of the HRL method, given that the policies for the low-level executors are based on pre-set rules (recall that only the manager policy is trained).

## 5.2 Training Targets in Fully-cooperative Game

In this section, we run multiple MARL algorithms to train the target agents competing against rule-based (greedy) cameras. As expressed in Fig. 5, target agents experience more conclusive performance gains by incorporating the multi-agent communication module compared to the camera agents shown in Fig. 4b. IPPO with the TarMAC protocol may achieve comparable performance to MAPPO+TarMAC, given that the critic model of the latter additionally has access to the global states. MAPPO+I2C attains the best convergence out of all methods implemented. Fig. 6 reflects the difficulty in normalized reward for two different settings. In `2C` vs. `4T` (`0O`), partial observability does not significantly hinder learning for MAPPO but causes an approximately 0.35 decrease in normalized episode reward for the IPPO. In `4C` vs. `8T` (`0O`), IPPO failed

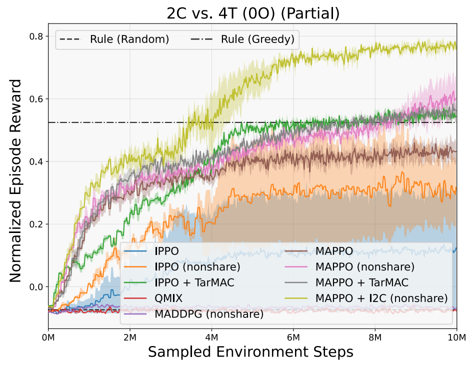


Figure 5: The learning curves (with intervals of one standard deviation across three separate runs) of various MARL methods in training target agents. Rule-based greedy agents control the cameras.

[3] Abbreviated "camera" as "`C`", "target" as "`T`" and "obstacle" as "`O`". Collectively `4C` vs. `8T` (`9O`) refers to a game setting where four camera agents play against eight target agents with nine obstacles in the environment

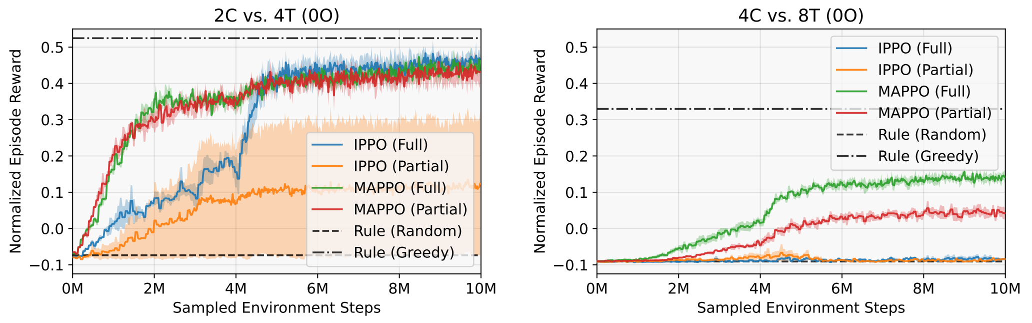


Figure 6: A comparison between two different settings shows the significance of the observability mode in learning meaningful policies for the target agents.

in both modes of observability while MAPPO suffered an approximately 0.1 decrease in the reward when switched to partial observability.

### 5.3 Zero-sum Fully-competitive Game

In Section 5.1 and 5.2, we present the results of training camera or target agents against fixed-policy opponents. However as these models implicitly treat their opponents as integrated parts of the non-stationary environment, this would often result in over-fitting or failure to generalize against new opponents [39]. In realistic deployment, a stable and robust solution is often preferred over a better-performing but brittle solution. Therefore to improve the robustness of the camera policy, we proposed to train camera and target agents in co-evolution with a zero-sum payoff structure. For the experiment, we trained the agents with three population-based methods: PSRO-Nash [39], Fictitious Self-Play [40], (asymmetric) self-play, and present the performance comparison in Fig. 7.

