

# Solving Multi-Entity Robotic Problems Using Permutation Invariant Neural Networks

Tianxu An<sup>\*1</sup>, Joonho Lee<sup>\*1</sup>, Marko Bjelonic<sup>2</sup>, Flavio De Vincenti<sup>3</sup>, Marco Hutter<sup>1</sup>

**Abstract**—Challenges in real-world robotic applications often stem from managing multiple, dynamically varying entities such as neighboring robots, manipulable objects, and navigation goals. Existing multi-agent control strategies face scalability limitations, struggling to handle arbitrary numbers of entities. Additionally, they often rely on engineered heuristics for assigning entities among agents. We propose a data driven approach to address these limitations by introducing a decentralized control system using neural network policies trained in simulation. Leveraging permutation invariant neural network architectures and model-free reinforcement learning, our approach allows control agents to autonomously determine the relative importance of different entities without being biased by ordering or limited by a fixed capacity. We validate our approach through both simulations and real-world experiments involving multiple wheeled-legged quadrupedal robots, demonstrating their collaborative control capabilities. We prove the effectiveness of our architectural choice through experiments with three exemplary multi-entity problems. Our analysis underscores the pivotal role of the end-to-end trained permutation invariant encoders in achieving scalability and improving the task performance in multi-object manipulation or multi-goal navigation problems. The adaptability of our policy is further evidenced by its ability to manage varying numbers of entities in a zero-shot manner, showcasing near-optimal autonomous task distribution and collision avoidance behaviors.

## I. INTRODUCTION

HUMANS routinely deal with multi-entity problems in daily life. For example, human workers collaborate with co-workers to construct structures, or a group of friends splits up to find various products in a grocery store. In such scenarios, individuals naturally decide where to work within the site, with whom to collaborate, and which object or sub-task to prioritize. Can robots have similar capabilities? Despite recent advances that have made robots proficient in tasks like locomotion [1, 2, 3], navigation [4, 5, 6], and object manipulation [7, 8, 9], existing approaches cannot be readily applied to collaborative multi-entity tasks. The majority of literature focuses on solving single-robot problems, leaving a gap in addressing the complexities of multi-agent collaboration and coordination in real-world scenarios.

Our goal is to develop a framework to tackle multi-entity problems involving multiple mobile robots, as illustrated in Fig. 1. To explore this problem, we define three multi-entity

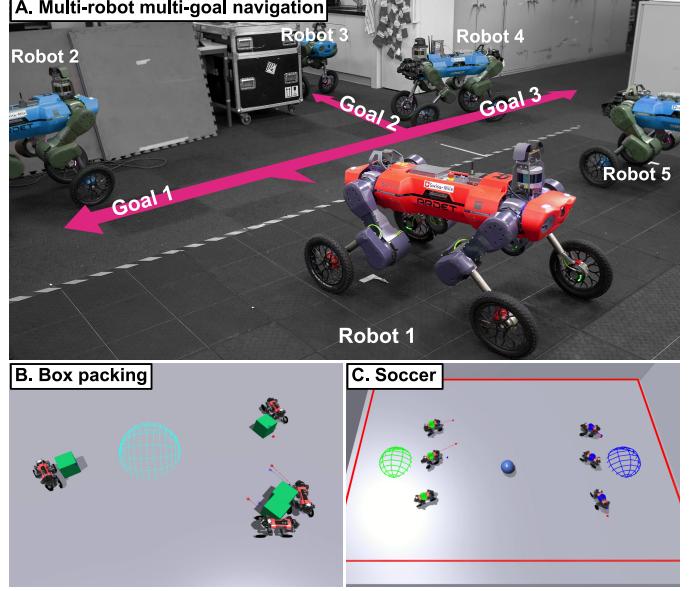


Fig. 1. Multi-entity problems studied in this work. (A) Robot 1 is given multiple goals to visit while interacting with other robots. (B) Multiple robots packing multiple boxes into the goal region. (C) Soccer.

problems that involve four entity types: collaborators, opponents, navigation goals, and objects to manipulate. Specifically, Fig. 1A shows a multi-robot multi-goal (MRMG) navigation problem where robots are given multiple navigation goals, Fig. 1B illustrates box packing problem where robots have to move boxes to the packing site in the center, and Fig. 1C shows a soccer game. Each task episode may have a different number of entities of each type. We aim to develop a control strategy that can effectively handle flexible number of entities and generalize to new scenarios.

Existing approaches to controlling multi-agent systems (MAS) struggle at tackling two main challenges. First, scaling up to flexible numbers of entities. Traditional centralized methods [10, 11, 12, 13] struggle at scaling their controllers to higher numbers of entities, because the number of decision variables can grow arbitrarily depending on the number of entities, leading to high computational cost. Although decentralized controllers can alleviate the computation cost, many of them still focus on tasks with a fixed number of entities. For example, multi-agent controllers in [14, 15, 16] only accept a fixed number of robots. Similarly, multi-agent policy by Baker et al. [17] limits the maximum numbers of agents and manipulable objects. Second, prioritization and assignment of goals and objects among agents is not straightforward. A common

<sup>\*</sup> Tianxu An and Joonho Lee contributed equally to this work. (Corresponding author: Joonho Lee).

<sup>1</sup> The authors are with Robotic Systems Lab; ETH Zurich, Switzerland

<sup>2</sup> The author was with Robotic Systems Lab; ETH Zurich, Switzerland. The author is now with Swiss-Mile Robotics AG, Switzerland.

<sup>3</sup> The author is with Computational Robotics Lab; ETH Zurich, Switzerland

practice is to assign goals and objects among agents based on predefined rules or heuristics, such as manually assigning each agent with a specific goal [18, 19, 20] or assigning objects based on proximity or manual commands [21, 22, 23].

To tackle these challenges, we develop decentralized multi-agent reinforcement learning (MARL) policies that enable a robot to work with flexible numbers of entities. We adopt a decentralized approach to maintain constant inference time, leveraging distributed computational costs regardless of the number of agents. Additionally, we opt for a model-free reinforcement learning (RL) approach to eliminate hand-engineered heuristics typically present in optimization-based approaches.

We further enhance the performance of our agents when prioritizing and distributing entities in long-horizon tasks by using a permutation-invariant neural network called global entity encoder (GEE). Leveraging an architecture similar to PointNet [24], GEE processes an arbitrary number of state vectors from all entities for each entity type. This architectural choice ensures that agents can act consistently regardless of the order in which they perceive the entities. In other words, the action remains invariant to the permutations in the input space (i.e., order of entities). By eliminating order dependency in the decision making, we enable agents to assess the relative importance of different entities without being biased by ordering. In doing so, we remove the need for engineered heuristics.

We validate our approach on the the previously mentioned multi-entity problems in Fig. 1. For each problem, we use the same policy to control each robot. The robots with identical policy collaborate with an arbitrary number of other robots across various task configurations. Our permutation invariant network policy effectively addresses multi-entity tasks and outperforms typical fixed-input encoders. Simulation experiments show that our approach enables robots to autonomously prioritize entities depending on the situation and engage in collaborative problem solving.

Furthermore, we conduct real-world MRMG navigation experiments with two wheeled-legged quadrupedal robots [25], with communication facilitated by WI-FI and gRPC framework [26]. Our real-world experiments demonstrate intelligent multi-agent navigation behavior, where robots autonomously distribute tasks and avoid collisions.

