Asymmetric Self-Play-Enabled Intelligent Heterogeneous Multirobot Catching System Using Deep Multiagent Reinforcement Learning

Yuan Gao, *Member, IEEE*, Junfeng Chen, Xi Chen, Chongyang Wang, Junjie Hu, *Member, IEEE*, Fuqin Deng, *Member, IEEE*, and Tin Lun Lam, *Senior Member, IEEE*

***Abstract*—Aiming to develop a more robust and intelligent heterogeneous system for adversarial catching in security and rescue tasks, in this article, we discuss the specialities of applying asymmetric self-play and curriculum learning techniques to deal with the increasing heterogeneity and number of different robots in modern heterogeneous multirobot systems (HMRS). Our method, based on actor-critic multiagent reinforcement learning, provides a framework that can enable cooperative behaviors among heterogeneous multirobot teams. This leads to the development of an HMRS for complex catching scenarios that involve several robot teams and real-world constraints. We conduct simulated experiments to evaluate different mechanisms' influence on our method's performance, and real-world experiments to assess our system's performance in complex real-world catching problems. In addition, a bridging study is conducted to compare our method with a state-of-the-art method called S2M2 in heterogeneous catching problems, and our method performs better in adversarial settings. As a result, we show that the proposed framework, through fusing asymmetric self-play and curriculum learning during training, is able to successfully complete the HMRS catching task under realistic constraints in both simulation and the real world, thus providing a direction for future large-scale intelligent security & rescue HMRS.**

***Index Terms*—Asymmetric self-play, catching systems, heterogeneous multirobot system (HMRS), reinforcement learning (RL).**

Manuscript received 28 November 2022; accepted 9 February 2023. Date of publication 12 April 2023; date of current version 8 August 2023. This work was supported in part by the National Key R&D Program of China under Grant 2020YFB1313300, in part by the China Postdoctoral Foundation under Grant 2022M721838, and in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2022A1515110787. This paper was recommended for publication by Associate Editor J. Kelly and Editor W. Burgard upon evaluation of the reviewers' comments. *(Corresponding author: Tin Lun Lam.)*

Yuan Gao, Junfeng Chen, and Junjie Hu are with the Shenzhen Institute of Artificial Intelligence and Robotics for Society, The Chinese University of Hong Kong, Shenzhen 518100, China (e-mail: gaoyuankidult@gmail.com; chenjunfeng@stu.pku.edu.cn; hujunjie@cuhk.edu.cn).

Xi Chen is with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: xi8@kth.se).

Chongyang Wang is with the University College London, WC1E 6BT London, U.K. (e-mail: wangchongyang@tsinghua.edu.cn).

Fuqin Deng is with the Shenzhen Institute of Artificial Intelligence and Robotics for Society, The Chinese University of Hong Kong, Shenzhen 518100, China, and also with the School of Intelligent Manufacturing, Wuyi University, Jiangmen 354300, China (e-mail: dengfuqin@cuhk.edu.cn).

Tin Lun Lam is with the Shenzhen Institute of Artificial Intelligence and Robotics for Society, The Chinese University of Hong Kong, Shenzhen 518100, China, and also with the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen 518100, China (e-mail: tllam@cuhk.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2023.3257541.

Digital Object Identifier 10.1109/TRO.2023.3257541

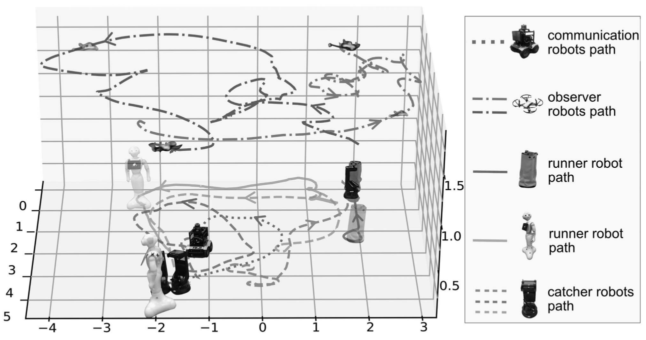


Fig. 1. Illustration of paths generated by different kinds of robots using our method in the catching scenario. In this scenario, the catching team consists of three subteams of three types of robots, namely, catcher robots, communication robots, and observer robots. The catching team needs to act cooperatively to catch a group of runner robots, surrounding them by intelligently using obstacles and walls while taking other constraints (e.g., communication, observational limit) into consideration.

I. Introduction

VERY few things in our environment are certain. Apart from inevitable events, learning and adapting is essential if we want to make any decisions. The same is also unfortunately true if we would like to make robotic systems. The dynamic and unpredictable nature of the world we live in makes it challenging to have a single type of robot to fit diverse situations. Consequently, we humans have developed robots like quadruped robots or quad-copter robots and assembled teams using different heterogeneous robots for complicated tasks. However, while the number and heterogeneity of robots are growing, the scientific and engineering challenges of enabling these heterogeneous robots to cooperate for the greater good of humanity need more exploration. As pointed out by Amanda Prorok [2] and other colleagues in the community [3], communication-enabled heterogeneous multirobot systems (HMRS) have been viewed as holding versatility in that each robot in the system can either act individually or cooperatively for different tasks. Although this characteristic dramatically extends the feasibility of the systems to adapt to a different environment for various tasks, the search space of automatic planning to complete these tasks is large. This allows us to consider improvements that take more uncertainty, which is generated by the increasing number and heterogeneity of robot systems, into consideration. A modern catching system made of heterogeneous robots shares the versatility of the HMRS but it also poses new challenges to the scheduling and planning algorithms.

Motivated by enabling an intelligent and robust HMRS, this work explains reasons why an HMRS that is subject to realistic constraints needs to leverage deep reinforcement learning (DRL) and other related techniques to solve the adversarial catching problem, which will further lead to establishing a real-world heterogeneous system to complete real-world catching scenarios. Fig. 1 Historically, the task planning and execution problem is considered hierarchical [4], and this is mainly because heterogeneous robots first need to identify their "skills" before communicating and cooperating to complete complicated tasks. Searching through an exponentially increasing large space is computationally intractable at scale. On the top layer, when multiple robots' skills are generated, a sequential decision-making process is then required for each robot to interact with the environment and other robots to complete different tasks. To make robots perform sequential decision-making is not a trivial problem due to the necessity of intelligently evaluating different situations. It requires frameworks spanning from optimization theory, dynamic programming, game theory, and decentralized control. As a complementary addition to traditional methods, using DRL-based approaches could be one of the prominent solutions to take more uncertainties into consideration and largely reduce humans' effort to achieve better performance. This is enabled by making robots learn through providing reward signals, and having the deep multiagent RL algorithms learn/optimize a strategy through trial-and-error experiments. Though one could argue that designing the reward signals requires some effort from humans, compared with modeling the complicated system dynamics, coming up with intuitive rewards is easier.

There were abundant successful studies of solving planning and control using traditional methods, i.e., optimization and programming algorithms, but when we are facing an increasing heterogeneity and number of agents, the challenge in HMRS still remains. It is mainly because of two reasons. First, DRL-based algorithms can perform well in planning tasks, especially in game settings [13], [14]. In recent studies, this still limits the task that the system can perform. For example, in studies aimed at generating HMRS' skills (e.g., trajectories, paths, a sequence of robot's actions, etc.), automated planning algorithms [7] and optimization algorithms [15] are still used in the decision-making process. The study of [7] implemented a mission-planning procedure for collaborative scientific sampling task on a moon-analogue site. The other work [15] used a scheduling and coordination mechanism to construct a wall with a predefined structure. Both works are excellent in their area, but the planning and decision-making process is only limited to straightforward tasks and involves less uncertainty. Second, the modeling process in traditional methods typically takes much more time than the learning-based methods. There are at least two factors involved. The first is that, instead of constructing a mathematical model based on the ideal assumptions, the learning-based methods can automatically learn with generalization to deal with more dynamic situations. The second is that the traditional methods lack the ability to make abstractions in increasingly complicated task environments. This, in machine learning terms, is called lack of the ability of representation learning [16]. Today, when most of the perceptual methods are deep learning-based, the traditional planning and control methods need a complementary method that can interpret similar structures, allowing rich input features to be taken into consideration. It is more apparent when it comes to HMRS' decision-making. For example, artificial intelligence (AI) systems constructed using traditional methods across multiple disciplines are proven not to be able to meet human-level intelligence [17]. The catching problem we consider has been studied relatively little before, although similar problems have been studied in tracking problems. In the following text, we will also include the tracking problem as part of the literature review for a fair comparison. Table I summarizes classic and recent state-of-the-art catching/tracking systems and the methods used. In order to achieve a higher level of intelligence and to deal with a larger number of agents in modern HMRS, we take the liberty of using a deep learning-based approach for our tracking/catching system. Though people have thought of using the RL-based method for the decision-making processes of HMRS, it was mainly on a conceptual level or just for homogeneous multirobot systems. For example, in this survey [13], the authors mentioned that different Markov decision process (MDP) frameworks can be used for the decision-making process and have the potential to be used in the HMRS. However, using multiagent RL in real world on HMRS still has problems to solve. This is on account of two aspects. The first aspect is that the presence of the heterogeneity in the system increases the training difficulty due to the increased complexity in cross-team and intrateam collaborative patterns. The second aspect is that the uncertainty in the real world causes the application of real-world RL to be hard, meaning balancing the training process to make it flexible enough in the real world is not easy. These problems together make it difficult to see multiagent RL algorithms applied in real-world HMRS. In practice, for real-world HMRS, only a few branches of methods are used. For example, in one branch, the traditional methods, like search-and-retrieval behavioral components [5] or sensor-feedback algorithms [18], are based on human-designed state machines. This branch lacks some level of flexibility. The other branches include swarm intelligence [19] and partially observable stochastic games. These methods can meet the requirements of scalability but not cooperative intelligence. One notable implementation of cooperative behavior is the blockchain-based cooperation strategy [20] but it has not demonstrated the method's ability to handle heterogeneity. According to [13], on different measures, the decentralized partially observable Markov decision process (Dec-POMDP) modeled HMRS can complement the traditional methods (e.g., swarm intelligence, graph-theoretic models) to achieve a high level of heterogeneity. Increasing the scalability while maintaining the heterogeneity of the multirobot system needs different DRL techniques to be utilized. To enable a more complex heterogeneous system in the future, in this work, we would like to answer two key questions related to the development of complicated HMRS.

TABLE I
CLASSIC AND RECENT STATE-OF-THE-ART HMRS FOR CATCHING-LIKE TASKS

Name	Experiments with Real-world Robots	Intelligent Tracked/Catched Objects	Adversarial Tracked/Catched Objects	Intelligent Intrateam Cooperation	Cross Team Cooperation	Holistic Planning of All Teams	Subject to Complex Constraints (*e.g.,* obstacles, communication and observational limit)
Swarmanoid (2013) [5]	Foot-bots, Hand-bots, Eye-bots	×	×	×	✓	×	✓
A Haptic-enabled System (2021) [6]	UAGs and UAVs	✓	×	✓	✓	×	×
ARCHES (2020) [7]	Rover, Drone and so on	×	×	×	✓	×	✓
A Drone Catching System (2020) [8]	VTOL-UAV and Communication Manipulator	×	×	×	✓	×	✓
A Swarm Catching Algorithm (2020) [9]	×	×	×	✓	×	×	✓
An Ocean Front Tracking System (2021) [10]	ASVs and AUVs	×	×	×	✓	×	×
A Maritime Tracking System (2020) [11]	BoB-0 units and BoB-1 units.	×	× (unspecified clearly)	✓	✓	×	✓
ECBS-CT Algorithm (2019) [12]	×	×	×	✓	✓ (minor modifications)	×	✓
S2M2 Algorithm (2021) [1]	×	×	×	✓	✓ (minor modifications)	✓	✓
Ours	Quadruped robots, Quadcopter robots and Wheeled robots (outdoor)	✓	✓	✓	✓	✓	✓

1) How to enable cooperation within a homogeneous robot team and between heterogeneous robot teams?
2) How to balance the learning complexity and the level of intelligence of the heterogeneous robot teams?

To show the ability of our framework, we demonstrate it through an implementation of HMRS in complicated catching scenarios with real-world constraints. Compared with other catching-like (tracking) research, our work is distinct from them in three ways. 1) We focus on a final real-world heterogeneous setting. 2) We consider both intrateam and cross-team cooperation with more than two teams of robots. 3) We examine the situation where the opponents can also escape intelligently. As far as we know, we are the first to consider real-world catching scenarios with more than two *intelligent* robot teams, especially in the case of having opponents also escaping intelligently. Table I summarizes the differences between our work and recent catching-like real-world HRMS. Section II-2) explains this table in detail.

In summary, we would like to argue that, to meet the complexity and dynamics of modern tasks, it is of great importance to consider adding a data-driven RL-based framework to establish a robust and intelligent HMRS. In addition, the asymmetric self-play mechanism in RL could be a convenient way to enable intelligent cooperation of different teams of heterogeneous robots, thus reducing the computation needed for the planning of heterogeneous robotic systems for real-world applications.[1] More specifically, the contributions we made are as follows.

1) We show how to model a team of heterogeneous robots under the deep multiagent RL framework to have intrateam and cross-team cooperation under realistic constraints in a complicated catching scenario. This provides some important insight for the future development of large-scale intelligent security and rescue HMRS.
2) To tackle the difficulties in training, we researched and fused two DRL techniques, namely, an asymmetric self-play method and a curriculum-based training strategy, in developing this system. We found that our proposed framework is especially suited as a method to elicit the intelligence of HMRS while being able to remain trainable for the learning process of the dynamic catching scenarios of HMRS.
3) To show that our design is both implementable in simulation and in the real world, we conducted real-world experiments consisting of three heterogeneous robot teams, including three catcher robots, two observer robots, and a communication robot, together with runner robots as opponents. We also analyzed performance differences between simulated and real-world experiments.

[1]A video version of our presentation could be found with link: https://youtu.be/tPn4RxMPmiY

II. Related Work

Our work is related to different fields of study. Here we conduct our literature review on different aspects, namely multirobot systems, catching-like problems, DRL, and the combined DRL techniques (i.e., asymmetric self-play and curriculum learning). The first two aspects are more related to the multirobot system and its applications. The latter two are more related to the methods we used in this study.

1) Multirobot Systems: Multirobot systems have turned out to be a very interesting field of robotics in recent years along with the development of computing systems' arithmetic and communication capabilities [13]. As time passed, all aspects of multirobot systems began to be considered. Whether it is task assignment from the top level or motor control learning from the bottom level, the complexity of multirobot systems has called new challenges to roboticists, ranging from multirobot task allocation [21], [22], path planning [23], [24] to exploration [25], [26]. However, except for limited works on heterogeneous settings (e.g., [2], [13], [27], [28], [29]), most of the multirobot problems focused on homogeneous settings. We conducted a systematic search of the relevant literature on heterogeneous multiple robots and found that research in this area is still mainly based on splitting the problem of heterogeneous robot control into multiple aspects, such as the "Workflow for the MRS" proposed in [13], [30]. In terms of applications, people studied agricultural scenarios [31], [32], SLAM [33], [34], and exploration [35].