Note that the exploitability of a policy or a population of policies is an intuitive measure of robustness. The exploitability, in the context of MATE, is formally defined as:

$$\text{exploitablility}(\Pi^{\mathcal{C}}, \Pi^{\mathcal{T}}) = \frac{1}{2} \sum_{i \in \{\mathcal{C}, \mathcal{T}\}} \left[ U^i\big(\mathbf{BR}(\Pi^{-i}), \Pi^{-i}\big) - U^i\big(\Pi^i, \Pi^{-i}\big) \right], \tag{2}$$

where $\mathcal{C}, \mathcal{T}$ refer to "camera group" and "target group", $-i = \{\mathcal{C}, \mathcal{T}\} \setminus \{i\}$, $\mathbf{BR}$ stands for "best response", $U^i(\cdot, \cdot)$ $(i \in \{\mathcal{C}, \mathcal{T}\})$ are the utility functions, and $\Pi^i$ $(i \in \{\mathcal{C}, \mathcal{T}\})$ are the policy populations accordingly. For fully-competitive settings, $U^{\mathcal{C}} + U^{\mathcal{T}} = 0$.

The exploitability can be interpreted as the average performance difference between the best-response (BR) and current policies. Low exploitability implies that both opposing groups have approximately converged to their best-response policies at the current iteration, indicating proximity to the Nash equilibrium. In Fig. 7, both sub-figures show that PSRO-Nash and self-play can converge to policy populations that are less exploitable than the populations trained against non-

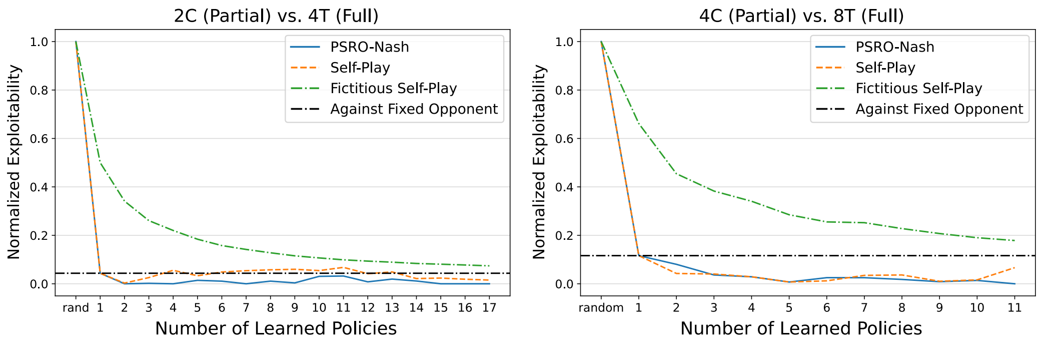


Figure 7: The exploitability of both the populations of cameras' and targets' policies at each population iteration. The left and right figures differ in the settings of the environments.

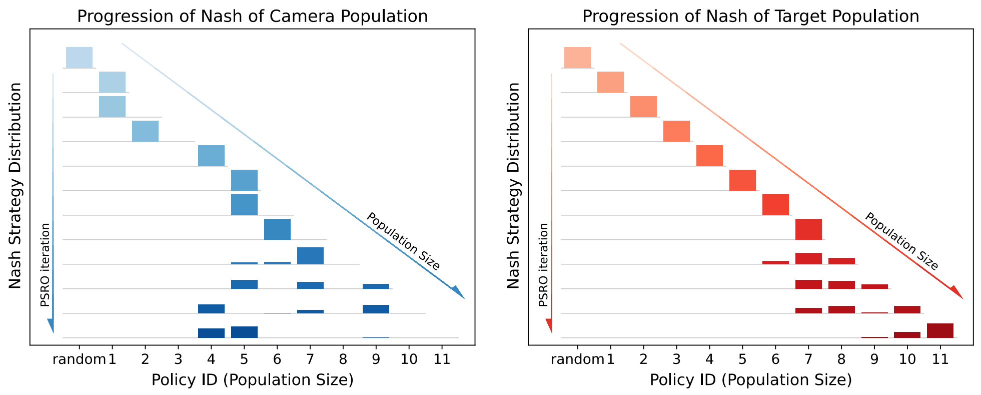


Figure 8: The probability distribution of the meta-strategy for each PSRO iteration for the policy population of both camera and target teams. Both teams adaptively change their policy according to the adversaries. The agents are trained under the `4C` vs. `8T` (`00`) configuration.

evolving, fixed-policy opponents. Since Fictitious Self-Play uniformly samples policy from the policy distribution, the random policy has an equal probability of being sampled as the other policies in the distribution at every population iteration, which therefore explains the slow decreasing trend in exploitability. Fig. 8 visualizes the progression of distribution for the meta-strategy taken by the players (the camera team and the target team). A relatively scattered Nash distribution indicates the presence of multiple plausible strategies.