Our key contributions are as follows:

- 1) We introduce a scalable control policy for multi-entity robotic problems. Our policy is based on permutation invariant neural networks, which allows it to handle flexible numbers of different entities.
- 2) We verify the effectiveness of our system through real-world multi-entity navigation experiments with two wheeled-legged quadrupedal robots [25]. We provide important technical details regarding training and hardware implementation for communication.

## II. RELATED WORK

### A. Permutation Invariant Neural Networks

Permutation invariant neural networks ensure output invariance to permutations of the input data. Set inputs are

common in the robotics domain and take the form of point clouds, sensory inputs, and neighboring agents' states in multi-agent problems. Various network structures achieve the permutation invariance property. For instance, PointNet [24] and DGCNN [27] use pooling operations on point cloud inputs. Self-attention encoders [28] can also preserve permutation invariant inputs by removing any positional encodings from the input elements [29, 30]. Another line of work uses Mean-Field Approximation to model large-scale multi-agent states or collective actions, thus maintaining the permutation invariance properties of neighboring agents [31, 32]. Likewise, Zhang et al. [33] ensure the permutation invariance using explicit constraints. They introduce a decentralized multi-agent navigation algorithm that constrains the network outputs to match those with shuffled orders of neighboring agents. In this work, we adopt the PointNet [24] architecture to ensure permutation invariance of all entities in the same category, such as robots, objects, and goals.

### B. Multi-object Manipulation

Many research tackle the problem of controlling robots to manipulate multiple objects. A large body of work in multi-object manipulation focuses on a single manipulator sequentially packing various objects [7, 8, 34]. These methods prioritize identifying individual objects and their visual relationship with the background but de-emphasize the inter-object relationships and the manipulation sequence. Li et al. [9] and Huang et al. [35] use graph neural networks (GNN) to reason about relationships among variable numbers of objects, enabling a single manipulator arm to stack blocks into multiple configurations by planning a manipulation sequence.

It became more popular recently to control multiple agents to increase efficiency in multi-object manipulation. Envall et al. [22] propose a differentiable task assignment and motion planning (TAMP) strategy to control multiple manipulator arms to smoothly handover blocks. The authors also experiment on the scenarios with multiple objects. However, the approach uses manual heuristics to decide the manipulation sequence. Another line of work in multi-agent multi-object manipulation is multi-agent foraging (MAF) [36, 37]. MAF problems closely resemble our box packing task, both of which ask a team of agents to pack arbitrary numbers of loads into one or multiple packing sites. Agents in MAF often need to closely interact with collaborators and automatically determine the manipulation sequence of objects. Similar to MAF, in our box packing task, we ask multiple robots to implicitly plan a manipulation sequence that considers all objects and collaborators. This requires each robot to discern the relationship of objects with respect to the target, collaborators, and themselves.

### C. Multi-goal Navigation

When it comes to multi-goal navigation, a robot usually receives all the goal locations at the beginning and then navigates to each one. Schmalstieg et al. [38] use a greedy strategy that always navigates the robot to the closest goal in an unknown map, while Chen et al. [5] propose computing an

offline path from a known map without a specific goal-visiting order and following the planned path during execution. Both methods are likely to get stuck if a goal point is unreachable, as common in real-world scenarios, e.g., due to improper goal setup or unexpected map changes.

For MRMG navigation, early works separate goal assignment and cooperative path planning into two problems [19, 20]. More recent works apply MARL algorithms to let robots autonomously decide goal assignments and cooperative path planning with neighbor awareness [37, 39]. Similar to this approach, we couple goal assignment and cooperative path planning into a single problem. Our policies are optimized synchronously to prioritize and distribute goals while planning their motion to avoid collisions with neighbors.

#### D. Competitive RL

Competitive RL adopts self-play to simultaneously improve policies of the training teams and their opponents in a wide range of games, such as hide and seek, Go, soccer, and Sumo [17, 40, 41, 42, 43]. In our soccer task, we also apply self-play. Unlike soccer policies in [41, 42], our policies do not restrict the number of players on either team, thanks to the permutation invariant encoder which enable ego robot to accept arbitrary numbers of teammates and opponents.

#### E. Neighbor Awareness in Multi-Agent Systems

Most MASs require robots to be aware of their neighbors to facilitate effective collaborative behaviors. Early works consider fully decentralized controllers and attempt to induce neighbor awareness through environment changes [16, 44, 45, 46]. However, these approaches are limited when agents work closely, as the agents can collide with each other due to the lack of mutual observability. In contrast, centralized strategies guarantee global neighbor awareness, but they suffer from high computational burdens and scalability issues [10, 11, 12, 13]. To mitigate these problems while incentivizing collaboration, some researchers adopt the centralized training decentralized execution (CTDE) paradigm [47, 48, 49, 50, 51, 52, 53]. In this approach, the agents do not actively communicate at execution time, and all emerging collaborations are the result of a centralized training in physics-based simulators.

In recent times, there has been an increase in the use of wireless networks to enable communication between agents for enhanced collaboration during the execution. Optimization-based controllers have utilized these communication channels to control a swarm of drones for navigation in the wild [54]. However, optimization-based methods are often computationally expensive and may require heuristics, resulting in sub-optimal performance. Therefore, many researchers have combined communication networks with decentralized RL policies to achieve faster execution times and more robust behaviors.

Early approaches using RL endow agents with an auxiliary neural network for generating communication messages and deciding whether to share them with others [55, 56]. However, in these approaches, receiving agents passively process the messages either discretely or by aggregating their mean from all broadcasting neighbors. This approach can be problematic

because each agent essentially treats all neighboring agents as equally important, often resulting in inundation with irrelevant information while potentially overlooking crucial neighbors. ATOC [39] tackles this issue by employing an Attention Network to determine if an agent should initiate communication, while a bi-directional LSTM network handles shared messages from collaborators. Ha et al. [18] employ a similar approach using LSTM for message aggregation to enable a variable number of manipulators to converge on a joint end-effector state using a decentralized control policy for each. Nevertheless, using LSTM as the aggregator poses challenges as neighboring agents are not permutation invariant, necessitating the reordering of neighbors based on hand-engineered heuristics. This can lead to earlier neighbors' states being forgotten, potentially limiting the ego agent's ability to learn from the most important neighbors.

Niu et al. [57] introduces a Graph-Attention based approach aimed at enhancing multi-agent coordination. Initially, a central planner learns a graph representing communication directions at each policy step. Subsequently, an Attention network is employed for each agent to prioritize messages from neighbors, considering incoming communication directions, including messages from the agent itself. While this mechanism ensures permutation invariance in neighbor orders, scalability issues arise due to the centralized system. Das et al. [58] adopt a similar approach, but each agent solely relies on an Attention network to determine message priorities based on the Attention weights of other agents. More recently, Yu et al. [59] have also leveraged Attention Networks in MARL for real-robot exploration tasks within simplified grid worlds.

Another line of work to facilitate agent communication is by constructing overhead 2-D maps, from where ego agents can observe information of others [36, 37]. Inspired by social insects such as ants, Shaw et al. [36] allow agents to communicate with temporary pheromone trails in 2-D maps. On the other hand, Wu et al. [37] generates agents' 2-D intention maps to inform others about their next actions. As a form of visual information, orders of neighboring agents in overhead 2-D maps are naturally permutation invariant. However, this approach restricts working environments to single-level terrains.