2) Catching/Tracking Problem and its Heterogeneous Multirobot Variants: The catching problem was studied mainly by the single-robot community, with the manipulator as a major robot in the applications [36], [37]. When it comes to multirobot systems, a few works including [9], [38] scarcely considered the catching problem. Compared with our work, they only studied the homogenous swarm robots with physical quantities as observations. The moving target in this work is preprogrammed and nonintelligent. Unlike the catching problem, the tracking problem was one of the great concerns of robotics during its initial development. At that time, the overarching field was still called cybernetics and pioneered by Norbert Wiener and W. Ross Ashby [39], [40]. Considering that it is only less than 60 years since the idea was first proposed, it is amazing to see that along with the development of other technologies, the tracking problem has been given a new meaning. For example, people not only have considered different tracking observations including camera-based (e.g., [41], [42]) or LiDAR-based (e.g., [43], [44]) devices but also have considered more sophisticated tracking systems. In recent years, as more and more devices are equipped with reliable communication methods, systems like grouped unmanned aerial vehicles (UAVs) [45] or modular robots [46] are also used for tracking. Due to the tradeoff between scalability and heterogeneity, previous works discussing the real-world intelligent cooperation between multiple heterogeneous teams of robots were mainly limited to two kinds of robots, e.g., wheeled robots teams and quad-copter robots teams. They normally focus on simpler tasks like coalition formation [13]. Table I compares different recent real-world catching/tracking-like tasks with our system. Not only our system can achieve intrateam and cross-team cooperation, but our system can deal with adversarial tracked objects intelligently. With the capability to include complex constraints, our system also plans the cooperation of all teams holistically, meaning that our system processes the advantages of simplicity and has a lower degree of dependency on domain experts [47]. The tracking problem is also studied in the area of multiagent path finding, with the evaluations normally limited to simulations. To date, the optimization-based methods, which are evolved from conflict-based search [48] or priority-based search [49], claim to have the ability to perform the reach-avoid behavior in continuous planning for multiple agents [1]. To evaluate the performance of this state-of-the-art algorithm in the multiagent path finding problem in our scenario, we conduct a bridging study to compare its performance with our approach.

3) Multiagent Deep Reinforcement Learning Algorithms: Reinforcement learning (RL) has been used since the early days of robotics. Over years, the RL has been applied to different areas of robotics, including path planning [50], collision avoidance [51], robot control [52], vision representation learning [53], and so on. Together with the emergence of different machine learning techniques like meta learning or architectures like transformer [54], people are able to develop more and more complicated robot systems. For the purpose of enabling cooperative behaviors among robots, people started to also develop algorithms that can attribute rewards more efficiently within the agent groups. Famous examples include MA-PPO [55] which was able to train a five-agent team to win a competition against humans. Other algorithms include QMIX [56] and QPLEX [57] that can be used for different multiagent scenarios. The cooperation-aware algorithm we are using is called multiagent posthumous credit assignment (MA-POCA) [58]. Compared with single agent DRL algorithms, this algorithm analyzes the reward signals and assigns them to different agents for their corresponding performance and inherited the idea of counterfactual baseline in the COMA algorithm [59], [60]. The MA-POCA algorithm can add/remove robots dynamically during training. This is achieved by utilizing a residual self-attention mechanism, which is architecturally similar to those used in the vanilla Transformer architecture but without positional encodings. Moreover, the MA-POCA algorithm also follows the centralized training and decentralized execution (CTDE) paradigm, allowing intelligent cooperation during the training, and decentralized deployment for scalability [61] during execution.

4) Asymmetric Self-Play and Curriculum Learning: We further extend the multiagent RL algorithms mentioned above with a combination of asymmetric self-play and curriculum learning to achieve training of our catching HMRS. The self-play mechanism has been studied in the area of the game for more than half a century [62], [63]. After years of development, it is now one of the mechanisms that enabled many state-of-the-art intelligent systems like AlphaGo [64]. In robotics, many works have further extended the basic self-play mechanism and used its asymmetric version for different applications. For example, the authors in [65], [66], and [67] used asymmetric self-play for automatic curriculum learning and [66] used asymmetric self-play for robot manipulation. In our setup, to train an intelligent heterogeneous robot, a few asymmetric adversarial agents are also placed in the training process as opponents. We hypothesize that there are two advantages of applying this mechanism in training our heterogeneous systems. The first is that as the runner robots are learning to avoid the catcher robots, the catcher robots are able to gradually develop new techniques to counter the runner robots. This makes sure that the catching robots could develop more and more intelligent strategies over time. Second, since the team of catcher robots only gets rewards when they are close enough to runner robots, our problem is a sparse reward problem. Assigning a sparse reward is intuitive for humans but it also requires more exploration during training. In general, setting up opponents increases the exploration of our catcher robots, which further accelerates learning. Curriculum learning was first proposed by Elman [68] as a psychological concept, and it was then argued by Bengio that it could be used in the DRL systems [69]. For complicated tasks, it would be easier for the algorithms to learn intended outcomes by following an "easy-to-hard" procedure. We use curriculum learning from two perspectives. On one hand, we use curriculum learning to gradually ensure the performance of the whole catching team. On the other hand, compared with the noncurriculum case, we try to reduce the overall training complexity, making sure that we are able to get better results with fewer overall training steps.

III. Interaction Modeling

In this section, we introduce the basic concepts used in our methods starting from problem formulation to robot modeling.

TABLE II
CONFIGURATIONS OF DIFFERENT ROBOTS IN SIMULATED AND REAL EXPERIMENTS

Name	Speed in Simulation (m/s)	Speed in Real-world (m/s)	Number in Simulation	Number in Real-world
Catcher Robots	2	0.2	3	3
Observer Robots	5	0.5	2	2
Communication Robots	1.8	0.18	2	1
Runner Robots	2.2	0.22	2	2

A. Problem Description

We consider general searching and catching problems through the cooperation of heterogeneous robot teams in real-world scenarios. Specifically, we consider a setup of having a group of observer robots, a group of communication robots, and a group of catcher robots. These robots need to cooperate in searching and catching several intelligent moving targets (named runner robots) in a dynamic environment. Due to the heterogeneity, different groups of robots need to take roles that match their functionalities. The system also needs to adapt to real-world conditions (e.g., higher communication and computation capabilities). To further describe different robots, the characteristics are listed as follows.

1) *Catcher Robots*: A group of robots that are employed to take the primary responsibility of catching the intelligent moving targets. The robots can see a limited range of their surroundings. Compared with a runner robot, the catcher robot is a bit slower.
2) *Observer robots*: Observer robots are fast-moving flying vehicles. Unlike ground robots, observer robots cannot check the tracked target directly but are able to see a much larger range. To support the catching behavior, they are assigned to catch the team of runner robots in the air and report their locations to the team of catcher robots.
3) *Communication Robots* The communication robots are slower but more stable vehicles. They are able to equip larger and heavier computing and communication devices. However, as supporting units, they are not able to catch moving targets. The communication robots collect the poses of all robots and act jointly as a distribution center, sharing some of the information with other robots. This is ensured by making sure their distances to other robots in the catching team stay in a certain range (e.g., 45 m).
4) *Runner Robots*: The team of runner robots is modeled as an opponent team to train the whole catching team through asymmetric self-play. Compared with the catcher robot, the runner robot is a bit faster. This makes sure that the team of catcher robots needs to cooperate intelligently to catch the runner robot.

The configurations of simulation and real-world experiments are a bit different. Table II summarizes the characteristics of each robot in different experiments. Different robots are modeled in Unity simulation and tracked by a motion capture system in the indoor real-world environment and by an ultra-wideband localization system in the outdoor real-world environment.

B. Preliminaries

Before introducing our method, we first describe the basic mathematical modeling of RL.

1) Markov Decision Process (MDP): In RL, we model a single agent's interaction with the environment as an MDP tuple, $(\mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma)$. $\mathcal{S}$ represents the state space, and $\mathcal{A}$ represents the action space. $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ denotes the transition probabilities. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to r$ is a reward function, and $\gamma \in [0, 1]$ is a discount factor. Given an environment, where μ is the distribution of the initial state, the agent tries through trial and error. The agent tries to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected discounted reward with the maximized policy defined as $\pi^* = \arg\max_\pi \mathbb{E}^\pi_{s_0\sim\mu}[V^\pi(s)]$, where $V^\pi(s) = \mathbb{E}^\pi[\sum_{t=0}^{\infty} \gamma^t r_t | s = s_0]$. The MDP is defined generally for a single agent and to include the consideration of the multiagent, we need to extend the definition of MDP.

2) Decentralized Partially Observable MDP (Dec-POMDP): To analyze our environmental settings, we model our system as a fully cooperative multiagent task in terms of a Dec-POMDP [70] defined by an extended MDP tuple $M = (\mathcal{N}, \mathcal{S}, \mathcal{A}, P, \Omega, \mathcal{O}, r, \gamma)$, where $\mathcal{N} \equiv \{1, 2, \ldots, N\}$ is a finite set of agents and $\mathcal{S}$ is a finite set of global states. At each time step, every agent $i \in \mathcal{N}$ chooses an action $a_i \in \mathcal{A}$ on a state $s_i \in \mathcal{S}$, which results in a collective action $\mathbf{a} \equiv \{a_1, a_2, \ldots, a_n\} \in \mathcal{A}^n$. Our system operates in a partially observable environment, in which each agent i receives a partially observable $o_i \in \Omega$ following the observation probability function $\mathcal{O}(o_i|s, a_i)$, resulting in an action-observation history $\tau_i \in \mathcal{T}$ with corresponding constructed individual policy $\pi(a|\tau_i)$ to jointly maximize team performance. The formal objective is to find a joint policy $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ that maximizes a joint value function $V^\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \pi]$.

C. Agent Modeling

In our HMRS, two essential characteristics need to be thoroughly considered. The first is heterogeneity, which means that we need to specify the input–output problem of the control chain due to different robot configurations. An ideal heterogeneous system controlled by a learning algorithm should ignore some differences in configuration, and thus, have a similar form of input and output. Also, for the whole system to be trained in a limited time, the multirobot system must also utilize the data generated by each homogeneous robot in each group. When considering the heterogeneous setup and the multirobot system together, we also have to consider an implementable way to establish a mechanism for cooperation between multirobot systems of each configuration. Moreover, we note here that our setup is based on the assumption that each robot knows its global position through GPS or other localization systems. Fig. 4. To have a more rigorous description, we will define our problem

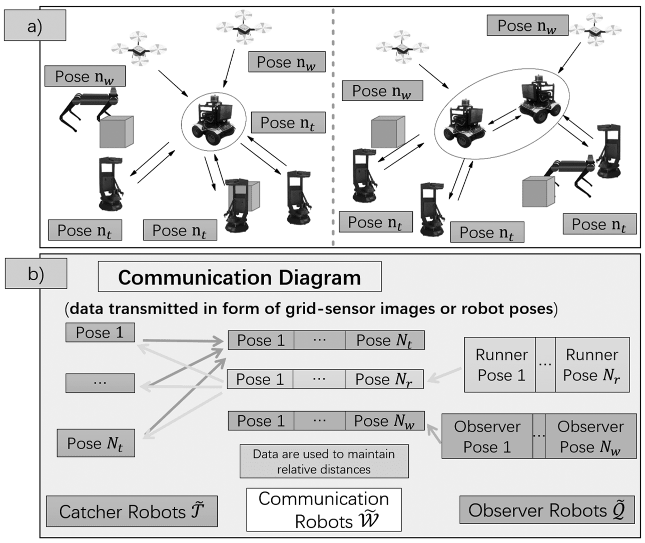


Fig. 2. Illustration of the communication mechanisms between different groups of robots in two scenarios. (a) The left side shows the situation where only one communication robot is utilized for communication, and the right side shows the situation in which a team of two communication robots needs to collect the poses of all robots and act jointly as a distribution center. In our setup, the wheeled robots team needs to take care of the communication functionality. Consequently, the communication robots need to try to optimize their poses over the whole catching scenario. (b) A communication diagram showing paths of information transmitted among the team of robots.

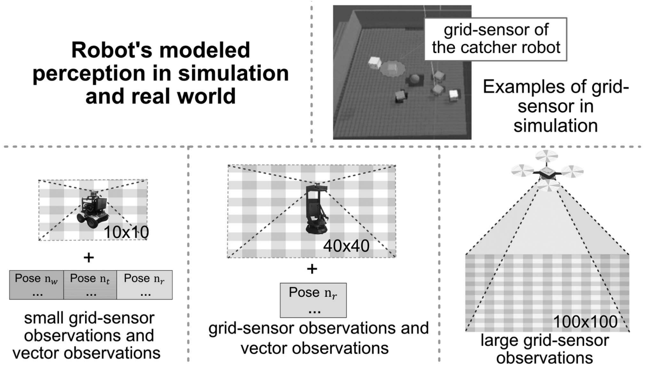

Robot's modeled perception in simulation and real world
grid-sensor of the catcher robot
Examples of grid-sensor in simulation
10x10
+
Pose n_w
Pose n_t
Pose n_r
small grid-sensor observations and vector observations
40x40
+
Pose n_r
grid-sensor observations and vector observations
100x100
large grid-sensor observations

Fig. 3. Illustration of each kind of robot in the catching team and their corresponding primary type of perception in both simulation and the real world. The communication robot receives a small grid-sensor observation and a vector observation (e.g., locations and velocities). The catcher robots and observer robots receive, respectively, a smaller and larger grid-sensor as inputs. If the observer robots are present in the catching team, they send the location of the runner robots to the catcher robots.

mathematically as an optimization problem in the following section.