## 6 Conclusion

Multi-Agent Tracking Environment (MATE) is a novel multi-agent simulation for benchmark multi-agent reinforcement learning algorithms in the target coverage problem. MATE hosts a zero-sum cooperative-competitive, asymmetric game between "cameras" and "targets", in which cameras gain team rewards to maximize the number of the covered targets. MATE incorporates the intra-team peer-to-peer (P2P) communication feature and trainable adversarial target agents. This environment is built purely in Python and integrated with OpenAI Gym API [41] enabling great compatibility and extensibility to most distributive RL frameworks. The lightweight of this environment also ensures high sampling efficiency. MATE allows for flexible configurations for the simulation environments along with highly customized scenarios to fulfill specific research needs. We conduct a series of benchmarks to show the performance of target and camera agents trained by MARL methods and algorithms in various settings. We hope this work will serve as a useful guide for the community using MATE to conduct further research.

**Limitations.** The first-stage focus of MATE is to provide an all-in-one benchmark for testing various MARL algorithms and a new platform for studying distributed target coverage tasks with trainable adversaries. Admittedly, the focus of MATE is lesser on the aspects of visual perceptions, for example, evaluations in three-dimensional space. For these purposes, one of our future works will extend MATE into the high-quality 3D game engine, e.g., developing realistic environments on Unreal Engine 4 (UE4) with UnrealCV [79].

**Fair Use of the Dataset.** Despite MATE was not purposefully designed for scenarios that incur direct violation of privacy or harm the well-being of others, the theme of MATE brings up a valuable topic of discussion with regard to a recognized conflict between the advancement of AI technologies and the integrity of social well-being. Although the possibility of direct application of MATE to other exploitative systems remains slim, the transfer of knowledge between multi-agent tracking systems is theoretically plausible. While enjoying the benefits of training smarter camera systems using MATE, we do have genuine concerns about the negative societal impacts due to the misuse of tracking technologies in repressive surveillance. We hereby advocate for more responsible use of multi-agent tracking environments including MATE and we condemn the act of using MATE or other similar systems for malicious activities.

## 7 Acknowledgements

This project was supported by MOST-2018AAA0102004, NSFC-62061136001, China National Post-doctoral Program for Innovative Talents (Grant No. BX2021008), and Qualcomm University Research Grant. We also thank Jing Xu, Yurong Chen, and Yuanfei Wang for their insightful discussions.

## References

[1] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, 2008.

[2] Hamid Aghajan and Andrea Cavallaro. *Multi-camera networks: principles and applications*. Academic press, 2009.

[3] Jing Li, Jing Xu, Fangwei Zhong, Xiangyu Kong, Yu Qiao, and Yizhou Wang. Pose-assisted multi-camera collaboration for active object tracking. *AAAI Conference on Artificial Intelligence*, 34(01):759–766, Apr. 2020.

[4] Samira Hayat, Evşen Yanmaz, and Raheeb Muzaffar. Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint. *IEEE Communications Surveys & Tutorials*, 18(4):2624–2661, 2016.

[5] Spynel-X Infrared Search & Track System - Wide area surveillance. https://hgh-infrared.com/spynel-x.

[6] Andries M. Heyns. Optimisation of surveillance camera site locations and viewing angles using a novel multi-attribute, multi-objective genetic algorithm: A day/night anti-poaching application. *Computers, Environment and Urban Systems*, 88:101638, 2021.

[7] Mark M. Pitt. Smuggling and price disparity. *Journal of International Economics*, 11(4):447–458, 1981.

[8] U S Patrol. Privacy impact assessment update for the border surveillance systems (BSS). *DHS/CBP, PIA-022 (A)*, 2018.

[9] Matthias Müller and Vladlen Koltun. Openbot: Turning smartphones into robots. In *International Conference on Robotics and Automation (ICRA)*, 2021.

[10] Wenhan Luo, Peng Sun, Fangwei Zhong, Wei Liu, Tong Zhang, and Yizhou Wang. End-to-end active object tracking via reinforcement learning. In *International Conference on Machine Learning*, pages 3286–3295, 2018.