In our work, we use a lightweight encoder architecture inspired by PointNet [24]. This architecture allows each agent to process information from a variable number of permutation invariant entities, effectively facilitating intelligent robot behaviors that dynamically focus on key entities during policy execution.

## III. METHOD

#### A. Overview

The overview of our MARL pipeline is shown in Fig. 2. We adopt a hierarchical architecture, similar to [16], where high-level policies produce body position or velocity commands to the low-level policies. In this work, our focus lies on training high-level policies. The low-level policy is pre-trained and fixed. Each agent observes entities using separate GEEs per entity type. (Fig. 2A, B) These GEEs produce fixed size feature

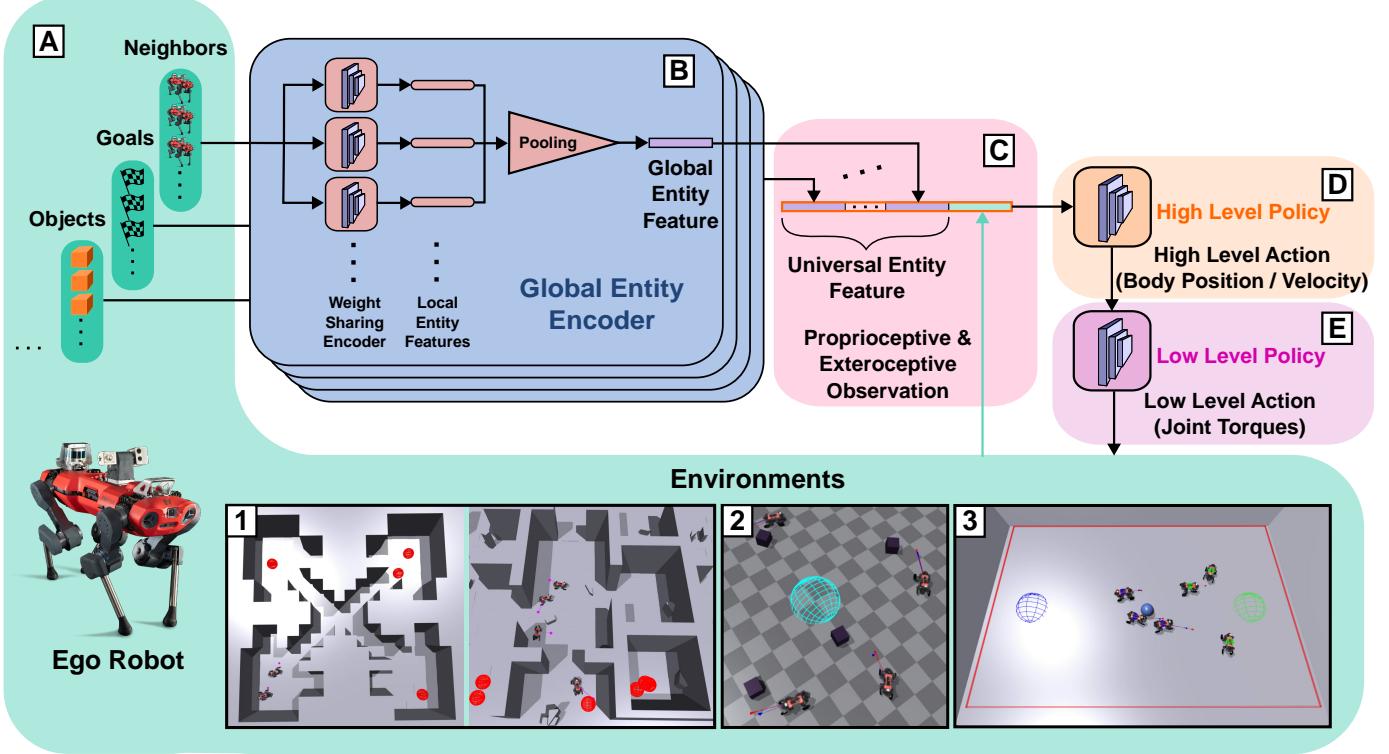


Fig. 2. Pipeline overview for a decentralized ego robot. (A) The ego robot and its environments: robot can observe entities such as neighbors, goals, objects, etc. (A-1) MRMG navigation environments. (A-2) Box packing environment. (A-3) Soccer environment. (B) All entities belonging to one category are passed to one GEE. Their entity states are first passed into individual weight sharing encoders to get local entity features. The local entity features are then max-pooled to obtain the global entity feature for this entity category. (C) All global entity features belonging to different entity categories are concatenated to form the universal entity feature, which is then concatenated with the ego robot’s proprioceptive & exteroceptive observations from the environment to form the input to the high-level policy. (D) The high-level policy is an RL policy network that outputs high-level actions for the ego robot. The high-level actions can be the target body position or body velocity, depending on required task settings. (E) The low-level policy takes high-level actions and outputs joint torques to control the ego robot. The low-level policy is pretrained based on the work by Lee et al. [60] and fixed when training high-level policies.

vectors via pooling operations for each entity type (Fig. 2C). These features, along with the robot’s local observations, are then concatenated and provided to the downstream part of the network. The local observations include both proprioceptive and exteroceptive measurements. The GEE is trained end-to-end via RL.

For our hardware experiments, we establish a robot communication pipeline using the gRPC framework [26], as described in Appendix D.

### B. Problem Formulation

We model the multi-entity problems mentioned in Fig. 1 as decentralized partially observable Markov decision process (Dec-POMDP)s. A Dec-POMDP is defined by a tuple:

$$\langle \mathcal{A}, \mathcal{S}, \mathcal{U}, \Omega, \mathcal{O}, \mathcal{P}, \mathcal{R}, \gamma \rangle,$$

where  $\mathcal{A}$  denotes the set of all robots  $\mathcal{A} = \{a^n\}_{n=1}^N$ , and  $s \in \mathcal{S}$  is the global state of the environment. Unlike previous works [15, 16, 42], where the number of robots  $N$  is fixed for all environments, we allow  $N$  to be a random system state. Since we use identical robots, they all share the same action space  $\mathcal{U}$  and observation space  $\Omega$ . We use  $\mathbf{o} \in \Omega^N$  and  $\mathbf{u} \in \mathcal{U}^N$  to represent the joint observations and joint actions of all robots, where  $\mathbf{o} = (o^1, \dots, o^N)$  and  $\mathbf{u} = (u^1, \dots, u^N)$ . Due to partial observability, each robot  $a^n$  cannot observe the

full global state  $s_t$  at time step  $t$ , but a partial observation state  $\mathbf{o}_t^n \in \Omega$  output by the observation function  $O(s_t, a^n)$ .  $\mathcal{P}$  denotes the transition dynamics, with  $\mathcal{P}(s_{t+1}|s_t, \mathbf{u}_t)$  being the probability of transitioning to the next global state  $s_{t+1}$  by taking the joint action  $\mathbf{u}$  at global state  $s_t$ .  $\mathcal{R}: \mathcal{S} \times \mathcal{U} \rightarrow \mathbb{R}$  is the common reward function shared by all robots, and  $\gamma \in [0, 1]$  is the discount factor. The robots collectively roll out a trajectory  $\tau = (s_0, \mathbf{u}_0, s_1, \dots)$ , based on the shared decentralized policy  $\pi(u^n|o^n): \Omega \times \mathcal{U} \rightarrow [0, 1]$ . The goal of all robots is to find the best policy to maximize the cumulative discounted reward

$$\mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, \mathbf{u}_t) \right].$$

We used two different action spaces. For MRMG navigation, the high-level policy commands target body  $xy$  positions in the current baseframe. This is then tracked by a position tracking controller by [61] for robot experiments. For other problems, we used target body velocities consisting of linear  $xy$  velocity and yaw rates in base frame for simplicity. More implementation details can be found in Appendix A, and we refer the interested reader to [60] for more details about the low-level policy. To ensure that the outputs of our high-level policies adhere to the constraints of our robot’s physical capabilities, we adopt a bounded action space based on Beta distribution [62, 63]. We refer readers to Appendix B for more information.