1) Mathematical Problem Definition: We consider a collaborative task involving three *teams* of heterogeneous robots, noted as $\tilde{\mathcal{H}} \equiv \{\tilde{\mathcal{T}}, \tilde{\mathcal{Q}}, \tilde{\mathcal{W}}\}$, where $\{\tilde{\mathcal{T}}, \tilde{\mathcal{Q}}, \tilde{\mathcal{W}}\}$ are respective sets containing catcher robots, observer robots, and communication robots. They are defined as $\tilde{\mathcal{T}} \equiv \{\mathcal{T}_1, \ldots, \mathcal{T}_{n_t}, \ldots, \mathcal{T}_{N_t}\}$, $\tilde{\mathcal{Q}} \equiv \{\mathcal{Q}_1, \ldots, \mathcal{Q}_{n_q}, \ldots, \mathcal{Q}_{N_q}\}$, $\tilde{\mathcal{W}} \equiv \{\mathcal{W}_1, \ldots, \mathcal{W}_{n_w}, \ldots, \mathcal{W}_{N_w}\}$, where N_t, N_q, N_w are total numbers of robots in each respective team. For each team defined in $\tilde{\mathcal{H}}$, there exists an extended MDP named $M_{\tilde{\mathcal{T}}}$, $M_{\tilde{\mathcal{Q}}}$, and $M_{\tilde{\mathcal{W}}}$ (see Section III-B2) with the state of each robot at time t defined as $\mathcal{T}^t_{n_t}$, $\mathcal{Q}^t_{n_q}$, and $\mathcal{W}^t_{n_w}$. For simplicity, we define a set containing states of each team of robots at arbitrary time t as $\tilde{\mathcal{T}}^t = \{\mathcal{T}^t_1, \ldots, \mathcal{T}^t_{n_t}, \ldots, \mathcal{T}^t_{N_t}\}, \tilde{\mathcal{Q}}^t, \tilde{\mathcal{W}}^t$. The algorithms applied to each team should ensure that, given reward functions, the teams of heterogeneous robots will converge to a cooperative strategy. To stimulate the intelligence of the heterogeneous robots, we also define a team of opponents as $\tilde{\mathcal{U}} \equiv \{\mathcal{R}_1, \ldots, \mathcal{R}_{n_w}, \ldots, \mathcal{R}_{N_r}\}$, where N_r represents the number of the runner robots. The overall goal of our system is to make sure that the catcher robots team stays in close range with the runner robots team subject to some constraints, e.g., communicating with each other. Formally, we define it as an optimization problem

$$\begin{aligned} \arg\min_\theta \quad & f_{\pi_\theta}\left(\tilde{\mathcal{T}}, \tilde{\mathcal{Q}}, \tilde{\mathcal{W}}, \tilde{\mathcal{U}}\right) \\ \text{s.t.} \quad & g_{\pi_\theta}\left(\tilde{\mathcal{T}}, \tilde{\mathcal{Q}}, \tilde{\mathcal{W}}\right) \le \mathcal{C}_1 \\ & g^*_{\pi_\theta}\left(\mathcal{I}_{n_i}, \tilde{\mathcal{W}}\right) \le \mathcal{C}_2 \forall \mathcal{I}_{n_i} \in \left(\tilde{\mathcal{T}} \cup \tilde{\mathcal{Q}}\right) \\ & k_{\pi_\theta}\left(\tilde{\mathcal{Q}}, \tilde{\mathcal{U}}\right) \le \mathcal{C}_3 \\ & c_{\pi_\theta}\left(\tilde{\mathcal{T}}, \tilde{\mathcal{Q}}, \tilde{\mathcal{W}}\right) = 0 \end{aligned} \tag{1}$$

where $f_{\pi_\theta}(\cdot)$ is defined as

$$f_{\pi_\theta}\left(\tilde{\mathcal{T}}, \tilde{\mathcal{Q}}, \tilde{\mathcal{W}}, \tilde{\mathcal{U}}\right) = \int \hat{f}_{\pi_\theta}\left(\tilde{\mathcal{T}}^t, \tilde{\mathcal{Q}}^t, \tilde{\mathcal{W}}^t, \tilde{\mathcal{U}}^t\right) \mathrm{d}t \tag{2}$$

$$= \int \sum_{\mathcal{R}^t_{n_r} \in \tilde{\mathcal{U}}^t} \min_{\mathcal{T}^t_{n_t} \in \tilde{\mathcal{T}}^t} \left(\mathrm{dist}\left(\mathcal{R}^t_{n_r}, \mathcal{T}^t_{n_t}\right)\right) \mathrm{d}t. \tag{3}$$

We aim to reduce the distances between the catcher robots and the runner robots. dist$(\cdot,\cdot)$ is a function that calculates the Euclidean distance between two robots $\mathcal{R}^t_{n_r}$ and $\mathcal{T}^t_{n_t}$ at time t. In the MDP setup, the function can then be discretized as $\sum_{t=0}^{T} \sum_{\mathcal{R}^t_{n_r} \in \tilde{\mathcal{U}}^t} \min_{\mathcal{T}^t_{n_t} \in \tilde{\mathcal{T}}^t} (\mathrm{dist}(\mathcal{R}^t_{n_r} \mathcal{T}^t_{n_t}))$. Similarly, function $g_{\pi_\theta}(\cdot)$ is defined as

$$g_{\pi_\theta}\left(\tilde{\mathcal{T}}, \tilde{\mathcal{Q}}, \tilde{\mathcal{W}}\right) = \int \sum_{\mathcal{I}^t_{n_i} \in \left(\tilde{\mathcal{T}}^t \cup \tilde{\mathcal{Q}}^t\right)} \min_{\mathcal{W}^t_{n_w} \in \tilde{\mathcal{W}}^t} \mathrm{dist}\left(\mathcal{I}^t_{n_i}, \mathcal{W}^t_{n_w}\right) \mathrm{d}t \tag{4}$$

$$= \sum_{t=0}^{T} \sum_{\mathcal{I}^t_{n_i} \in \left(\tilde{\mathcal{T}}^t \cup \tilde{\mathcal{Q}}^t\right)} \min_{\mathcal{W}^t_{n_w} \in \tilde{\mathcal{W}}^t} \mathrm{dist}\left(\mathcal{I}^t_{n_i}, \mathcal{W}^t_{n_w}\right) \tag{5}$$

indicating that the communication robot team needs to reduce the distance between the communication robot team and other catching robots down to an arbitrary threshold $\mathcal{C}_1$. This is because the communication robots need to keep the communication distance short for the whole team (see Fig. 2). After a careful and thorough literature investigation regarding the modeling of moving base stations for multirobot communication, we think an additional reward should be considered to satisfy the communication constraints individually. Previous work has shown that in the current wireless communication, the constraints of the

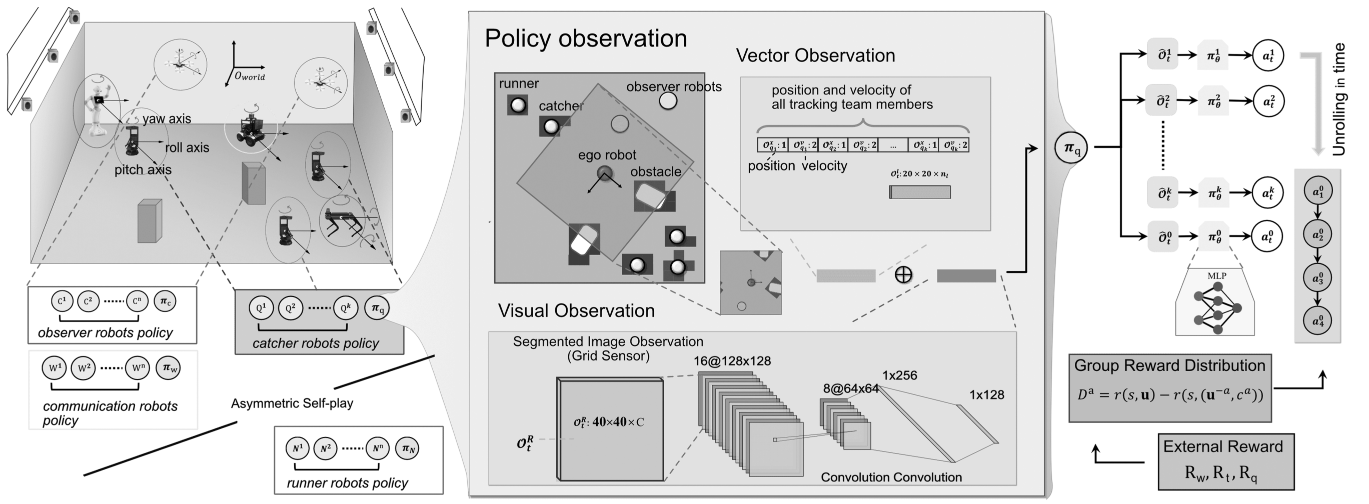


Fig. 4. Illustration of our real-world system and the training framework we used. In the top left, we can see the scenario setup for the whole experiment, while the bottom left shows the strategies for each corresponding team. The middle part shows the input and output structures of each policy network. The right side shows the relationship between the policy network and the credit assignment of the reward function.

mobile base station need to maximize the coverage area based on the minimum transmitted power [71]. Among all proposals [71], [72], [73], we decided to maintain a communication radius as mentioned in [73], which is widely accepted by the mobile communication community and experiment with it together with the distance reward. The communication range for communication robots is specified as $g^*_{\pi_\theta}(\mathcal{I}_{n_i}, \tilde{\mathcal{W}})$ in Formula 1 and should be smaller than an arbitrary threshold $\mathcal{C}_2$ (e.g., 45 m). Function $k_{\pi_\theta}(\cdot)$, defined as

$$k_{\pi_\theta}(\tilde{\mathcal{Q}},\tilde{\mathcal{U}}) = \int \sum_{\mathcal{R}^t_{n_r}\in\tilde{\mathcal{U}}^t} \min_{\mathcal{Q}^t_{n_q}\in\tilde{\mathcal{Q}}^t} \left(\text{dist}\left(\mathcal{R}^t_{n_r}, \mathcal{Q}^t_{n_q}\right)\right) \text{dt} \quad (6)$$

$$= \sum_{t=0}^{T} \sum_{\mathcal{R}^t_{n_r}\in\tilde{\mathcal{U}}^t} \min_{\mathcal{Q}^t_{n_q}\in\tilde{\mathcal{Q}}^t} \left(\text{dist}\left(\mathcal{R}^t_{n_r}, \mathcal{Q}^t_{n_q}\right)\right) \quad (7)$$

regularizes the observer robots team so that it can stay above the runner robots and provide the locations of the runner robots to the team of catcher robots. $\mathcal{C}_3$ is also an arbitrary real number. Lastly, the whole system needs to avoid obstacles and each other in the system. In this case, the $c_{\pi_\theta}(\tilde{\mathcal{T}},\tilde{\mathcal{Q}},\tilde{\mathcal{W}})$ is defined as

$$c_{\pi_\theta}(\tilde{\mathcal{T}},\tilde{\mathcal{Q}},\tilde{\mathcal{W}}) = -\int \sum_{\mathcal{I}^t\in\tilde{\mathcal{K}}^t,\mathcal{J}^t\in\tilde{\mathcal{K}}^t,\mathcal{I}^t\neq\mathcal{J}^t} \hat{c}\left(\mathcal{I}^t,\mathcal{J}^t\right) \text{dt} \quad (8)$$

$$= -\sum_{t=0}^{T} \sum_{\mathcal{I}^t\in\tilde{\mathcal{K}}^t,\mathcal{J}^t\in\tilde{\mathcal{K}}^t,\mathcal{I}^t\neq\mathcal{J}^t} \hat{c}\left(\mathcal{I}^t,\mathcal{J}^t\right) \quad (9)$$

where $\tilde{\mathcal{K}}^t = \tilde{\mathcal{T}}^t \cup \tilde{\mathcal{Q}}^t \cup \tilde{\mathcal{W}}^t$ and $\hat{c}(\cdot,\cdot)$ is defined as a piecewise function

$$\hat{c}(\mathcal{I},\mathcal{J}) = \begin{cases} 1, & \text{dist}(\mathcal{I},\mathcal{J}) \leq \epsilon \\ 0, & \text{dist}(\mathcal{I},\mathcal{J}) > \epsilon \end{cases} \quad (10)$$

where ϵ is an arbitrarily small number indicating a safe distance between robots.

2) Semiunified Observational Model: As stated previously, an extended MDP is defined as $M = (\mathcal{N},\mathcal{S},\mathcal{A},P,\Omega,\mathcal{O},r,\gamma)$. In the following text, we will specify the observation $\mathcal{O}$ and action $\mathcal{A}$ of each kind of agent. We aim to have observational models of the same structure. In this way, we are able to use similar algorithmic structures to process the input from all kinds of robots. Fig. 3 shows the simplified representations for each kind of robot. The representation can be described as a $N_{\text{grid}} \times N_{\text{grid}} \times C$ tensor, where N is the input length and C is the number of all categories. This is inspired by the grid-like sensors proposed by [74] and [75]. This makes sure that it is both learnable and implementable using signals from the real world. Fig. 3 shows the segmented perception of three different kinds of robots. Among them, to simulate the real situation, the observer robots use a large tensor($N_{\text{grid}} = 100$) as their input signal. The catcher robots receive a mid-size tensor($N_{\text{grid}} = 40$). The communication robots have a small observation ($N_{\text{grid}} = 10$) and receive the locations of other robots in the catching team through communication.

3) Decision-Making Level Output: While training and executing, all MA-POCA algorithms generate a set of experiences. Depending on the robot types, different groups may have different action spaces. For the team of catcher robots and communication robots, we define the action space as a 2-D continuous vector space consisting of linear and angular velocities (e.g., differential drive kinematics). Given the max linear velocity $v^x_{\max}$ and max angular velocity $\omega^x_{\max}$ of a robot $x \in \tilde{\mathcal{T}} \cup \tilde{\mathcal{W}}$, the sampled action $a^x_t \in \{-1,1\}$ is scaled by multiplying $v^x_{\max}$ or $\omega^x_{\max}$ to give the desired velocity commands. The team of quadcopter robots has omnidirectional drive kinematics, meaning that the sampled actions have a 3-D continuous vector action space. In adddition, we also collect trajectories of different robots during the real-world experiment. If a robot diverges too much from the desired location due to turbulence, it changes to a pure pursuit strategy and follows a trajectory generated by the algorithm. Trajectories contain a list of tuples defined as

$(x^t, y^t, z^t, \theta^t) = \mathcal{I}^t \forall \mathcal{I}^t \in \tilde{\mathcal{K}}^t$, where x, y, and z represent the global locations of the robot and θ indicates the current pose of the robot. When high-level decisions are made, a PID controller ensures the robot will go to the given position (x, y, z) with a heading direction θ.

4) Communication Between Different Multirobot Teams: To ensure cooperation between groups of heterogeneous robots for this task while considering the differences between simulation and the real world, we will have to design and regulate the overall communication process. Theoretically, a team of catcher robots should have the capabilities to support massive connectivity and can deliver fast and low-latency communications. However, many systems have the limitation that they lack the ability to generate a strong enough signal for each device. One solution to this problem is to use communication robots as a dynamic communication center by taking advantage of the fact that they can carry heavier equipment and can have more power. Fig. 2 shows the two different communication schemes of the heterogeneous robot teams in HMRS. To implement a flexible communication mechanism, in the following section, we will describe how we use reward functions to encourage the communication behavior.

5) Reward for Heterogeneous Robot Teams: RL algorithms use reward functions as indicators to optimize their policies. When a single-agent algorithm is used, the rewards are given to the individual robots. When the algorithm MA-POCA is used, the rewards are given to different teams of heterogeneous robots. In this section, we introduce how rewards are defined for each team of robots. For the team of catcher robots, the reward function could be decomposed into two parts. The first part is the common part for collision avoidance, allowing them to stay away from each other and obstacles. The reward for obstacle avoidance is defined as

$$R_{ca}^{\tilde{\mathcal{X}}} = -\sum_{t=0}^{T} \sum_{\substack{\mathcal{I}^t \in \tilde{\mathcal{X}}^t, \mathcal{J}^t \in \{\tilde{\mathcal{K}}^t \cup \tilde{\mathcal{O}}^t\} \\ \mathcal{I}^t \neq \mathcal{J}^t}} \hat{c}\left(\mathcal{I}^t, \mathcal{J}^t\right) \tag{11}$$

where $\tilde{\mathcal{X}}^t$ indicates the set of robots that would like to stay away from each other and obstacles, $\tilde{\mathcal{O}}^t$ is a set containing all obstacles.