[11] Wenhan Luo, Peng Sun, Fangwei Zhong, Wei Liu, Tong Zhang, and Yizhou Wang. End-to-end active object tracking and its real-world deployment via reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(6):1317–1332, 2019.

[12] Ball tracking - A new era of tactical game analysis. https://kinexon.com/technology/ball-tracking.

[13] Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*, 2020.

[14] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*, 2020.

[15] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[16] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[17] P Peng, Q Yuan, Y Wen, Y Yang, Z Tang, H Long, and J Wang. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

[18] Yuanpei Chen, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen Marcus McAleer, Hao Dong, Song-Chun Zhu, and Yaodong Yang. Towards human-level bimanual dexterous manipulation with reinforcement learning. In *Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[19] Yiran Geng, Boshi An, Haoran Geng, Yuanpei Chen, Yaodong Yang, and Hao Dong. End-to-end affordance learning for robotic manipulation. *arXiv preprint arXiv:2209.12941*, 2022.

[20] Ming Zhou, Jun Luo, Julian Villella, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, Aurora Chongxi Huang, Ying Wen, Kimia Hassanzadeh, Daniel Graves, Dong Chen, Zhengbang Zhu, Nhat Nguyen, Mohamed Elsayed, Kun Shao, Sanjeevan Ahilan, Baokuan Zhang, Jiannan Wu, Zhengang Fu, Kasra Rezaee, Peyman Yadmellat, Mohsen Rohani, Nicolas Perez Nieves, Yihan Ni, Seyedershad Banijamali, Alexander Cowen Rivers, Zheng Tian, Daniel Palenicek, Haitham bou Ammar, Hongbo Zhang, Wulong Liu, Jianye Hao, and Jun Wang. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. In *Conference on Robot Learning (CoRL)*, 11 2020.

[21] Minne Li, Zhiwei Qin, Yan Jiao, Yaodong Yang, Jun Wang, Chenxi Wang, Guobin Wu, and Jieping Ye. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *The World Wide Web Conference*, pages 983–994, 2019.

[22] Yaodong Yang, Lantao Yu, Yiwei Bai, Jun Wang, Weinan Zhang, Ying Wen, and Yong Yu. A study of ai population dynamics with million-agent reinforcement learning. *arXiv preprint arXiv:1709.04511*, 2017.

[23] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[24] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4295–4304. PMLR, 10–15 Jul 2018.

[25] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[26] Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2021.

[27] Jakub Grudzien Kuba, Xidong Feng, Shiyao Ding, Hao Dong, Jun Wang, and Yaodong Yang. Heterogeneous-agent mirror learning: A continuum of solutions to cooperative marl. *arXiv preprint arXiv:2208.01682*, 2022.

[28] Jing Xu, Fangwei Zhong, and Yizhou Wang. Learning multi-agent coordination for enhancing target coverage in directional sensor networks. *Advances in Neural Information Processing Systems*, 33:10053–10064, 2020.

[29] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[30] Ming Zhou, Ziyu Wan, Hanjing Wang, Muning Wen, Runzhe Wu, Ying Wen, Yaodong Yang, Weinan Zhang, and Jun Wang. Malib: A parallel framework for population-based multi-agent reinforcement learning. *arXiv preprint arXiv:2106.07551*, 2021.

[31] Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2022.

[32] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. *Policy Optimization Provably Converges to Nash Equilibria in Zero-Sum Linear Quadratic Games*. Red Hook, NY, USA, 2019.

[33] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5887–5896. PMLR, 09–15 Jun 2019.

[34] Julien Perolat, Bilal Piot, and Olivier Pietquin. Actor-critic fictitious play in simultaneous move multistage games. In *International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 919–928. PMLR, 09–11 Apr 2018.

[35] Xiaotie Deng, Yuhao Li, David Henry Mguni, Jun Wang, and Yaodong Yang. On the complexity of computing markov perfect equilibrium in general-sum stochastic games. *arXiv preprint arXiv:2109.01795*, 2021.

[36] Ziluo Ding, Tiejun Huang, and Zongqing Lu. Learning individually inferred communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, volume 33, pages 22069–22079, 2020.

[37] Yuanfei Wang, Fangwei Zhong, Jing Xu, and Yizhou Wang. Tom2c: Target-oriented multi-agent communication and cooperation with theory of mind. In *International Conference on Learning Representations*, 2022.