The rewards and observations for our three Dec-POMDPs are displayed in Table III and IV in the Appendix, for brevity.

In all experiments, the agents do not have access to the global map information. Each agent only observes a  $4.35\text{ m} \times 2.85\text{ m}$  rectangular local height map around itself.

### C. Global Entity Encoder

Since we let the number of robots  $N$  be random, each robot  $a^n$  needs to observe flexible numbers of neighbors in a fixed size observation vector  $o^n$ . Such variability of numbers also applies to other task-related entities, such as navigation goals, boxes, opponents, etc. To this end, we approximate the observation function as:

$$o^n = O(s, a^n) \simeq O(f(s, a^n), s^n), \quad (1)$$

where  $f(s, a^n)$  is the universal entity feature (Fig. 2C) and  $s^n$  is the local observation of robot  $a^n$ . The universal entity feature is the concatenation of a number  $E$  of global entity features, each corresponding to a unique entity category in the environment:

$$f(s, a^n) = [g^1(s, a^n), \dots, g^E(s, a^n)]. \quad (2)$$

Each  $g$  represents a global entity feature from a GEE. In our box packing task, for example, two GEEs are used ( $E = 2$ ), where one is applied to all neighbors while another is applied to all boxes. Depending on the number of entity categories in the environment, any number of GEEs can be chosen for a given task.

The operation to get a global entity feature  $g$  and the universal entity feature  $f$  is shown in Fig. 2B and 2C. Similarly to PointNet [24], for each robot  $a^n$ , all observed entities from the same category are passed to a weight sharing MLP to get corresponding local entity features; then, all local entity features are max-pooled to obtain an associated global entity feature  $g$ . All global entity features representing different entity categories are concatenated to get the universal entity feature  $f$ . Together with the robot's proprioceptive and exteroceptive observations from the environment, the observation state  $o^n$  is passed to the Policy Net to get a single robot's action  $u^n$ .

### D. Training Environments

Here we describe the training environments for the three multi-entity problems. All environments have time limit of 45 s and the number of entities is randomized per episode.

1) *MRMG Navigation*: The MRMG navigation task requires a group of robots to visit multiple goal positions. Each goal has to be visited at least once by any robots. We represent each goal as a 0.5 m radius sphere and initialize it 15–30 m away from the centroid of the robot group. An episode ends successfully when all goals are reached, or ends in failure if the time limit is exceeded. We design different terrains as shown in Fig. 2A-1. During training, each episode involve a maximum of four robots and up to four goals, with the number of entities varying between episodes. Note that in policy deployment, the number of entities can be higher than the maximum values in the training stage.

A terrain curriculum similar to Rudin et al. [2] is used, where the terrain becomes more difficult if a task is completed successfully, or becomes easier if no goal is reached. Terrain difficulty is associated with certain attributes, such as the narrowness of corridors, ground flatness, height of stairs, etc.

2) *Box Packing*: The box packing task requires a group of robots to push a group of boxes to a packing site, as shown in Fig. 2A-2. The packing site is a 1.5 m radius sphere, while the robots and boxes are randomly positioned in various direction within 1.5–10 m. An episode ends in failure if any robot falls or if the time limit is exceeded. Similar to MRMG navigation, each training episode involve a maximum of four robots and up to four boxes, with the number of entities varying between episodes.

We uniformly sample the initial distances of the robots and boxes from the packing site. If the robots successfully push all boxes to the packing site, we increase the upper bounds of the sampling range by 1 m. Otherwise, we decrease both upper bounds by 1 m. The sampling range is capped between 2–10 m for robots and 1.5–10 m for boxes.

3) *Soccer*: Two groups of robots try to push a ball into the other team's goal while defending their own. As shown in Fig. 2A-3, the goal is a 1 m-radius sphere located near the end of each side of the soccer environment. The field is a  $18\text{ m} \times 12\text{ m}$  rectangle and  $45^\circ$  slopes are installed at the field boundaries to prevent the ball from rolling away. At the beginning, all robots are lined up in a row in front of their respective goals, and the soccer ball is initialized at the center with random initial velocity. An episode terminates whenever the ball enters either team's goal, or if the time limit is exceeded. We assign 1–3 robots on the training side, while 0–3 robots are on the opponent side.

We implemented a self-play curriculum, a popular method in competitive RL implementations [17, 41, 42, 43]. In this strategy, the robots on the opponent team are equipped with earlier policies used against the training team. This allows both teams' strategies to improve simultaneously.

## IV. EXPERIMENTS AND RESULTS

We validate our approach in a series of hardware and simulation experiments. Our tests on real robots, as discussed in Section IV-A, qualitatively showcase the effectiveness of our policies. Section IV-B quantitatively analyzes our policies' internal reasoning during intensive robot collaborations and examines the impact on task performance when varying the number of collaborators. We further conduct ablation studies in Section IV-C to show the contribution of the GEE architecture and communication on robots' intelligent collaborative behaviors. Finally, we compare our MARL policy with a centralized optimization-based controller in a simplified multi-agent task in Section IV-D and show that our policy automatically learns near-optimal solutions.

### A. Robot Navigation

We conduct three different multi-entity navigation experiments using two wheeled-legged quadrupedal robots [25]. The