The second part includes a positive reward signal when any catcher robot approaches a runner robot, indicating the general purpose of the catcher robot team. When a catcher robot is able to stay in a close range with a runner robot, it receives a sparse reward r^t at time t, with the final reward defined as

$$R_t = \sum_{t=0}^{T} \sum_{\substack{\mathcal{I}^t \in \tilde{\mathcal{K}}^t \\ \mathcal{J}^t \in \tilde{\mathcal{U}}^t}} \hat{c}\left(\mathcal{I}^t, \mathcal{J}^t\right) + R_{ca}^{\tilde{\mathcal{T}}}. \tag{12}$$

The team of observer robots tracks the team of runner robots in the air and reports their locations to the team of catcher robots. As a consequence, they will try to stay close to the runner robots' team. The reward function of the observer robots is defined as

$$R_q = -\sum_{t=0}^{T} \sum_{\mathcal{R}_{n_r}^t \in \tilde{\mathcal{U}}^t} \min_{\mathcal{T}_{n_t}^t \in \tilde{\mathcal{T}}^t} \left(\text{dist}\left(\mathcal{R}_{n_r}^t, \mathcal{Q}_{n_t}^t\right)\right) + R_{ca}^{\tilde{\mathcal{Q}}^t}. \tag{13}$$

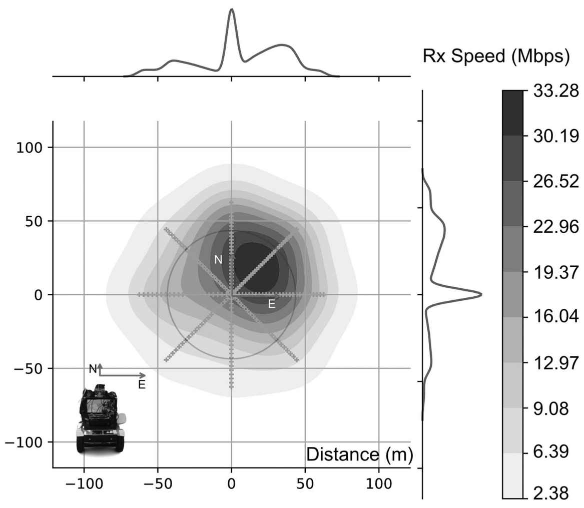


Fig. 5. Estimated Rx bandwidth around the communication robot. From data collected in polar space, we use a radial basis function with a Gaussian kernel to estimate the Rx speed. The red circle shows our assumed range of communication.

Practically, for faster convergence, when the number of runner and observer robots is equal, we only calculate the minimization operation once. In this case, the reward for each observer robot is always its negative distance from the closest runner robot, when the scene is initialized, meaning that the association between an observer robot and a runner robot is decided at the beginning of the scenario. The main function of the communication robots is to keep the distances to the other robots in the catching team short and stay in the communication range as long as possible. Fig. 5. To make sure of that, the reward function for the communication robot is defined as

$$R_w = -\sum_{t=0}^{T} \sum_{\mathcal{I}^t \in (\tilde{\mathcal{T}}^t \cup \tilde{\mathcal{Q}}^t)} \left(R_w^{\min} - \omega_w \mathcal{H}\left(R_w^{\min} - \mathcal{C}_2\right)\right) + R_{ca}^{\tilde{\mathcal{W}}^t} \tag{14}$$

where $R_w^{\min} = \min_{\mathcal{W}_{n_w}^t \in \tilde{\mathcal{W}}^t} \text{dist}(\mathcal{I}^t, \mathcal{W}_{n_w}^t)$, $\mathcal{H}(\cdot)$ is the Heaviside step function to indicate possible catastrophic results if other robots run out of the communication distances, and ω_w is the weight assigned to the step reward function (e.g., 0-0.1). Among the three different reward functions, R_c is sparse and R_w and R_p are nonsparse rewards. For the whole system, the final reward is the addition of all of them $\mathcal{L}_R = R_t + R_w + R_c$. After the reward is assigned to each robot team, the reward is assigned to each robot using the MA-POCA algorithm. During the training, $\mathcal{L}_R$ is not easy to optimize while having all the robots running either cooperatively (i.e., catcher robots, observer robots, and communication robots) or competitively (i.e., runner robots). Here we follow a curriculum strategy to train $\mathcal{L}_R$. In the first stage of training in the simulation, catcher robots are assumed to be able to communicate with each other. They are also given the global locations of runner robots, using the standard domain randomization technique [76]. In the second stage, the observer robots are trained to follow the runner robots in the air. Once the observer robots see the runner robots, they can send the runner

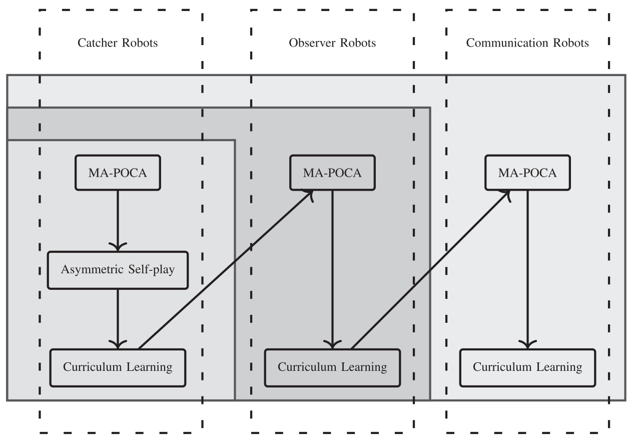


Fig. 6. Development of our method through systematic experiments. As indicated by arrows, we start by testing MA-POCA's performance for a team of catcher robots. Following that, we add two runner robots for asymmetric self-play to stimulate the intelligence of the catcher robots team. Then we continue with adding a curriculum to avoid more and more obstacles in this process. From left to right, the same process is also repeated for observer robots and communication robots. Green, red, and blue colors represent the first, second, and third stages of development.

robots' global locations to the catcher robots. In the third stage, the team of communication robots is trained to take care of signal coverage problems. Three steps of the curriculum are described in Fig. 6, and they are needed to stabilize the overall training performance.

6) Design of Self-Play Mechanism and Reward for the Runner Robots Team: A group of runner robots is trained as a team to stimulate the intelligence of the heterogeneous robot teams. For this purpose, we follow the definition in paper [63], designing a self-play module or training scheme by using the notions of the *menagerie* $\boldsymbol{\pi^o}$, where all played policies are kept in, with the *policy sampling distribution* Ω, and the *gating function* G, specified by the tuple $< \Omega(\cdot|\cdot,\cdot), G(\cdot|\cdot,\cdot) >$. The gating function must decide which policy is kept and which policy is discarded. More specifically, Ω is a policy sampling distribution over the menagerie $\boldsymbol{\pi_k}$ and current policy π_k^t. As the purpose of our self-play scheme is to stimulate the other group, we need to 1) train the runner robots team to have some level of intelligence 2) but also give enough time for the heterogeneous robot team to train to win over the runner robots team. We employ a time-constraint asymmetric self-play strategy [63], in which we maintain two menageries $\boldsymbol{\pi_k}$ and $\boldsymbol{\pi_r}$, which represent collected policies from the catcher robots group and the runner robots group, respectively. The sampling function associated with sampling distribution Ω of the catcher robots group is specified as

$$\phi_{\Omega_k}(\pi_r|\boldsymbol{\pi_k}, \pi_k^t) = \begin{cases} \pi_r^{t+1} & \forall\{t+1\} \leq T \\ \pi_r^T & \forall\{t+1\} > T \end{cases} \tag{15}$$

meaning that both groups use the newest policies to play with each other up until a moment T. After that moment T, the runner robots team will always use the same policy π_r^T, giving more time to the catcher robots team to learn. Meanwhile, the sampling function associated with sampling distribution Ω of the runner robots group is specified as

$$\phi_{\Omega_k}\left(\pi_t|\boldsymbol{\pi_r}, \pi_r^t\right) = \pi_k^{t+1}\forall\{t+1\} \tag{16}$$

which indicates that the runner robots team will always choose the newest catcher robots policy as the opponent. Before time T, to stimulate the runner robots team to run intelligently, a reward R_r is given to the runner robots team, which is opposite to the reward given to the catcher robots team

$$R_r = -\sum_{t=0}^{T} \sum_{\mathcal{I}^t \in \tilde{\mathcal{U}}^t, \mathcal{J}^t \in \tilde{\mathcal{T}}^t} \hat{c}\left(\mathcal{I}^t, \mathcal{J}^t\right). \tag{17}$$

This design makes sure that the catcher robots team and the runner robots team will compete against each other, iteratively increasing both teams' intelligence until time T.

IV. Experiments and Results

To show our method's performance in simulation and real-world heterogeneous multirobot scenarios, we gradually demonstrate our development through different experiments. In the following text, we first demonstrate the performance of our system in simulation. We show the comparison between two conditions, namely, with curriculum learning and without curriculum learning. We also demonstrate intelligence elicited by the asymmetric self-play method through a few examples. Moreover, we conduct a few ablation studies to see which observational formation is better for our system and compare the performances of single-robot and multi-robot teams to show the power of cooperative intelligence. Finally, we show how we implemented our system in the real world and demonstrate with experiments that our system can achieve predefined requirements. In the real-world experiment, we define the criteria for successful catching behavior and compare the success rate in the simulated and the real-world conditions. In addition, we also compare the obstacle avoidance abilities in both simulation and real-world conditions.

A. Our Setup

Two different kinds of setups are utilized in this study, namely, a simulated setup and a real-world setup. The simulated setup serves as a theoretical investigation before real-world experiments.

1) Simulated Setup: To demonstrate our method capabilities in the real world, we first conduct experiments in simulated environments through curriculum-based staged developments, showing that our methods are able to learn a catching strategy by intelligently utilizing the self-play mechanism and different functionalities of the teams of the heterogeneous robots. The simulated environments are reasonable abstractions of real-world scenarios on three levels. On the first level, the opponent runner robots are trained with multiagent RL algorithms, giving them some level of intelligence. This will then further stimulate the catcher robots to develop more intelligent strategies. Simultaneously, the environment consists of randomly generated obstacles, reducing the gap between simulated and real environments. Lastly, different heterogeneous robot teams' abilities are modeled fully according to their real-world functionalities,

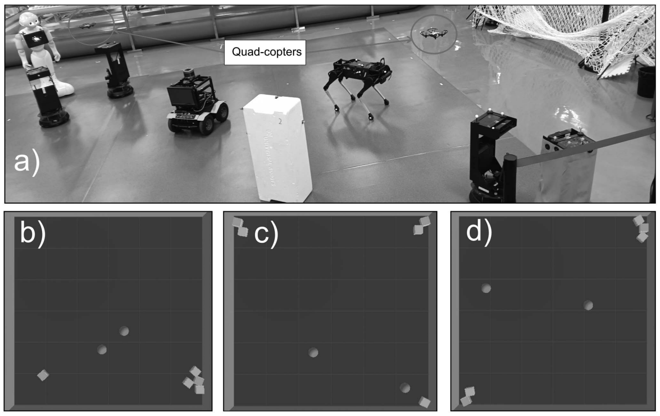


Fig. 7. Examples of different cases of entering the equilibrium in both simulation and real world. In this figure, purple blocks represent runner robots and the blue robots represent the catcher robots. (b)–(d) represent three different cases of equilibrium in simulation. (a) Equilibrium in the real world.

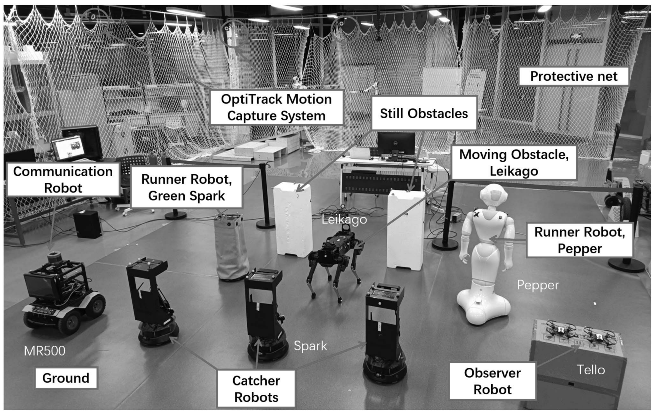


Fig. 8. Different components in the indoor real-world experiment. Except for the protective equipment, there are three catcher robots, one moving obstacle, two static obstacles, two runner robots, two observer robots, and a communication robot in the scenario. The names of robots are marked with white labels around them. In heterogeneous robotic systems, the rational allocation of work according to the different tasks is an important part of ensuring the completion of tasks.

ensuring that the modeling gap is minimized in our study. More specifically, we construct our scenarios in Unity ML-Agent [75], which is a general platform for intelligent agents. Different developmental stages of simulated experiments are defined in Fig. 6, which illustrates the stage-by-stage development of our baseline system. From left to right, we gradually developed the team's cooperative strategy of catcher robots, observer robots, and communication robots, respectively. For each team of robots, we also have a two-step or three-step team building schedule. In this schedule, we first test the performance of the robot teams for generating cooperative behavior. Then we continue with introducing the runner robots as a mechanism of asymmetric self-play. Finally, we also designed a learning curriculum for different robots. The goal of the catcher robots is to chase and stay close to the runner robots; the goal of observer robots is to keep the runner robots in their observations so that they can send the locations to the catcher robots. The purpose of the communication robots team is to keep distances to other robots short in the catching team and stay in the communication range as long as possible. Fig. 7 shows an example of our basic setup of a team of catcher robots interacting with two runner robots. In this figure, red blocks represent the randomly initialized obstacles, blue blocks represent the catcher robots team, and purple robots represent the runner robots.

2) Indoor Real-World Setup: The real-world indoor experiment was carried out in a laboratory, in which we established a testing area similar to our simulated environment but with two major modifications. The first major modification is the size of the area. Due to the real-world limitation, we reduced the area's size in accordance with the experimental conditions. The second modification is the obstacle. In the simulated experiment, we have 20 randomly generated obstacles, and in the real world, we have two randomly generated obstacles due to the size limit of the area. In this environment (see Fig. 7), we used three different robots to represent three teams of heterogeneous robots. For the team of catcher robots, we used three SPARKs.[2] In this scenario, they are assigned to collaboratively find and catch a team of two runner robots while avoiding obstacles and each other. The SPARK we adopted is a differentially driven mobile robot, which is currently widely used for research and education purposes. The observer robot we used is the tello robot provided by DJI[3] and the communication robot we used is MR500 from Robot++.[4] We used Pepper robot[5] and a green spark robot as the runner robots. A Leikago[6] robot is also adopted as a moving obstacle for testing purposes. Fig. 8 shows all the experiment equipment. Except for the robots, the other setups are described as follows.