[38] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

[39] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[40] Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In Francis R. Bach and David M. Blei, editors, *International Conference on Machine Learning*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 805–813. JMLR.org, 2015.

[41] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[42] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, volume 80, pages 3053–3062. PMLR, 10–15 Jul 2018.

[43] Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Yi Su, Hang Su, and Jun Zhu. Tianshou: A highly modularized deep reinforcement learning library. *Journal of Machine Learning Research*, 23(267):1–6, 2022.

[44] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[45] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. In *International Conference on Learning Representations*, 2018.

[46] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1407–1416. PMLR, 10–15 Jul 2018.

[47] S Mini, Siba K Udgata, and Samrat L Sabat. Sensor deployment and scheduling for target coverage problem in wireless sensor networks. *IEEE Sensors Journal*, 14(3):636–644, 2013.

[48] Huadong Ma and Yonghe Liu. Some problems of directional sensor networks. *International Journal of Sensor Networks*, 2(1-2):44–52, 2007.

[49] M Amac Guvensan and A Gokhan Yavuz. On coverage issues in directional sensor networks: A survey. *Ad Hoc Networks*, 9(7):1238–1255, 2011.

[50] Dimitrios Zorbas, Luigi Di Puglia Pugliese, Tahiry Razafindralambo, and Francesca Guerriero. Optimal drone placement and cost-efficient target coverage. *Journal of Network and Computer Applications*, 75:16–31, 2016.

[51] Ahmed Saeed, Ahmed Abdelkader, Mouhyemen Khan, Azin Neishaboori, Khaled A Harras, and Amr Mohamed. Argus: realistic target coverage by drones. In *ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 155–166, 2017.

[52] Mouhyemen Khan, Karel Heurtefeux, Amr Mohamed, Khaled A Harras, and Mohammad Mehedi Hassan. Mobile target coverage and tracking on drone-be-gone uav cyber-physical testbed. *IEEE Systems Journal*, 12(4):3485–3496, 2017.

[53] Eva Tuba, Romana Capor-Hrosik, Adis Alihodzic, and Milan Tuba. Drone placement for optimal coverage by brain storm optimization algorithm. In *International Conference on Hybrid Intelligent Systems*, pages 167–176. Springer, 2017.

[54] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision*, pages 17–35. Springer, 2016.

[55] Lianmin Zheng, Jiacheng Yang, Han Cai, Ming Zhou, Weinan Zhang, Jun Wang, and Yong Yu. Magent: A many-agent reinforcement learning platform for artificial collective intelligence. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[56] Cinjon Resnick, Wes Eldridge, David Ha, Denny Britz, Jakob Foerster, Julian Togelius, Kyunghyun Cho, and Joan Bruna. Pommerman: A multi-agent playground. *arXiv preprint arXiv:1809.07124*, 2018.

[57] Diego Perez-Liebana, Katja Hofmann, Sharada Prasanna Mohanty, Noburu Kuno, Andre Kramer, Sam Devlin, Raluca D. Gaina, and Daniel Ionita. The multi-agent reinforcement learning in malmÖ (marlÖ) competition. *arXiv preprint arXiv:1901.08129*, 2019.

[58] Nolan Bard, Jakob N. Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H. Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, Hugo Larochelle, Marc G. Bellemare, and Michael Bowling. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280, 2020.

[59] Joseph Suarez, Yilun Du, Clare Zhu, Igor Mordatch, and Phillip Isola. The neural mmo platform for massively multiagent research. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021.

[60] Karol Kurach, Anton Raichuk, Piotr Stanczyk, Michal Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. Google research football: A novel reinforcement learning environment. In *AAAI Conference on Artificial Intelligence*, 2020.

[61] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Boehmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12208–12221, 2021.

[62] Filippos Christianos, Lukas Schäfer, and Stefano Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 10707–10717, 2020.

[63] Charles Beattie, Thomas Köppe, Edgar A. Duéñez-Guzmán, and Joel Z. Leibo. Deepmind lab2d. *arXiv preprint arXiv:2011.07027*, 2020.