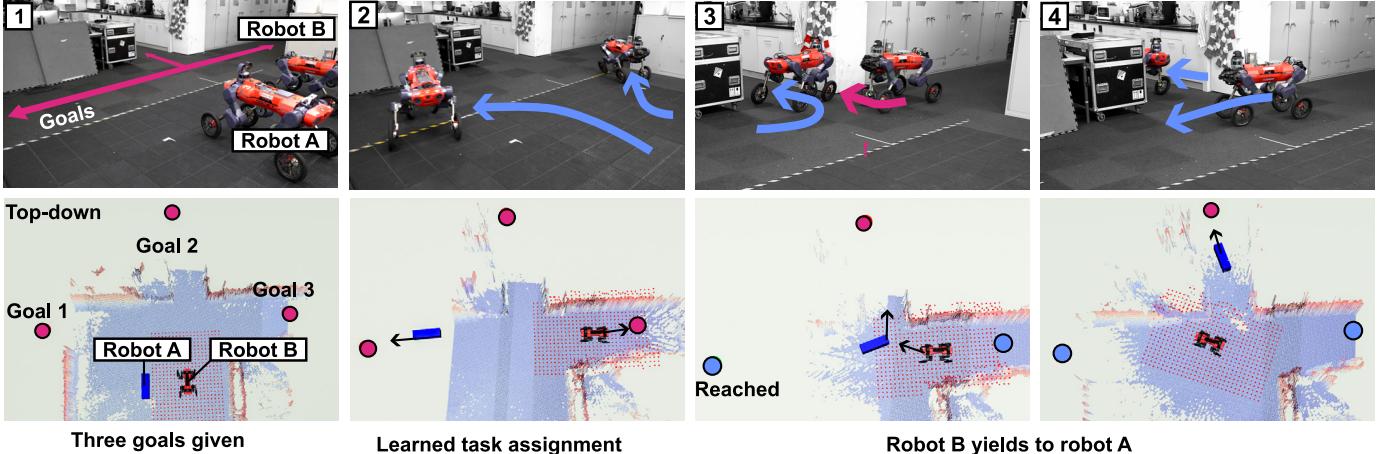
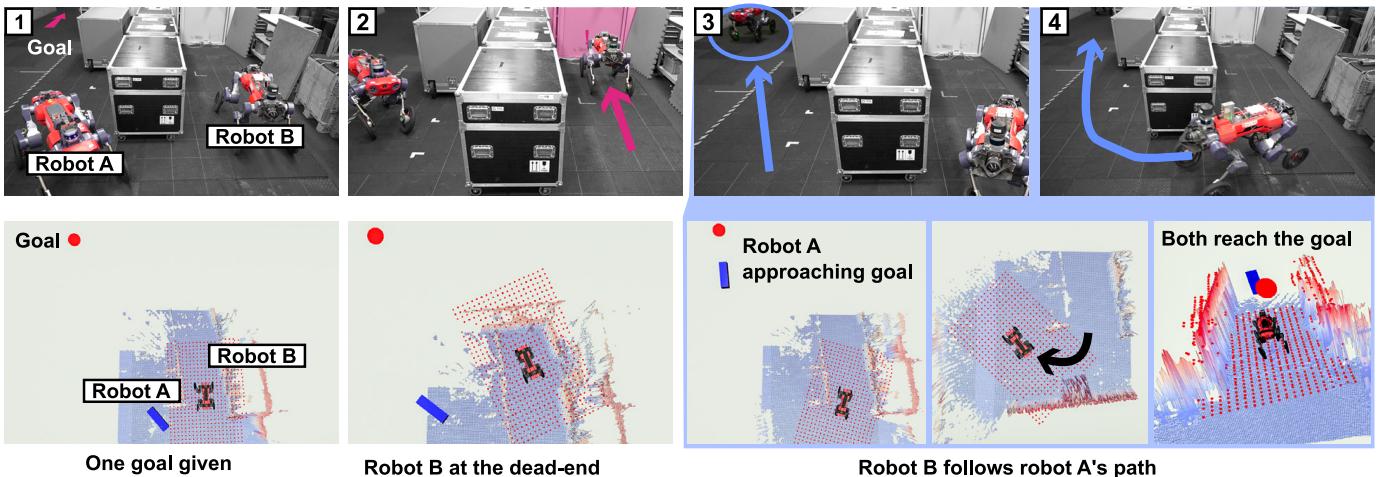
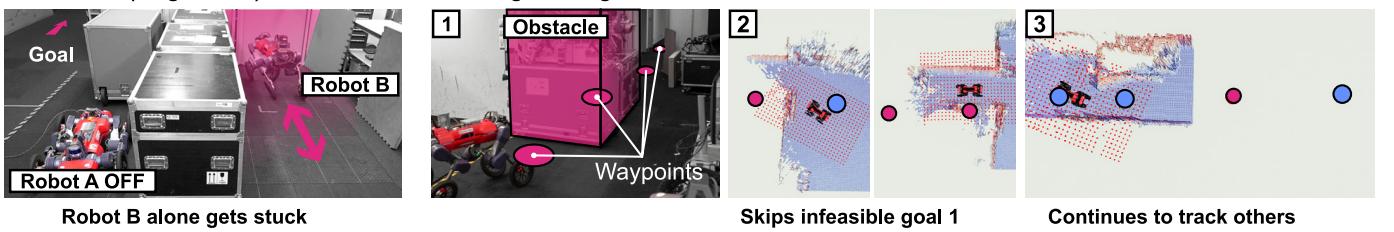
**A. Multi-robot multi-goal****B. Dead-end (Two robots)****C. Dead-end (Single robot)**

Fig. 3. Robot experiments. (A) MRMG navigation with two robots and three goals. (B, C) Dead-end experiment. (D) Single-robot Multi-goal navigation.

results are shown in the supplementary video. In each experiment, the same control policy is used across all robots, and the robots' states are shared through gRPC communication—see Appendix D. Robots in all experiments do not possess the global map or share any map information. Instead, each robot only observes a  $4.35\text{ m} \times 2.85\text{ m}$  rectangular local height scan centered on itself, as indicated by the red points in the top-down views in Fig. 3.

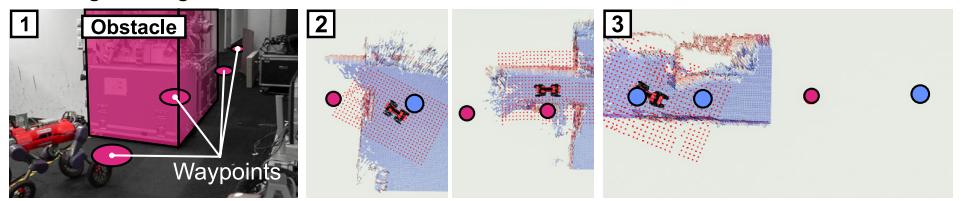
*1) MRMG Navigation:* Two robots are tasked with reaching three shared goals, as shown in Fig. 3A-1. The task is completed once all goals are reached, regardless of which robot reaches each goal. Each robot strategically chooses the closest goal while leaving the farther goal to the other robot (Fig. 3A-2). Upon finishing the first two goals (goal 1 and goal

3 in the figure), both robots try to reach goal 2 and meet at the narrow corridor. At the entrance, the robots avoid collision, and one robot stops and leaves the last goal to the other robot.

The distribution of goals and collision avoidance behavior observed in this experiment is learned, with no handcrafted state machine or task scheduling as seen in [20, 38].

*2) Dead End:* In the experiment shown in Fig. 3B and C, two robots are given a single waypoint to reach. Unlike the previous experiment, both robots have to reach the waypoint. Robot A has an obstacle-free path to the goal in the line of sight, whereas robot B starts from a corridor with a dead end. It is important to note that neither robot has access to a global map; they rely solely on local perception.

At the beginning, both robots navigate towards the goal.