1) *Computer Setup:* We used an edge-computing device for the control of our robots, which is essentially a computer server. The computer (Intel Core i9-10900 K CPU @ 3.70 GHz, 32 G RAM) consists of 20 cores and is installed with Linux Ubuntu 18.04 LTS and robot operating system (ROS).
2) *Environment Setup:* In the indoor real-world experiments, we construct two task scenarios in the laboratory, which resemble the simulated scenarios to reduce the gap between simulation and the real world, improving the deployment accuracy of the trained model in the real-world robots. In Fig. 8, the antiglare floor represents restricted areas. In addition, protective nets are placed around the area to prevent observer robots from flying out accidentally. The whole scene maps to 6×6 square metres.
3) *Communication Setup:* The communication in our indoor scenario is ensured by a high-speed, low-latency Wi-Fi router. The trained model, receiver, and emitter replay buffer are connected to the robots' local network

[2][Online]. Available: https://github.com/NXROBO/spark

[3][Online]. Available: https://store.dji.com/shop/tello-series?set_region=US&from=store-nav

[4][Online]. Available: https://www.linkedin.com/company/robotplusplus/

[5][Online]. Available: https://www.softbankrobotics.com/emea/en/pepper

[6][Online]. Available: http://www.unitree.cc/e/action/ShowInfo.php?classid=6&id=1

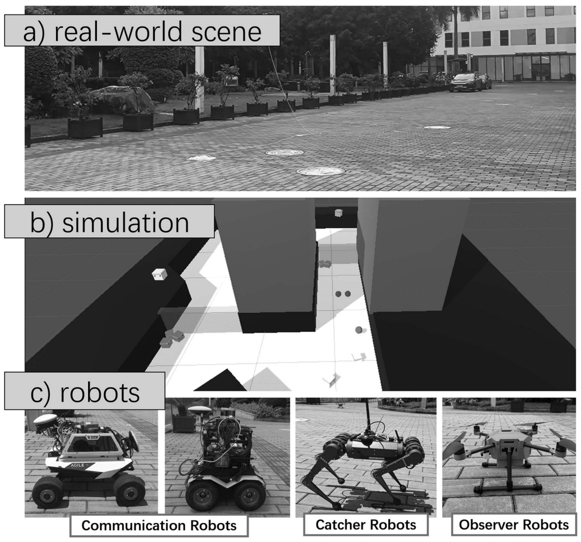


Fig. 9. Outdoor real-world demonstration setup. (a) Whole demo area inside the Chinese University of Hong Kong, Shenzhen. (b) Simulation built using a 3-D map of the University. (c) Type of robots used in the experiments.

by utilizing ROS. To ensure consistency between robots, commands are sent in 20 ms and executed in 100 ms.

3) Outdoor Real-World Setup: The real-world demonstration was carried out at the Chinese University of Hong Kong, Shenzhen. To this end, we first make a simulated area similar to the University environment. We demonstrate the outdoor catching scenario to show that our method is robust enough to deal with outdoor turbulences with current technological conditions, such as localization inaccuracy, control limitations, etc. Compared with the indoor experiment, we chose a much larger area (60×60 m) to test our method's integration with the outdoor environment. We use the same method for intrateam and cross team cooperation. However, to adapt to the outdoor scale, we followed prior work [77] and built open-source MIT Mini-cheetah quadruped robots as the catching robots, which are relatively faster compared with Spark robots and more suitable on bumping terrains. Moreover, we use an Amov quadcopter to counter wind turbulences and we also installed a longer antenna for stabler communication performance. Fig. 9 shows our robots used in the outdoor environment . The robot is trained using a 3-D campus map of the Chinese University of Hong Kong, Shenzhen, which provides information on obstacles in a 3-D environment. MR500 and Amov R300[7] are unmanned electronic vehicles capable of traveling on bumpy terrains for a long time. Three different MIT cheetah robots are used, with two as catcher robots and one as runner robot.

B. Metrics for the Evaluation

We use two main metrics for our later experiments. The first is the total reward generated by the HMRS. In the following sections, the total reward is described by the addition of (12)–(14) and is used to evaluate the performance of different configurations. The other metric is the definition of successful catching behavior, which is different in simulated and real-world experiments. In the simulation, we test whether each step is successful, and we utilize (12) to count whether the catcher robots are close enough to the runner robots' overall running steps, with a team of the same opponents. For real-world experiments, we focus on whether each episode is successful. Therefore it is required that each step is successful for some N consecutive times before the episode is considered successful. After the algorithms have converged, the catcher robots and the runner robots can enter an equilibrium in which either catcher robots or obstacles surround the runner robots. More precisely, after entering the equilibrium for 20 steps (around 2 s), we count the scenario as successful in the real world. Mathematically, the success of the catcher robots is ensured by two conditions. 1) All runner robots must move no more than an arbitrary distance L_1 in 2 s, $\forall \mathcal{R}^t \in \tilde{\mathcal{R}}^t, \mathcal{R}^t - \mathcal{R}^{t-20} \leq L_1$. 2) For all catcher robots, they need to surround at least one runner robot. $\forall \mathcal{Q}^t \in \tilde{\mathcal{Q}}^t, \exists \mathcal{R}^t \in \tilde{\mathcal{U}}^t, \text{dist}(\mathcal{R}, \mathcal{Q}) \leq L_2$, where L_2 is an arbitrary distance. In our real experiment, L_1 and L_2 are set as 0.3 and 0.5 m.

C. Simulated Experiments

1) Experiments on Different Mechanisms: As previously stated in the method section, we build our baseline system through systematic experiments combining different mechanisms of DRL. More specifically, self-play and curriculum learning are the main mechanisms employed in our development. As shown in Fig. 6, we first tested the performance of noncooperative and cooperative behavior by using single-agent and multiagent setups (i.e., POCA-same-beha/POCA-diff-beha) on a team of catcher robots. After this, we added two runner robots as the team of opponents for asymmetric self-play to stimulate the intelligence of the quadruped robot team. Lastly, we continue with different stages of training observer robots and communication robots, respectively. Crossing the different stages, the whole structure of the learning process could be understood as a curriculum enabling different teams of heterogeneous robots to act cooperatively to achieve the common goal specified by $\mathcal{L}_R$.

a) General Learning Performance: We train our system based on the curriculum learning principle and design three stages in our learning process. In the first stage, we train the catcher robots team together with the runner robots team. While keeping the observer robots team and the communication robots team away, we also introduce fewer nonstationary elements in our heterogenous system, which helps to reduce the variances of the rewards received during the interaction [78]. After that, we then bring the observer robots team and the communication robots team, respectively, into our system to include the relevance of heterogeneity. The result shows that our system is able to converge to a stable strategy after three stages of learning while keeping some level of intelligence. Fig. 10 shows the learning curves of different stages, namely, the green, blue, and orange curves. Compared with the strategy without curriculum strategy, our method has a relatively higher total reward and less variance. However, we need to note that higher reward does not always lead to better performance when the system is highly

[7][Online]. Available: https://amovrobotlab.com/

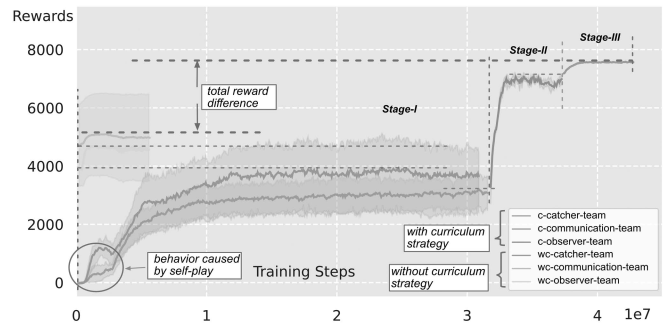


Fig. 10. Illustration of the learning curves of our system. Two groups of curves are compared in this figure. One group, which consists of pink, yellow, and green curves, is trained without the curriculum strategy. The other group, which consists of green, blue, and red curves, is trained with the curriculum strategy. A total reward difference can be found between the two groups.

dynamic. This is due to that a chaotic learning process affects the team of runner robots as well. If the runner robots are not able to run intelligently, they are more likely to be caught by the catcher robots, resulting to have a higher reward for the catcher robots alone. To have a fair comparison between these two groups, we conduct a study using the policies learned from both groups. We compare the catcher robot policies against the policy of the runner robots learned in the noncurriculum group using the metrics mentioned in Section IV-B. For total of 200 runs, we received on average ($M = 261.21$) steps for the group that uses the curriculum strategy vs ($M = 257.1$) steps for the group that does not use the curriculum strategy. To further illustrate the interaction between the runner robots team and our heterogeneous catching system more clearly, we sample a few interactions from two different groups and evaluate the behavior. Fig. 11 illustrates three sampled behaviors of the curriculum-based strategy. It shows that our system is able to catch the runner robots after using different strategies. The details of Fig. 11 will be explained in Section IV-A.c.

b) Single-agent Verus Multiagent: We aim to make our teams of heterogeneous robots act cooperatively to catch the runner robots. To make sure of that, we experimented with two sets of configurations. One of the configurations is named same behavior POCA (same-beha-POCA), which means that each team of the same type of robots shares a same policy, allowing the multiagent algorithm to redistribute the reward to each robot. The other configuration is named different behavior POCA (diff-beha-POCA), which means each agent acts as a single-agent team and gets all the rewards. Except for the above-mentioned difference, the other configurations remain the same. We hypothesize that the former algorithm could provide a better performance in our catching scenarios due to cooperative behaviors. Fig. 12 illustrates the differences between the two groups. On the left side, both policy networks are trained with 3.16×10^7 steps, while on the right side, the diff-beha-POCA group has run $3 \times 3.16 \times 10^7$ steps. This is because the single-agent (diff-beha-POCA) policy needs to observe three times more data to be comparable with the multiagent (same-beha-POCA) policy, as the multiagent policy gets data from three catcher robots simultaneously. After balancing the total amount of data, we can observe that when each catcher robot acts individually,

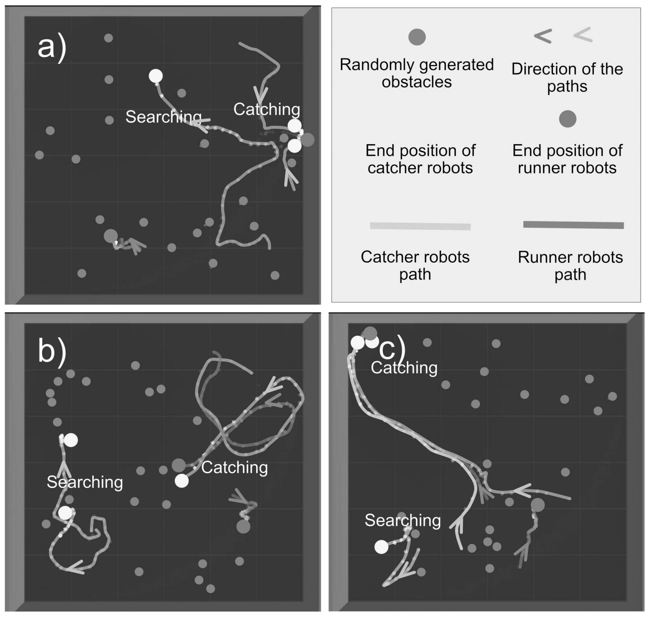


Fig. 11. Illustration of three different scenarios emerged from learned strategy. In these time-lapse figures, the yellow points represent the catcher robots team, and the blue points represent the runner robots team. The red circles are obstacles that both runner robots and catcher robots need to avoid during the overall process. The endpoints of the important trajectories are marked with bright circles. From (a) to (c), each figure represents a scenario where robots need to make intelligent decisions.

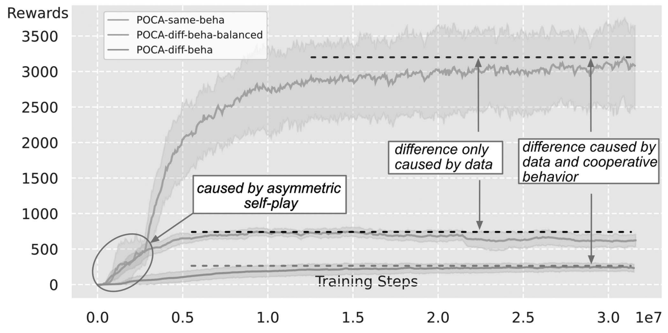


Fig. 12. Comparison of single-robot noncooperative and multirobot cooperative cases. The green line shows the multirobot cooperative case, in which each type of robots are assigned to a team. The orange and purple lines represent the cases in which each robot is a single-robot team. Among these two lines, the orange line shows the situation when the single-robot learns all by itself. Whilst, the purple line shows the situation when the single-robot gets three times more data. This is due to that in the multirobot case, the POCA algorithm gets data from three catcher robots.

the catcher robots team receives a significantly less amount of reward than when they act cooperatively.

c) Asymmetric Self-Play: For each level of abstraction mentioned in Section IV-A, we repeated the experiments and compared their performance against runner robots by simultaneously training asymmetric self-play agents. This asymmetric self-play agent is utilized to stimulate the intelligence of the heterogeneous system. To ensure the learning performance of the catcher robots, each self-play agent is trained with 9×10^5 steps, and each catcher robot is trained with approximately 1.054×10^7 steps. This mechanism makes sure that the runner

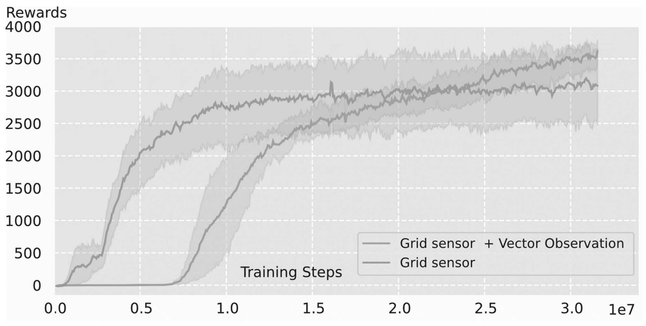


Fig. 13. Comparison of two different observations. The pink line indicates the situation where the team of catcher robots uses grid sensors as observations. The green line indicates the situation in which the team of catcher robots uses grid sensors together with the location of the runner robots as observations. We can see that when the location information is given to the catcher robot team, they are able to get more rewards in the environment, indicating that the catcher robots can suspend the runner robots more.

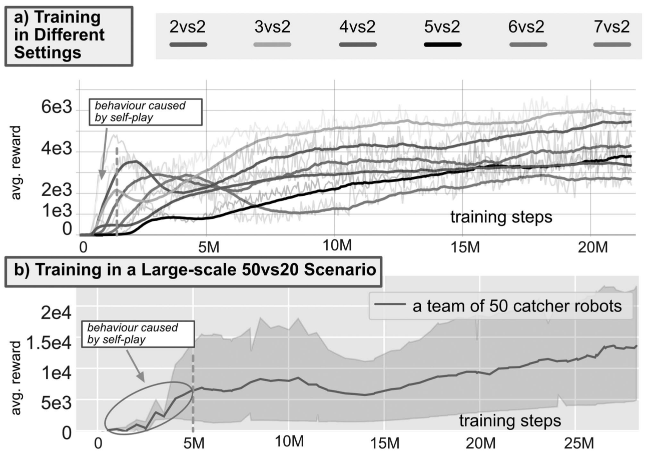


Fig. 14. Evaluation of large-scale catcher robot teams. (a) Training processes of catcher robot teams of different sizes. (b) Results of training in a large-scale 50vs20 scenario.

can learn a decent strategy of avoiding the catcher robot team in the first 9×10^5 steps. After training the team of catcher robots, we noticed a few interesting patterns emerge during the process. As previously mentioned, Fig. 11 illustrates three different cases where each robot needs to make intelligent decisions based on their observations (other robots are omitted in the figure). We need to note that each robot only has local observations, so they cannot see other robots far away from it. The critical trajectories' end points are marked with yellow and blue circles. Yellow circles are associated with catcher robots, and blue circles indicate runner robots. In all figures, small red dots represent obstacles that all robots need to avoid during the process. Fig. 11(a) illustrates a situation where a catcher robot (blue circles) observes that a runner robot is suspended by two of the catcher robots. As a consequence, it leaves them and searches for the other runner robot. Fig. 11(b) shows another catching scenario, where a runner robot used two swings to run away from the catcher robot, and the catcher robot was still able to follow the runner robot closely. Fig. 11(c) illustrates another scenario where two catcher robots cooperatively follow a runner robot. These three cases indicate that some amount of asymmetric self-play could help with the decision-making process of heterogeneous multiagent systems.