[64] Sharada Mohanty, Erik Nygren, Florian Laurent, Manuel Schneider, Christian Scheller, Nilabha Bhattacharya, Jeremy Watson, Adrian Egli, Christian Eichenberger, Christian Baumberger, Gereon Vienken, Irene Sturm, Guillaume Sartoretti, and Giacomo Spigler. Flatland-rl : Multi-agent reinforcement learning on trains. *arXiv preprint arXiv:2012.05893*, 2020.

[65] Ricky Sanjaya, Jun Wang, and Yaodong Yang. Measuring the non-transitivity in chess. *Algorithms*, 15(5):152, 2022.

[66] Fangwei Zhong, Peng Sun, Wenhan Luo, Tingyun Yan, and Yizhou Wang. AD-VAT: An asymmetric dueling mechanism for learning visual active tracking. In *International Conference on Learning Representations*, 2019.

[67] Fangwei Zhong, Peng Sun, Wenhan Luo, Tingyun Yan, and Yizhou Wang. Ad-vat+: An asymmetric dueling mechanism for learning and understanding visual active tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1467–1482, 2021.

[68] Fangwei Zhong, Peng Sun, Wenhan Luo, Tingyun Yan, and Yizhou Wang. Towards distraction-robust active visual tracking. In *International Conference on Machine Learning*, pages 12782–12792. PMLR, 2021.

[69] Yaodong Yang, Jun Luo, Ying Wen, Oliver Slumbers, Daniel Graves, Haitham Bou Ammar, Jun Wang, and Matthew E Taylor. Diverse auto-curriculum is critical for successful real-world multiagent learning systems. In *International Conference on Autonomous Agents and MultiAgent Systems*, pages 51–56, 2021.

[70] Le Cong Dinh, Stephen Marcus McAleer, Zheng Tian, Nicolas Perez-Nieves, Oliver Slumbers, David Henry Mguni, Jun Wang, Haitham Bou Ammar, and Yaodong Yang. Online double oracle. *Transactions on Machine Learning Research*, 2022.

[71] Nicolas Perez-Nieves, Yaodong Yang, Oliver Slumbers, David H Mguni, Ying Wen, and Jun Wang. Modelling behavioural diversity for learning in open-ended games. In *International Conference on Machine Learning*, pages 8514–8524. PMLR, 2021.

[72] Xiangyu Liu, Hangtian Jia, Ying Wen, Yujing Hu, Yingfeng Chen, Changjie Fan, Zhipeng Hu, and Yaodong Yang. Towards unifying behavioral and response diversity for open-ended learning in zero-sum games. *Advances in Neural Information Processing Systems*, 34:941–952, 2021.

[73] Xidong Feng, Oliver Slumbers, Ziyu Wan, Bo Liu, Stephen McAleer, Ying Wen, Jun Wang, and Yaodong Yang. Neural auto-curricula in two-player zero-sum games. *Advances in Neural Information Processing Systems*, 34, 2021.

[74] Bo Liu, Xidong Feng, Haifeng Zhang, Jun Wang, and Yaodong Yang. Settling the bias and variance of meta-gradient estimation for meta-reinforcement learning. *arXiv preprint arXiv:2112.15400*, 2021.

[75] Kris Cao, Angeliki Lazaridou, Marc Lanctot, Joel Z Leibo, Karl Tuyls, and Stephen Clark. Emergent communication through negotiation. In *International Conference on Learning Representations*, 2018.

[76] Yoram Bachrach, Richard Everett, Edward Hughes, Angeliki Lazaridou, Joel Z Leibo, Marc Lanctot, Michael Johanson, Wojciech M Czarnecki, and Thore Graepel. Negotiating team formation using deep reinforcement learning. *Artificial Intelligence*, 288:103356, November 2020.

[77] Diane Bouchacourt and Marco Baroni. Miss tools and mr fruit: Emergent communication in agents learning about object affordances. *arXiv preprint arXiv:1905.11871*, 2019.

[78] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, pages 1538–1546. PMLR, 2019.

[79] Weichao Qiu, Fangwei Zhong, Yi Zhang, Siyuan Qiao, Zihao Xiao, Tae Soo Kim, Yizhou Wang, and Alan Yuille. Unrealcv: Virtual worlds for computer vision. In *ACM International Conference on Multimedia*, pages 1221–1224, 2017.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We introduce a novel environment as a new benchmark that is applicable to MSMT tasks and test multi-agent learning algorithms
   (b) Did you describe the limitations of your work? [Yes] See Section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]