**D. Multi-goal navigation**

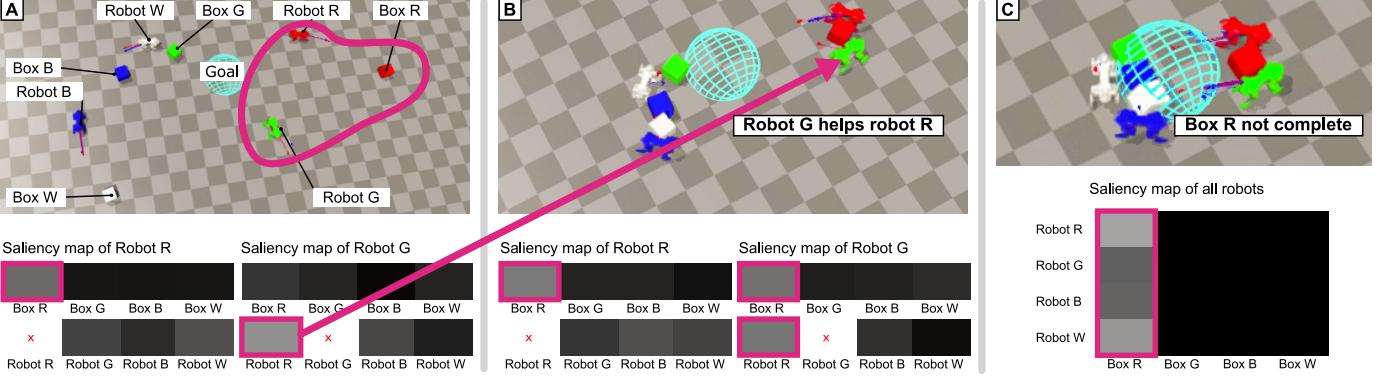


Fig. 4. Saliency maps of two robots during the box packing task. The four robots have to move the boxes into the goal point. (A) At the beginning, robot G does not focus on any box but attends to the states of robot R. (B) Then robot G joins robot R to transport the box R together. (C) Once all boxes are at the goal except for box R, all robots shift their focus to box R.

Then robot B encounters the dead end. Observing the robot A's successful path to the goal, the trapped robot B adapts and follows robot A's path, successfully escaping from the dead end. In contrast, Fig. 3C shows a single robot scenario. Without its collaborator, robot B fails to find the exit and becomes trapped in a local minimum, oscillating back and forth.

This experiment demonstrates the effectiveness of our approach: the control policy observes the other agents' behavior and utilizes this information to resolve navigation challenges.

3) *Multi-Goal Navigation with Obstacles*: Mobile robots are often tasked with multiple waypoints, e.g., path-following or multi-goal delivery tasks. Typically, the goals are given to the robot in a fixed order [5, 25, 38]. Additional state machines or human intervention may be necessary if a given waypoint becomes unreachable due to unexpected obstacles or environmental changes.

Our approach addresses this challenge by letting the policy determine the visiting order of waypoints dynamically. In another experiment, we give a robot multiple intermediate waypoints along a straight path, one of which is blocked by obstacles—see Fig. 3D. The robot navigates to the waypoints sequentially, automatically skipping the blocked waypoint. As shown in Fig. 3D-2, our multi-entity RL method can delegate such high-level decision-making challenges to the policy.

### B. Learned Collaborative Behaviors

We then analyse the emergent collaborative behaviors with two experiments on box packing task. For both experiments, we use the same policy trained in the environment depicted in Fig. 2A-2 following the training procedure in Section III-D2.

1) *Dynamic Focus on Neighboring Entities*: We conduct a sensitivity analysis to examine what entities each agent focuses on at each time step. Similarly to [1], we compute the gradient magnitude of the output feature of GEE with respect to each entity state. Intuitively, if an entity has no impact on the robot's behavior, perturbing the entity would not change the output of the GEE.

The resulting saliency map during the box packing task is shown in Fig. 4. It is observed that the robots strategically allocate their attention to boxes and adjacent robots that they

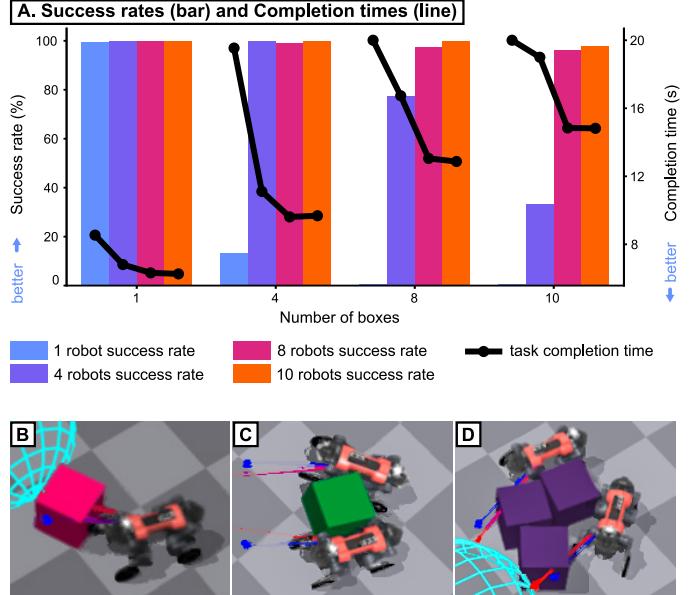


Fig. 5. Impact of Robot Numbers on Packing tasks. (A) Success Rate (bar columns) and Completion Time (dotted lines) of 1-10 robots packing 1-10 boxes. (B) 1 robot transports a box by walking sideways. (C) 2 robots are faster by squeezing the box in between them. (D) 2 robots can transport more than 1 box at the same time.

may interact with. This focus dynamically adapts in real-time to the evolving situation. This dynamic entity prioritization is a learned behavior, a significant advancement over traditional approaches. Conventional methods using heuristic-based task assignments lack the flexibility to adjust to changing contexts in this manner.

2) *Adaptation to Higher Entity Numbers*: We examine whether an increase in the number of robots leads to improved task performance, as well as the effectiveness of our policies across different numbers of targets. We trained a policy for the box packing task with a randomized number of robots and boxes, ranging up to four. We then execute the same policy with different combinations of robots and boxes.

Fig. 5A shows the success rate and completion time of different numbers of robots packing boxes within a 20 s time limit. We observe that, as the number of robots increases, the

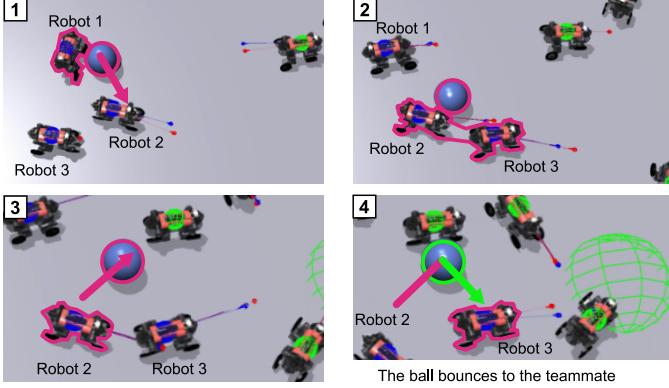


Fig. 6. Ball passing maneuver during the soccer game. The robots 1, 2, 3 advance towards the right side.

task is more likely to succeed and requires less time, especially when for higher numbers of boxes.

It is noteworthy that the completion time decreases as more robots are involved, even in scenarios with just one box. This tendency persists across all combinations of robot and box numbers once the number of robots surpasses the number of boxes.

The decrease in completion time would not be as significant if each box were always assigned to a single robot. We observe that when one robot pushes one box, it uses its side to avoid the box from slipping away (Fig. 5B). The presence of more robots allows for faster transportation by squeezing boxes between them, as shown in Fig. 5C and Fig. 5D.