2) *Ablation Study*

a) Location: We experimented with the influence of the zero-range sensing [79], which is the location and the pose of the moving runner robots. Each experiment was conducted six times to determine the algorithm's mean and variance to evaluate the performance more accurately. The results of with and without location information can be observed in Fig. 13. The red area in the figure indicates turbulence caused by the opposite learning robots. We can observe from the figure that the location information added extra stability to the system, indicating that zero-range sensing helps with the catching scenarios. This further suggests that if the observer robots can provide location information to catcher robots, the overall stability could be improved.

b) Influence of Reward Functions of Communication Robots: As described in Section III-C5, the reward functions for communication robots has two components. One reward function requires the team of communication robots to be close to catcher robots and quadcopter robots, and we also added a sparse reward to indicate that the team of communication robots needs to make sure that other robots stay within its coverage range. When only the first component is present, the team of communication robots only tries to stay in the middle of the team. When the coverage reward is added, the robots' relative location to other robots becomes a bit arbitrary, but they also start to care about ensuring most of the robots stay in the range. The results show that the second approach resulted in a 23% increase in the time to stay within the 45 m communication range, reaching 86% of the total time in the simulated experiment.

3) *Train With a Larger-Scale Catcher Robots Team:*

To test our method's learning performance with a large-scale catcher robots team, we gradually increase the number of robots in the catcher team and test whether our approach can still learn in different scenarios. Our results in Fig. 14(a) show that we can train a catcher robots team of 2 to 7 robots by utilizing the residual attention mechanism in the MA-POCA algorithm. In addition, to have a boundary test, we train a 50vs20 scenario, where a team of 50 catcher robots needs to catch a team of 20 runner robots in an area nine times larger than the scenario tested in Fig. 14(a) (i.e., 150×150). We trained the 50vs20 scenario five times, and the results in Fig. 14(b) show that our method can handle a large-scale catcher robot team during training. However, we need to point out that training the 50vs20 scenario takes around 20 GB VRAM and 72 h; consequently, we can only train it with an Nvidia 3090 GPU with Unity ML-Agents Toolkit. Another thing we noticed in Fig. 14(b) is that the results have a more considerable relative variance compared with scenarios with fewer catcher robots (see Fig. 13). This may be caused by a large number of catcher robots or the large strategic space of this scenario.

4) *Bridging Study:*

In this section, we conduct a bridging study to compare our method's performance with a method called S2M2 [1] in our catching scenario. To the best of our knowledge, there is no method dedicated to solving the intelligent catching problem using heterogeneous teams of robots. The works mentioned in Table I mostly used state-machine or planning-based algorithms, resulting in possible limitations on robustness. To name a few, the study conducted in [8] used a procedure with state identification and a state-based rapidly exploring random tree star (RRT*) algorithm. Another work [11] used statistical adaptive methods. These methods are not general enough to be compared. However, with some modifications, a recent state-of-the-art algorithm developed for the reach-and-avoid behavior of multirobot teams could be compared. The S2M2 is an optimization and priority-based search (PBS)-based algorithm that enables the reaching behavior of teams of robots while being able to avoid all the obstacles and each other during the process. The PBS family is claimed to be better in effectively coordinating agents than the conflict-based algorithms [1], such as conflict-based search. We compare our method with this algorithm based on a few reasons. The main reason is that this algorithm is claimed to be a state-of-the-art algorithm that deals with catching-like (reach-and-avoid, as described in [1]) tasks. Also, this algorithm is a relatively new algorithm that works in a continuous multirobot setup with obstacle avoidance functionality and has state-of-the-art performance. We used the python library of Gurobi 9.0.1 as the original S2M2 code[8] for a fair comparison.

A few customizations are applied to this algorithm to make it work in our HMRS catching scenario. The first is that the goals of the S2M2 are updated constantly because the intelligent runner robots are always moving. The second is that our catching team is an HMRS that consists of three teams of robots. For each team of robots, we need to have one S2M2. That means we need to have three S2M2 implementations for the whole catching team. Third, the S2M2 needs to get geometric information of all objects in the environment as input and this means that we need to assume that all the positions of obstacles and agents are known to the whole system. We name the method that considers the previous three modifications as S2M2-catching. The goals are updated using a K-nearest neighbors (KNN) algorithm with our case $k = 1$ in this study, meaning that each catcher robot always aims for the closest runner robot. Algorithm 1 summarizes the differences between the original S2M2 and our S2M2-catching. The blue area indicates where we assign the goals to agents, and the red area shows how we continuously select paths for agents.

With these modifications, we compare our method with S2M2 in a smaller environment that contains fewer robots. The main reason is that the S2M2-catching needed a lot of time to compute in a large environment. We conducted a pilot study and found that it takes around eight times more time to compute in a simulated environment with more robots and obstacles, making experiments in a larger environment unperformable. As a consequence, we evaluate two algorithms in a smaller environment (see Fig. 10) using the methodology widely employed in machine behavior analysis [80]. In total, 2400 runs with, respectively, 100, 500, 1000, 1500, 2000, and 2500 steps are tried to compare the performance of S2M2-catching with our learning-based method in the simulated environment. We compare the total times of successful catching behavior by utilizing the metrics of the simulated experiments introduced in Section IV-B and count in how many steps the catcher robots are close enough to the runner robots. We present the results over six different running steps, assuming that the whole environment is fully observable. After receiving the results, we first perform a two-tailed t-test to check whether there is a statistical difference between the two groups in the final 2500 steps conditions. We found that the S2M2-catching algorithm ($M = 873.15$, $SD = 320.28$) and our learning-based method ($M = 1075.19$, $SD = 255.38$), $t(398) = 6.95, p < .001$, are significantly different. In addition, the results of both groups passed the normality test when the step numbers are bigger than 1000. Also, to further analyze the trend, instead of using multivariate analysis of variances or a general linear model, we assume a Gaussian-like nonlinear transformation to the input at each step and perform a Gaussian process regression (GPR) via maximum likelihood estimation. Fig. 15 shows the estimated means and variances of each group. Demonstrating the trend, we also make predictions based on our learned GPR model, which can be seen in the figure. The statistics of the two algorithms are also explainable intuitively. In the beginning, the optimization-based method can find nearly

[8]We used the original S2M2 code from https://github.com/jkchengh/s2m2

Algorithm 1: S2M2 [1] With Our Modifications.

```
Input:
1:  $\mathcal{H}$: a set of heterogeneous teams of robots, with each team
    $h \in \mathcal{H}$ containing $N_h$ robots;
2:  $\mathcal{R}_h$: a set of robots associated with a team $h$, which
    contains start positions $\mathbf{s}$, goals $\mathbf{b}$, time requirements $T$
    and positions of obstacles $\mathbf{o}$;
3:  $\mathcal{U}$: a set of runner robots that contains runner robots'
    positions.
4:  $\mathcal{Q}$: a problem descriptor using a mixed integer linear
    programming (MILP) framework.
Output:
5:  $\mathcal{P}$: a path that describes how should each robot move.
6:  Add time constraints $T$ to $\mathcal{Q}$
7:  for $h \in \mathcal{H}$ do
8:      Update goal $\mathbf{b}$ using KNN and $\mathcal{U}$.
9:      for $t = 1$ to $T$ do
10:         Add goals constraints $\mathbf{b}$ to $\mathcal{Q}$
11:         for $r \in \mathcal{R}_h$ do
12:             Add start positions $\mathbf{s}$ as constraints to $\mathcal{Q}$
13:             Add obstacles $\mathbf{b}$ as constraints to $\mathcal{Q}$
14:             Get $\mathcal{P}$ from $\mathcal{Q}$ using a MILP solver.
15:         end for
16:         for $r \in \mathcal{R}_h$ do
17:             Solve conflicts of robots $\mathcal{P}$ using PBS.
18:         end for
19:     end for
20:     for $r \in \mathcal{R}_h$ do
21:         Update $\mathcal{P}_r$ by cutting at time $t$.
22:     end for
23: end for
```

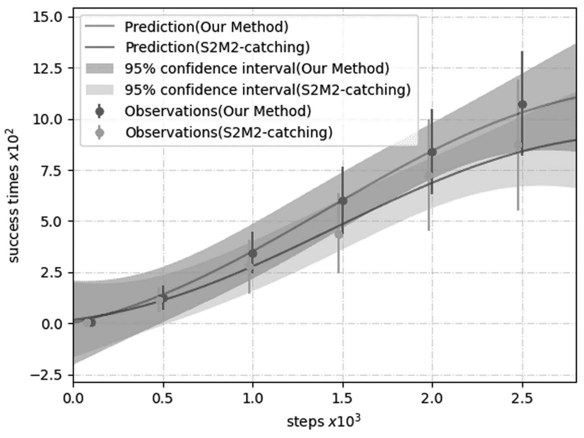


Fig. 15. Fitted Gaussian process regression models of the S2M2-catching algorithm and our method. Our method had an overall better performance. The S2M2-catching performed better when the number of steps are low (less than 250).

optimal paths to the runner robots, and they are the first to touch the runner robots. However, because the robots being tracked are intelligent and can make detours or circles around points, the optimization-based algorithms can quickly lose them. Instead, as our methods are better at intelligent strategies, such as encirclement or containment, they perform better in longer runs.

D. Real-World Indoor Experiments

With the results presented, we need to emphasize that the S2M2 deals with general continuous multirobot reach-and-avoid scenarios, and our results do not mean that it does not perform well in other tasks (see. [1]). Nevertheless, based on these results, we also point out a few advantages of our method in solving the intelligent catching problem, 1) the self-play and the DRL-based methods enable the agents to consider long-term strategy, making the catching more effective in the long term. 2) The assumption of always getting the position of all the obstacles in the system is strong. Instead, the representation used in our method, (e.g., grid sensors) can be easily transformed from other more realistic sensors (i.e., camera and Lidar).

The real-world experiments are conducted to show that our framework could be used in a scenario where different real robots can collaborate intelligently to complete a catching task. Concretely, this demonstration is divided into two experiments, mapping to two different hypotheses. The first hypothesis is that the control of the system can be used in reality as a whole. This is one of the most important hypotheses. An additional hypothesis is that randomly generated obstacles will increase the robustness needs of the system. We measure the degree of completion by setting the completion of the task in the simulation to be 100% and compare the completion rate of the system in the real world. For the second hypothesis, we measure it by setting up a moving obstacle in the real environment. We measure both the collision rate with the uniformly moving obstacles in the simulation, and the collision rate of the moving obstacles in reality under the assumption that moving obstacles have the

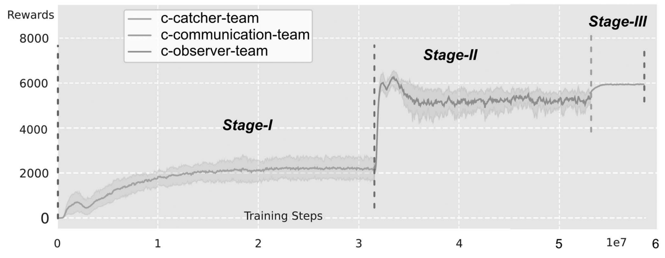


Fig. 16. Illustration of the learning curves of the trained model in a real-world experiment. The total reward increases over three stages of learning.

same movement speed as the robot. To make sure that our model could be used in reality, we first train a model in simulation. In this simulation, the characteristics of each robot are the same as the properties of the real-world robot, meaning that the trained model is similar to reality. Fig. 16 shows the learning of simulation in the real-world experiment. One property we can observe is that the trend of the learning curve is similar to the learning curve of the previous simulated experiments (as can be observed by comparing Figs. 16 and 10).

1) Success Rate and Number of Collisions: To quantify the system's performance, we utilize the definition of success in real-world experiments mentioned in Section IV-B. As previously described, we have eight robots in this system, including three catcher robots, two observer robots, a communication robot, and two runner robots. Three of the spark robots are set up as catcher robots. Two robots, a Pepper robot, and a green spark robot are utilized as runner robots. In addition, a Leikago robot is used as a moving obstacle to test the performance of the real-world system. We randomly initialize the scenario 18 times and measure the success rate of the real-world system and compare it with the simulated scenarios. Fig. 18 counts the successful cases in real experiments. Each experiment contains two possible successful cases so in total we have 54 possible successful cases. Theoretically, the real-world scenario should have more failures than the simulation due to the imperfect controllers. However, statistically speaking, the difference is not large enough. (Two sampled Z-test of proportions indicate that we cannot tell there is a significant difference $(p = 0.18 > .05)$).

a) Avoidance of moving obstacles and number of minor collisions: The real-world experiments show that our system is sufficiently robust against moving obstacles after training the whole system through domain randomization. To this end, in the real-world experiment, we used a quadruped robot Laikago as a moving obstacle. It approaches different robots during the interaction, testing whether catcher robots are able to avoid it. The obstacles' positions are reflected in the simulation in real time. We count the number of approaches and the number of successful avoidance during the interaction. Fig. 18 shows that in a total of 14 approaching trials, 13 of them are avoided by the system. For one time, this caused a minor collision. A minor collision means that two robots have a minor impact contact, not interrupting the whole process. In total, two minor collisions happened in the real-world experiments.

2) Demonstration of Intelligence: To further illustrate intelligence, Fig. 17 shows all the ending scenes of the 18 real-world

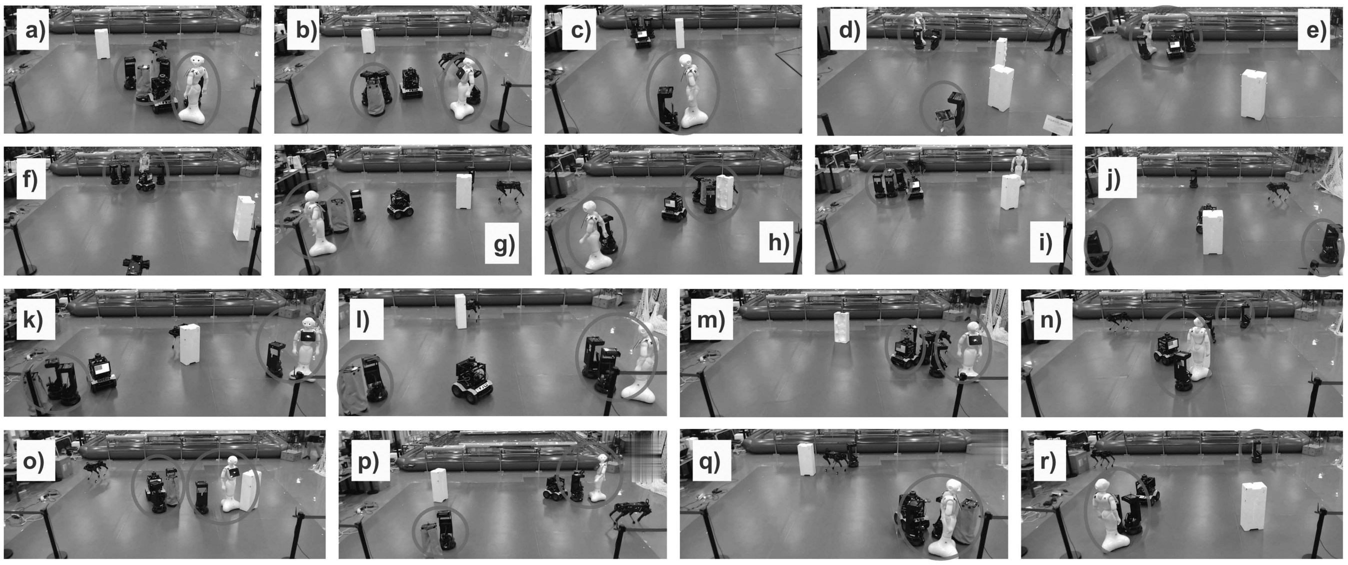


Fig. 17. A collection of figures depicting the end moments of all real-world scenarios. Red circles show the final equilibrium between the catcher robots and the runner robots. Another thing that could be observed in this figure is the location of the communication robot. It always stays relatively in the middle of the catching team.

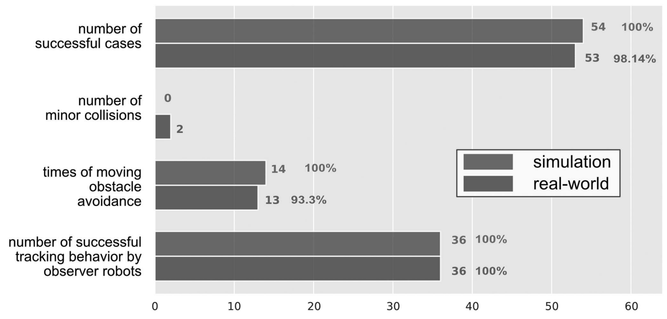


Fig. 18. Comparison of the simulation and real-world experiments. For the measurement of successful cases, we conducted 18 times repeated experiments. In total we have 18 × 3 robots = 54 possible successful cases.

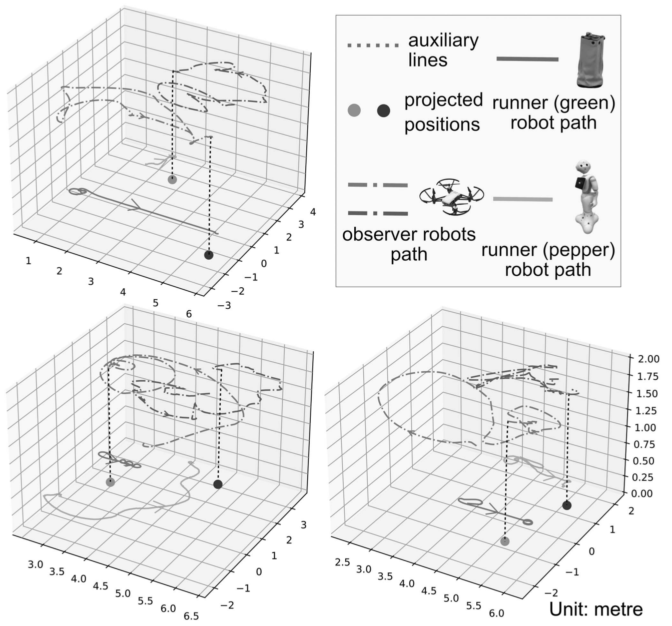


Fig. 19. Examples of the paths generated by observer robots and the runner robots. During the experiments, the observer robots stayed close to the runner robots. The projected final positions of the observer robots are also close to the final positions of the runner robots, showing that the catching behavior of the observer robots lasted until the final moment.

experiments. From this figure, we can also see that the team of catcher robots catches both runner robots through different means. In many cases, the catcher robots utilized the edge of the area to block the paths of the runner robots [e.g., in Fig. 17(c) and (k)]. However, in some cases, the obstacle is also used to block the path of the runner robots. [e.g., in Fig. 17(h) and (n)]. This shows that the catcher robots team developed intelligent strategies in our experiments.

3) Additional Statistical Metrics: To further quantify the real-world experiments, we illustrate two more metrics regarding the performance of the observer robots and the communication robot. The first metric is to evaluate whether the observer robots can follow the runner robots closely. This metric is quantified by counting whether observer robots can stay in a 3-m range of the runner robots for at least $4/5$ total interaction time. In all 18 times experiments, the behavior of the observer robots satisfies this condition (see Fig. 18), demonstrating that the observer robots are able to provide functionalities needed by the system. To exemplify a few, Fig. 19 shows the paths of runner robots and the observer robots. We can see that the observer robots stayed close to the runner robots during the experiments. The final projected points of observer robots in experiments are also very close to the runner robots.

The training process of the communication robot can already be observed as a function of reward, illustrated in Fig. 16. However, as a demonstration, we would like to show several examples that the communication robot can generate a path that

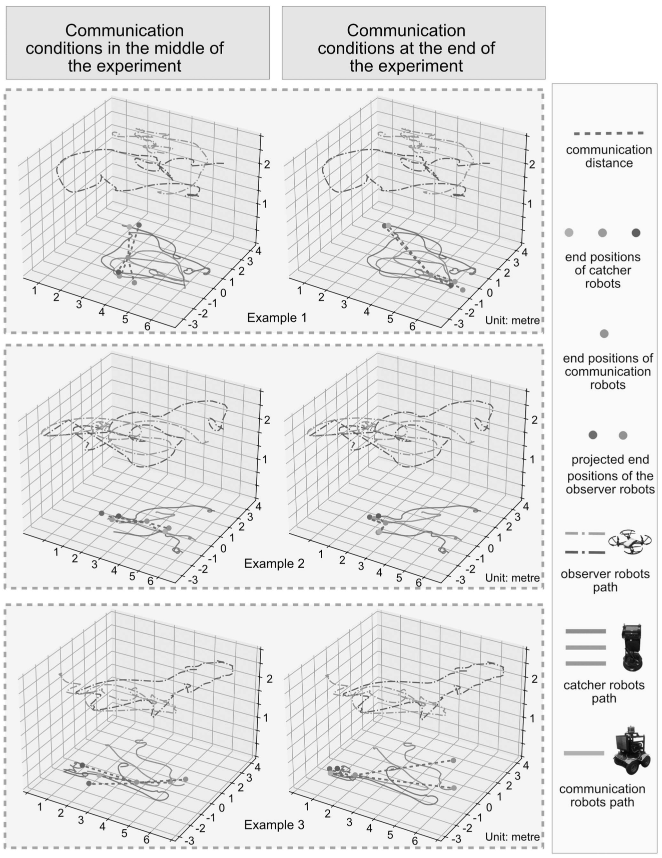


Fig. 20. Examples of the paths generated by the whole catching team. Three examples are shown to illustrate that the communication robot tends to keep the distances short and can stay in the range for joint communication in the middle of and at the end of the experiments.

tries to stay within the range for joint communication of the catching team. Fig. 20 shows three examples of the generated paths. For each example, we marked the projected position of each robot in the middle of the experiment and at the end of the experiment. Thereon, we use auxiliary lines to show the distances between the communication robot and other robots. Because the communication robots team follows the criteria described in (14), it always tries to minimize the overall communication distance.

E. Real-World Outdoor Experiments

As stated in the setup Section IV-A, we also conducted experiments in the outdoor environment. We would like to know whether our method, together with other common technologies, could work well with real-world constraints. Fig. 21 illustrates the paths of robots observed by the other robots during the catching scenario. We can observe from the figure that even with noises, the catcher robots are still able to surround the runner robot. The results show that our method demonstrated robustness during the tests.

1) Implementation Details: Fig. 22 shows the details of the system. For the localization [see Fig. 22(a)], we use the Nooploop ultrawideband (UWB) P-A solution[9] together with visual-inertial odometry to enhance the general localization and orientation detection of the robots in the outdoor environment. Four to six UWB anchors [see Fig. 22(a)] are placed in the 60×60-m environment. Each robot is equipped with two UWB modules that can average out location and orientation noises for better results. In addition, it also provides additional information to correct the heading position of the robots. For safety concerns, in the outdoor environment, we use a larger detection range (virtual collision bubble, see Fig. 22(b), for each robot to ensure

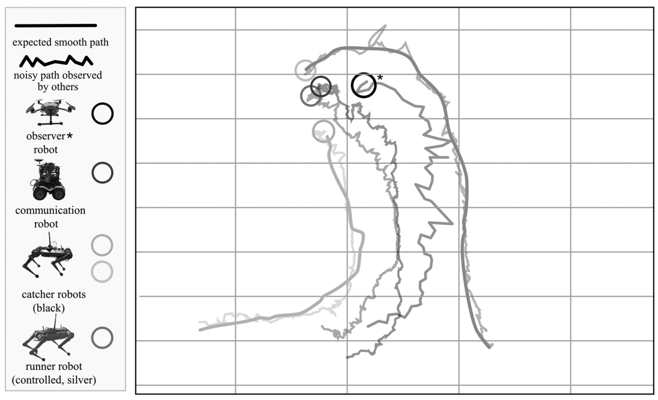


Fig. 21. Paths generated by different robots in the outdoor environment. Although the robots are affected by noises and do not generate the expected smooth paths, our system can still finish the task. * The noisy path of the quadcopter is extracted post-experiment from video, and the noisy paths of other robots are recorded by the UWB device during the experiment.

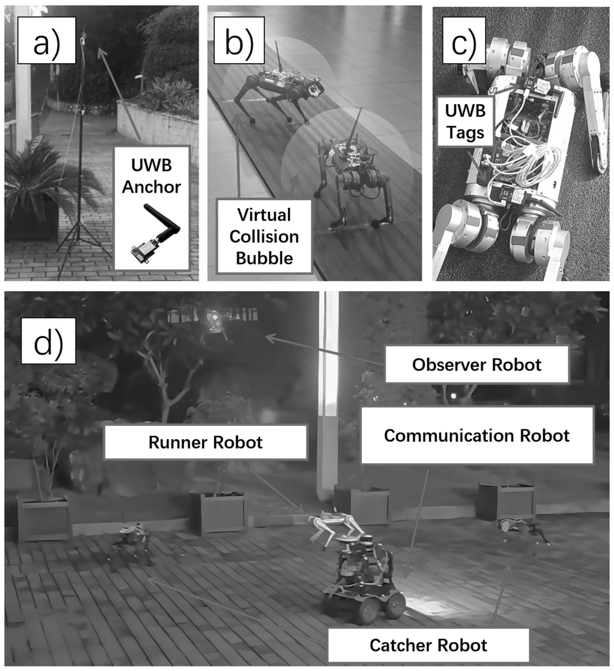


Fig. 22. Illustration of important details of the system. (a) UWB-based localization system. (b) Comparatively larger virtual collision bubble around the robot to avoid catastrophic failure for safety issues. (c) One of the ending situations of the experiment.

[9][Online]. Available: https://www.nooploop.com/en/

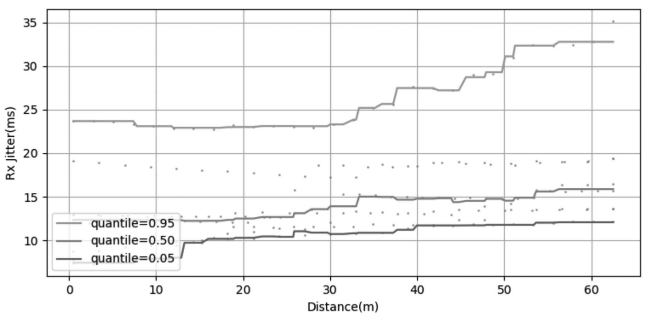


Fig. 23. Illustration of instant communication jitters. The red, green, and yellow lines show estimated 5%, 50%, and 95% quantiles of instant communication jitters as a function of distance.

that no dangerous collision will happen when robots have high speed.

2) Discussion of Possible Limitations in a Larger Area

a) Communication Range: Based on the assumption, we set the communication boundary to be 45 m in an outdoor environment. We collect statistical results from the communication robots to show the relationship between our assumptions and real-world communication situations. We use telecommunication customer premises equipment (CPE) to diversify the bandwidth needed by the robots. To estimate the bandwidth model, we consider a sampling process over two metrics 1) Rx bandwidth 2) Tx bandwidth in polar coordinates, in which we first perform a uniform angular sampling and a uniform sampling on the radius. In this way, we get data samples from eight directions. Due to that integral in polar space results in the length of the radius, we correct the sampling by reweighting the samples. Then we resampled the dataset and used the Radial basis function (with a Gaussian smoothing kernel) to estimate the speed over the whole space. Fig. 5 shows the estimated Rx bandwidth over the area. In this figure, we can observe that, though the signal on the communication robot is skewed, it can still have a larger than 5 Mbps speed within a 45-m range, showing that our assumption is correct.

b) Instant Communication Jitter: Another metric we are interested in is instant jitter. We would like to know whether distance can affect real-time communication. To this end, we used the same method to measure the delay between the communication and catcher robots. The distance does not affect communication jitter much, despite dramatic bandwidth loss, as shown in Fig. 5. Fig. 23 illustrates our results on instant jitter. We collect data in eight directions to estimate the instant jitter as a function of distance. After the data collection, we use an ensemble method, a histogram-based gradient boosting regression tree, as a general regression method to estimate the 5% quantile and 95% quantile of the communication delay. It can be observed that the estimated mean value is smaller than 20 ms at 60 m.

V. Conclusion

In this work, we built a system to face the growing complexity, heterogeneity, and robustness needs of adversarial catching in future security and rescue scenarios. To handle the increasing number of constraints, we used deep multiagent RL together with two mechanisms, namely, asymmetric self-play and curriculum learning, to handle the increasing heterogeneity and number of agents in HMRS. We also conducted a bridging study to evaluate the performance of a state-of-the-art multiagent path finding approach called S2M2 with our approach in HMRS catching scenarios. Combining different learning techniques, we were able to learn faster during the training and to achieve intrateam and cross-team cooperation among robots in the catching team. By merging our model with a low-level control algorithm, our system was able to be easily implemented in the real world, resulting in a working real-world HMRS that can perform catching tasks under realistic constraints. One of the future applications after this is to develop a larger system in an environment with more heterogeneity and uncertainty involved for security and rescue HMRS applications.

Acknowledgment

This research follows the United Nations' sustainable development goals of AI for civil purposes and the Chinese new generation of artificial intelligence ("AI") ethics code. Our team is committed to the safe, responsible, and transparent use of our robots for the benefit of society.

References

[1] J. Chen, J. Li, C. Fan, and B. C. Williams, "Scalable and safe multi-agent motion planning with nonlinear dynamics and bounded disturbances," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 13, pp. 11237–11245.

[2] A. Prorok, M. A. Hsieh, and V. Kumar, "The impact of diversity on optimal control policies for heterogeneous robot swarms," IEEE Trans. Robot., vol. 33, no. 2, pp. 346–358, Apr. 2017.

[3] A. Prorok, V. Kumar, B. Sadler, and G. Sukhatme, "Introduction to the special section on resilience in networked robotic systems," IEEE Trans. Robot., vol. 38, no. 1, pp. 2–4, Feb. 2022.

[4] B. Browning, J. Bruce, M. Bowling, and M. Veloso, "STP: Skills, tactics, and plays for multi-robot control in adversarial environments," *Proc. Inst. Mech. Eng., Part I: J. Syst. Control Eng.*, vol. 219, no. 1, pp. 33–52, 2005.

[5] M. Dorigo et al., "Swarmanoid: A novel concept for the study of heterogeneous robotic swarms," *IEEE Robot. Automat. Mag.*, vol. 20, no. 4, pp. 60–71, Dec. 2013.