This example shows a benefit of learning collaborative behaviors through RL. Such behavior cannot be easily hand-crafted, highlighting the effectiveness of our approach in solving complex problems.

### C. Enhanced Coordination with GEE

We identified two key components crucial to enhancing multi-robot coordination in our experiments: the observation of neighboring agents (neighbor awareness) and the Global Entity Encoder (GEE).

We conduct two ablation studies focusing on their roles in addressing multi-entity problems. We conducted two experiments with MRMG navigation (see Fig. 2A-1) and soccer environments (see Fig. 2A-3). The training procedures of the two policies are described in Section III-D1 and Section III-D3.

*1) Neighbor Awareness:* We first assess the influence of neighbor awareness on a 3 vs. 3 soccer game, as depicted in Fig. 6. Two teams are set to compete: one with full awareness of both teammates and opponents (neighbor-aware), and the other devoid of any awareness regarding either teammates or opponents (neighbor-unaware). Each policy for each team is trained independently with its corresponding setup.

The neighbor-aware team wins 86.4% of the matches against the neighbor-unaware team. In the snapshots in Fig. 6, we observe that robots in the fully aware team have learned collaborative strategies such as spreading out and passing balls to teammates (1 & 2), sometimes passing balls to their opponents and catch the ball when it bounces back (3 & 4).

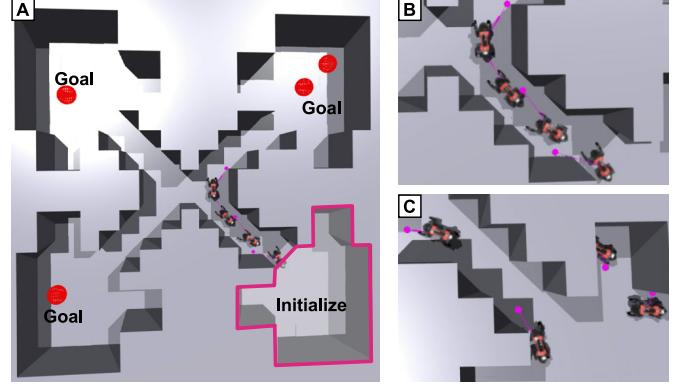


Fig. 7. Ablation study on GEE. (A) MRMG navigation test environment. Four robots are spawned in the bottom right corner, and the goals are sampled from the other three corners. (B) With GEE, when one robot finds a way out of the local minima, robots exhibit the leader-following behavior. (C) When GEE is replaced with a naive concatenation of features, the leader-following behavior does not emerge.

The neighbor-unaware team often collide with their teammates and does not exhibit such collaboration.

*2) Global Entity Encoder:* We now assess the role of GEE in enhancing the collaborative task performance. We replace the GEE with naively concatenating entity features and compared the success rate for MRMG navigation task (see Fig. 2A-1). Training the concatenation model is the same as training the GEE models as described in Section III-D1. The only difference is that we have to set a maximum entity number for each entity, and pad the concatenated features with zeros if the environment has fewer entities than the max number. We set the maximum entity numbers as those used when training the GEE model (4 robots, 4 goals). Note that the concatenation model cannot handle more than the max entity numbers but the GEE models can.

The GEE model yields a success rate of 96.7% on the task and an average completion time of 20.3 s, compared to 86.4% and 22.7 s for the concatenation. Fig. 7 shows their different behaviors in one example. In the cross-like corridor environment, the policy with GEE exhibits the leader-following behavior in Fig. 7B. The behavior is also shown in Fig. 3B. Once one of the robots finds a way out of the first room, the others follow the robot.

Our results show that the GEE significantly improves the performance of neighbor-aware policies by effectively leveraging global contextual information.

### D. Comparison to an Optimal Control Approach

To evaluate the optimality of our learned control policy, we conduct a comparative analysis with a centralized model predictive control (MPC) approach for the centroid control task illustrated in Fig. 8. In this task, the objective is to generate position and velocity targets for each cuboid to move the group's centroid to a goal position while avoiding collisions.

Our policy is trained with 1–6 cuboid agents in each environment and benchmarked against a simplified version of the planner by De Vincenti and Coros [13] as the baseline. The control task is formulated as a kinematic planar system

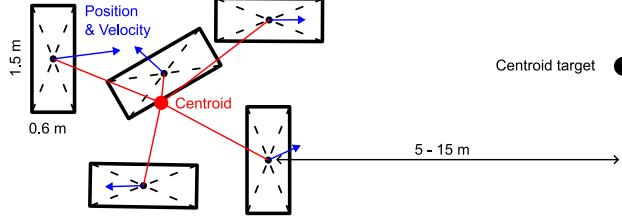


Fig. 8. Two-dimensional centroid control problem. A controller (Ours or MPC) controls the position and velocity of cuboids to let their centroid (red dot) track the target position. The cuboids are randomly spawned between 5–15 m away from the goal.

involving  $R$  cuboids. Both approaches solve for the desired velocity and position of each cuboid.

1) *MPC Formulation*: We approximate each robot  $i$  in a squad of  $R$  robots as a cuboid with a two-dimensional position  $\mathbf{p}_i \in \mathbb{R}^2$  and a yaw angle  $\psi_i \in \mathbb{R}$ . Given a discrete time horizon  $N \in \mathbb{N}$  and a step size  $\Delta t > 0$ , and having defined the  $i$ th robot's state and input sampled at time step  $k$  as  $\mathbf{x}_{i,k} := [\mathbf{p}_{i,k}^\top \ \psi_{i,k} \ \dot{\mathbf{p}}_{i,k}^\top \ \dot{\psi}_{i,k}]^\top$  and  $\mathbf{u}_{i,k} := [\dot{\mathbf{p}}_{i,k}^\top \ \dot{\psi}_{i,k}]^\top$ , respectively, we formulate our optimal control problem (OCP) as the following nonlinear programming problem:

$$\begin{aligned} \min_{\mathbf{X}, \mathbf{U}} \sum_{k=0}^{N-1} & \left[ \left\| \sum_i \mathbf{p}_{i,k}/R - \mathbf{c}^* \right\|_{1,\mathbf{W}_c} + \right. \\ & \left. \left\| \sum_i \dot{\mathbf{p}}_{i,k}/R - \dot{\mathbf{c}}^* \right\|_{\mathbf{W}_c}^2 + \right. \\ & \left. \sum_{i=0}^{R-1} \left\| \mathbf{x}_{i,k} - \mathbf{x}_{i,k}^* \right\|_{\mathbf{W}_x}^2 + \left\| \mathbf{u}_{i,k} \right\|_{\mathbf{W}_u}^2 \right] \quad (3a) \end{aligned}$$

$$\begin{aligned} \text{s.t. } & \mathbf{x}_{i,0} = \mathbf{x}_{i,m}, \forall i, \\ & \mathbf{x}_{i,k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k, \forall i, \forall k, \\ & \left\| \mathbf{p}_{i,k} - \mathbf{p}_{j,k} \right\| \geq 2\rho, \forall i, \forall j, \forall k, \\ & \left| b\dot{\mathbf{p}}_{i,k}^x \right| \leq v_{\max}^x, \forall i, \forall k, \\ & \left| b\dot{\mathbf{p}}_{i,k}^y \right| \leq v_{\max}^y, \forall i, \forall k, \\ & \left| \dot{\psi}_{i,k} \right| \leq \omega_{\max}, \forall i, \forall k, \end{aligned}$$