[6] M. Aggravi, A. A. S. Elsherif, P. R. Giordano, and C. Pacchierotti, "Haptic-enabled decentralized control of a heterogeneous human-robot team for search and rescue in partially-known environments," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4843–4850, Jul. 2021.

[7] M. J. Schuster et al., "The arches space-analogue demonstration mission: Towards heterogeneous teams of autonomous robots for collaborative scientific sampling in planetary exploration," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 5315–5322, Oct. 2020.

[8] E. Narváez, A. A. Ravankar, A. Ravankar, T. Emaru, and Y. Kobayashi, "Autonomous VTOL-UAV docking system for heterogeneous multirobot team," *IEEE Trans. Instrum. Meas.*, vol. 70, 2020, Art. no. 5500718.

[9] Q. B. Diep, I. Zelinka, and R. Senkerik, "An algorithm for swarm robot to avoid multiple dynamic obstacles and to catch the moving target," in *Proc. Int. Conf. Artif. Intell. Soft Comput.*, 2019, pp. 666–675.

[10] S. McCammon et al., "Ocean front detection and tracking using a team of heterogeneous marine vehicles," *J. Field Robot.*, vol. 38, no. 6, pp. 854–881, 2021.

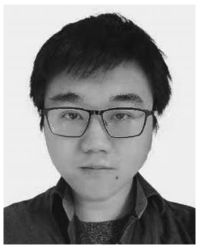
[11] H. L. Kwa, G. Tokić, R. Bouffanais, and D. K. Yue, "Heterogeneous swarms for maritime dynamic target search and tracking," in *Proc. IEEE Glob. Oceans: Singapore–US Gulf Coast*, 2020, pp. 1–8.

[12] L. Cohen, T. Uras, T. S. Kumar, and S. Koenig, "Optimal and bounded-suboptimal multi-agent motion planning," in *Proc. 12th Annu. Symp. Combinatorial Search*, vol. 10, no. 1, pp. 44–51, 2019.

[13] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative heterogeneous multi-robot systems: A survey," *ACM Comput. Surv.*, vol. 52, no. 2, 2019, Art. no. 29.

[14] D. Silver et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[15] M. Krizmancic, B. Arbanas, T. Petrovic, F. Petric, and S. Bogdan, "Cooperative aerial-ground multi-robot system for automated construction tasks," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 798–805, Apr. 2020.
[16] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[17] O. Vinyals et al., "Grandmaster level in starcraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.
[18] 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*, vol. 16, no. 7, 2016, Art. no. 1018.
[19] T. Abukhalil, M. Patil, S. Patel, and T. Sobh, "Coordinating a heterogeneous robot swarm using robot utility-based task assignment (RUTA)," in *Proc. IEEE 14th Int. Workshop Adv. Motion Control*, 2016, pp. 57–62.
[20] J. P. Queralta and T. Westerlund, "Blockchain-powered collaboration in heterogeneous swarms of robots," 2019, *arXiv:1912.01711*.
[21] C. Liu and A. Kroll, "Performance impact of mutation operators of a subpopulation-based genetic algorithm for multi-robot task allocation problems," *SpringerPlus*, vol. 5, no. 1, 2016, Art. no. 1361.
[22] H. Wang and M. Rubenstein, "Shape formation in homogeneous swarms using local task swapping," *IEEE Trans. Robot.*, vol. 36, no. 3, pp. 597–612, Jun. 2020.
[23] P. Das and P. K. Jena, "Multi-robot path planning using improved particle swarm optimization algorithm through novel evolutionary operators," *Appl. Soft Comput.*, vol. 92, 2020, Art. no. 106312.
[24] X. Huang, M. Sun, H. Zhou, and S. Liu, "A multi-robot coverage path planning algorithm for the environment with multiple land cover types," *IEEE Access*, vol. 8, pp. 198101–198117, 2020.
[25] W. Dai, H. Lu, J. Xiao, Z. Zeng, and Z. Zheng, "Multi-robot dynamic task allocation for exploration and destruction," *J. Intell. Robot. Syst.*, vol. 98, no. 2, pp. 455–479, 2020.
[26] R. H. Kabir and K. Lee, "Efficient multi-robot exploration with energy constraint based on optimal transport theory," 2020, *arXiv:2009.00862*.
[27] J. Li, M. Ran, and L. Xie, "Efficient trajectory planning for multiple non-holonomic mobile robots via prioritized trajectory optimization," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 405–412, Apr. 2021.
[28] T. Salam and M. A. Hsieh, "Adaptive sampling and reduced-order modeling of dynamic processes by robot teams," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 477–484, Apr. 2019.
[29] A. Prorok, M. A. Hsieh, and V. Kumar, "Formalizing the impact of diversity on performance in a heterogeneous swarm of robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 5364–5371.
[30] J. Kiener and O. Von Stryk, "Towards cooperation of heterogeneous, autonomous robots: A case study of humanoid and wheeled robots," *Robot. Auton. Syst.*, vol. 58, no. 7, pp. 921–929, 2010.
[31] C. Ju and H. I. Son, "Modeling and control of heterogeneous agricultural field robots based on Ramadge–Wonham theory," *IEEE Robot. Automat. Lett.*, vol. 5, no. 1, pp. 48–55, Jan. 2020.
[32] H. S. Ahn, I. Sa, and F. Dayoub, "Introduction to the special issue on precision agricultural robotics and autonomous farming technologies," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 4435–4438, Oct. 2018.
[33] M. Karrer, P. Schmuck, and M. Chli, "CVI-SLAM–collaborative visual-inertial SLAM," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 2762–2769, Oct. 2018.
[34] E. R. Boroson and N. Ayanian, "3D keypoint repeatability for heterogeneous multi-robot slam," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 6337–6343.
[35] I. Abraham and T. D. Murphey, "Decentralized ergodic control: Distribution-driven sensing and exploration for multiagent systems," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 2987–2994, Oct. 2018.
[36] R. Lampariello, D. Nguyen-Tuong, C. Castellini, G. Hirzinger, and J. Peters, "Trajectory planning for optimal robot catching in real-time," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 3719–3726.
[37] O. Koç, G. Maeda, and J. Peters, "Online optimal trajectory generation for robot table tennis," *Robot. Auton. Syst.*, vol. 105, pp. 121–137, 2018.
[38] Q. B. Diep, I. Zelinka, and S. Das, "Self-organizing migrating algorithm pareto," *Mendel*, vol. 25, no. 1, pp. 111–120, 2019.
[39] W. R. Ashby, *An Introduction to Cybernetics*. London, U.K.: Chapman & Hall Ltd, 1961.
[40] N. Wiener, *Cybernetics or Control and Communication in the Animal and the Machine*. Cambridge, MA, USA: MIT Press, 2019.
[41] A. Brunetti, D. Buongiorno, G. F. Trotta, and V. Bevilacqua, "Computer vision and deep learning techniques for pedestrian detection and tracking: A survey," *Neurocomputing*, vol. 300, pp. 17–33, 2018.
[42] R. Opromolla, G. Fasano, and D. Accardo, "A vision-based approach to UAV detection and tracking in cooperative applications," *Sensors*, vol. 18, no. 10, 2018, Art. no. 3391.
[43] M. Sualeh and G.-W. Kim, "Dynamic multi-LiDAR based multiple object detection and tracking," *Sensors*, vol. 19, no. 6, 2019, Art. no. 1474.
[44] J. Zhao, H. Xu, H. Liu, J. Wu, Y. Zheng, and D. Wu, "Detection and tracking of pedestrians and vehicles using roadside LiDAR sensors," *Transp. Res. Part C: Emerg. Technol.*, vol. 100, pp. 68–87, 2019.
[45] M. C. P. Santos, C. D. Rosales, M. Sarcinelli-Filho, and R. Carelli, "A novel null-space-based UAV trajectory tracking controller with collision avoidance," *IEEE/ASME Trans. Mechatron.*, vol. 22, no. 6, pp. 2543–2553, Dec. 2017.
[46] A. M. Romanov et al., "Modular reconfigurable robot distributed computing system for tracking multiple objects," IEEE Syst. J., vol. 15, no. 1, pp. 802–813, Mar. 2021.
[47] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[48] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artif. Intell.*, vol. 219, pp. 40–66, 2015.
[49] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 01, 2019, pp. 7643–7650.
[50] X. Lei, Z. Zhang, and P. Dong, "Dynamic path planning of unknown environment based on deep reinforcement learning," *J. Robot.*, vol. 2018, 2018, Art. no. 5781591.
[51] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6252–6259.
[52] X. Chen, A. Ghadirzadeh, J. Folkesson, M. Björkman, and P. Jensfelt, "Deep reinforcement learning to acquire navigation skills for wheel-legged robots in complex environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 3110–3116.
[53] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 512–519.
[54] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*,, 2017, pp. 1–11.
[55] C. Berner et al., "Dota 2 with large scale deep reinforcement learning," 2019, *arXiv:1912.06680*.
[56] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4295–4304.
[57] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "Qplex: Duplex dueling multi-agent Q-learning," in *Proc. Int. Conf. Learn. Representations*, 2020.
[58] A. Cohen et al., "On the use and misuse of absorbing states in multi-agent reinforcement learning," 2022.
[59] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018, pp. 2974–2982.
[60] D. T. Nguyen, A. Kumar, and H. C. Lau, "Credit assignment for collective multiagent RL with global rewards," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–12.
[61] G. Sartoretti, W. Paivine, Y. Shi, Y. Wu, and H. Choset, "Distributed learning of decentralized control policies for articulated mobile robots," *IEEE Trans. Robot.*, vol. 35, no. 5, pp. 1109–1122, Oct. 2019.
[62] A. L. Samuel, "Some studies in machine learning using the game of checkers. II–recent progress," *IBM J. Res. Develop.*, vol. 11, no. 6, pp. 601–617, 1967.
[63] D. Hernandez et al., "A generalized framework for self-play training," in *Proc. IEEE Conf. Games*, 2019, pp. 1–8.
[64] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[65] S. C. Raparthy, B. Mehta, F. Golemo, and L. Paull, "Generating automatic curricula via self-supervised active domain randomization," 2020, *arXiv:2002.07911*.
[66] O. OpenAI et al., "Asymmetric self-play for automatic goal discovery in robotic manipulation," 2021. [Online]. Available: https://openreview.net/forum?id=hu2aMLzOxC

[67] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus, "Intrinsic motivation and automatic curricula via asymmetric self-play," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–16.
[68] J. L. Elman, "Learning and development in neural networks: The importance of starting small," *Cognition*, vol. 48, no. 1, pp. 71–99, 1993.
[69] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 41–48.
[70] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. Cham, Switzerland: Springer, 2016.
[71] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage," *IEEE Commun. Lett.*, vol. 20, no. 8, pp. 1647–1650, Aug. 2016.
[72] E. Kalantari, H. Yanikomeroglu, and A. Yongacoglu, "On the number and 3D placement of drone base stations in wireless cellular networks," in *Proc. IEEE 84th Veh. Technol. Conf.*, 2016, pp. 1–6.
[73] J. Lyu, Y. Zeng, R. Zhang, and T. J. Lim, "Placement optimization of UAV-mounted mobile base stations," IEEE Commun. Lett., vol. 21, no. 3, pp. 604–607, Mar. 2016.
[74] K. Young and T. Tian, "Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments," 2019, *arXiv:1903.03176*.
[75] A. Juliani et al., "Unity: A general platform for intelligent agents," 2018, *arXiv:1809.02627*.
[76] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 23–30.
[77] B. Katz, J. D. Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 6295–6301.
[78] M. Abdoos, N. Mozayani, and A. L. Bazzan, "Traffic light control in non-stationary environments based on multi agent q-learning," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst.*, 2011, pp. 1580–1585.
[79] S. Josef and A. Degani, "Deep reinforcement learning for safe local planning of a ground vehicle in unknown rough terrain," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 6748–6755, Oct. 2020.
[80] I. Rahwan et al., "Machine behaviour," *Nature*, vol. 568, no. 7753, pp. 477–486, 2019.



Yuan Gao (Member, IEEE) received the M.S. degree in machine learning and algorithms from the University of Helsinki, Helsinki, Finland, in 2016, and the Ph.D. degree in Computer Science from Uppsala University, Uppsala, Sweden, in 2020, under the supervision of Prof. Ginevra Castellano and Prof. Danica Kragic.

He is currently an Assistant Research Scientist with the Shenzhen Institute of Artificial intelligence and Robotics for Society, Shenzhen, China. He has authored or coauthored robotics journals (e.g., IEEE TRANSACTIONS ON ROBOTICS, IEEE/ASME TRANSACTIONS ON MECHATRONICS, IEEE ROBOTICS AND AUTOMATION LETTERS). His research interests include machine behavior analysis, reinforcement learning, robotics, and general machine learning.

Junfeng Chen received B.S. and M.S. degrees in aircraft vehicle design from the School of Aeronautic Science and Engineering, Beihang University, Beijing, China, in 2019. He is currently working toward the Ph.D. degree in computer science with Beijing University, Beijing, China.

His research interests include robotics, robot learning, planning, and multirobot systems.

Xi Chen received the B.S. and M.S. degrees in computer science and technology from Sungkyunkwan University, Seoul, South Korea, in 2012 and 2014, respectively, and the Ph.D. degree in Computer Science from the Division of Robotics, Perception, and Learning, Royal Institute of Technology, in 2020.

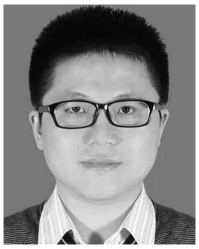
She is currently a Postdoc with the Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China. Her research interests include reinforcement learning, robotics, and general artificial general intelligence.

Chongyang Wang received the B.Eng. degree in electronic engineering from Southwest University in 2017, and the Ph.D. degree from University College London in 2022. He is currently a Postdoc at Department of Computer Science and Technology, Tsinghua University. His research interests include ubiquitous computing, human-computer interaction, and intelligent healthcare.

His research interests include intelligent healthcare. Particularly, he is focused on developing ubiquitous body sensing technology to support physical rehabilitation of people with chronic pain.

Junjie Hu (Member, IEEE) received the B.S. degree in computer science and technology from the Tianjin University of Science and Technology, Tianjin, China, in 2014, and the M.S. and Ph.D. degrees in computer vision and machine learning respectively, from the Graduate School of Information Science, Tohoku University, Sendai, Japan, in 2017 and 2020, respectively.

He is currently a Researcher with the Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen, China. His research interest includes machine learning, computer vision, and robotics.



Fuqin Deng (Member, IEEE) received the M.S. degree in control science and engineering from the Shenzhen Graduate School of Harbin Institute of Technology, Shenzhen, China, in 2007 and the Ph.D. degree in electrical and electronic engineering from University of Hong Kong, Hong Kong, in 2014.

He is a Distinguished Professor with School of Intelligent Manufacturing, Wuyi University, Nanping, China. His research interests include multirobot systems, machine learning, and machine vision applications.

Tin Lun Lam (Senior Member, IEEE) received the B.Eng. and Ph.D. degrees in automation and computer-aided engineering from The Chinese University of Hong Kong, Hong Kong, in 2006 and 2010, respectively.

He is currently an Assistant Professor with the School of Science Engineering, The Chinese University of Hong Kong. His research interests include multirobot systems, field robotics and human–robot interaction.