where  $\mathbf{c}^*$  and  $\dot{\mathbf{c}}^*$  are the reference centroid position and velocity, respectively, while  $\mathbf{x}_{i,k}^*$  is the reference state of the  $i$ th robot at time step  $k$ .  $\mathbf{W}_\square$  is a positive definite diagonal matrix for the weighted 1- and 2-norms  $\|\cdot\|_{1,\mathbf{W}_\square}$  and  $\|\cdot\|_{\mathbf{W}_\square}$ , respectively;  $\mathbf{x}_{i,m}$  is the  $i$ th robot's measured state. We integrate the input accelerations using a semi-implicit Euler method, which translates to the following system matrices:

$$\mathbf{A} := \begin{bmatrix} \mathbf{I}_3 & \Delta t \mathbf{I}_3 \\ \mathbf{0} & \mathbf{I}_3 \end{bmatrix}, \quad \mathbf{B} := \begin{bmatrix} \Delta t^2 \mathbf{I}_3 \\ \Delta t \mathbf{I}_3 \end{bmatrix}.$$

We provide more details about our MPC implementation in Appendix C.

2) *Results*: Table I shows the mean and standard deviation of three evaluation metrics for both our decentralized Multi-Entity RL policy and the centralized MPC planner on two setups: centroid control with 4 cuboids and 10 cuboids. We run 10 experiments with cuboids randomly positioned 5–15 m away from the goal and compare the final tracking errors, settling times, centroid travel distances, and solver time of

both methods on both setups. We remark that the 10-cuboid scenario is not seen during our policy training stage, as the training environments only consist of 1 to 6 cuboids.

Both approaches exhibit a very similar performance. With 4 cuboids, MPC leads by a narrow margin on the tracking error ( $\sim 1$  cm), but it has slightly longer settling times. In all setups, our RL policy showcases shorter centroid travel distances, suggesting more efficient goal-pursuit trajectories. The compared metrics indicate that our Multi-Entity RL policy can find near-optimal solutions in the considered scenario.

Our decentralized policy generalizes to unseen tasks involving 10 cuboids with minimal deterioration from the 4-cuboid setup, with  $\sim 3$  cm higher tracking errors and  $\sim 3.3$  s longer settling times. The increased solver times of the MPC approach highlight the limited scalability of conventional centralized methods. In contrast, our decentralized RL policy maintains nearly constant computation times regardless of the number of entities involved.

## V. CONCLUSION

We present a MARL framework using permutation invariant neural network encoders capable of processing various categories of entities in a wide range of robotic environments. We validated our approach by implementing and testing a complete system to control MAS with wheeled-legged quadrupeds. The system consists of decentralized multi-entity control policy and communication among robots using gRPC [26].

Our real-world MRMG navigation experiments with two robots demonstrated collaborative multi-goal navigation behaviors learned to optimize task performance without the use of heuristics. We observed intelligent sub-goal selection, navigation, and collision avoidance.

Furthermore, we conducted controlled experiments in simulation on three distinct multi-entity tasks, aiming to showcase the scalability and generalizability of our approach. Our experiments showed that the GEEs enables agents to process a flexible number of entities in execution time. GEEs enable each robot to collaborate with flexible numbers of teammates and prioritize suitable objects or goals. This resulted in improved overall performance compared to teams without the permutation invariant encoder structure or communication among robots. Additionally, we conducted a sensitivity analysis on GEEs, providing an evidence that our policy automatically adjusts its focus to different entities in the environment at each moment.

Moreover, we compared our policy to an existing optimal-control approach on a multi-agent navigation problem and showed that our policy learns a near-optimal solution in this scenario.

We plan to extend our research to encompass heterogeneous robots, such as robots equipped with different sensors or manipulators for specialized tasks. We hope this work could inspire future research on general purpose AI and robotics. Our goal is to advance robots beyond simple lab environments with constrained task configurations. We aim to enhance their capabilities to excel in diverse real-world environments with a wide range of entities and tasks.

TABLE I  
COMPARISON IN CENTROID CONTROL TASK.

	Tracking error (m)	Settling time to 0.1 m (s)	Centroid travel distance (m)	Solver time (ms)
<b>Ours 4 robots</b>	$0.0168 \pm 0.0085$	$4.70 \pm 1.48$	$10.11 \pm 3.03$	$0.51 \pm 0.03$
<b>MPC [13] 4 robots</b>	$0.0070 \pm 0.0030$	$5.35 \pm 0.92$	$10.38 \pm 1.90$	$9.61 \pm 0.29$
<b>Ours 10 robots</b>	$0.0478 \pm 0.0249$	$8.02 \pm 0.78$	$10.53 \pm 1.82$	$0.77 \pm 0.07$
<b>MPC [13] 10 robots</b>	$0.0044 \pm 0.0020$	$5.90 \pm 1.35$	$11.18 \pm 2.44$	$91.03 \pm 5.52$

## ACKNOWLEDGMENTS

We thank Yifan Liu for the support in implementing simulation environments for MRMG navigation training. We thank Alexander Reske and Turcan Tuna for helping with the real robot experiments.

This project was supported by the Swiss National Science Foundation through the National Centre of Competence in Digital Fabrication (NCCR dfab) and the European Research Council (Grants 852044, 101070596, and 101121321).

## REFERENCES

- [1] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning quadrupedal locomotion over challenging terrain,” *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.
- [2] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, “Learning to walk in minutes using massively parallel deep reinforcement learning,” in *Conference on Robot Learning*. PMLR, 2022, pp. 91–100.
- [3] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning robust perceptive locomotion for quadrupedal robots in the wild,” *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.
- [4] D. Hoeller, L. Wellhausen, F. Farshidian, and M. Hutter, “Learning a state representation and navigation in cluttered and dynamic environments,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5081–5088, 2021.
- [5] C. Chen, J. Frey, P. Arm, and M. Hutter, “Smug planner: A safe multi-goal planner for mobile robots in challenging environments,” *IEEE Robotics and Automation Letters*, 2023.
- [6] H. Kondoh and A. Kanezaki, “Multi-goal audio-visual navigation using sound direction map,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 5219–5226.
- [7] F. Wang and K. Hauser, “Stable bin packing of non-convex 3d objects with a robot manipulator,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8698–8704.
- [8] J. Mahler and K. Goldberg, “Learning deep policies for robot bin picking by simulating robust grasping sequences,” in *Conference on robot learning*. PMLR, 2017, pp. 515–524.
- [9] R. Li, A. Jabri, T. Darrell, and P. Agrawal, “Towards practical multi-object manipulation using relational reinforcement learning,” in *2020 ieee international conference on robotics and automation (icra)*. IEEE, 2020, pp. 4051–4058.
- [10] C. Yang, G. N. Sue, Z. Li, L. Yang, H. Shen, Y. Chi, A. Rai, J. Zeng, and K. Sreenath, “Collaborative navigation and manipulation of a cable-towed load by multiple quadrupedal robots,” *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 041–10 048, 2022.
- [11] S. Sukhbaatar, R. Fergus *et al.*, “Learning multiagent communication with backpropagation,” *Advances in neural information processing systems*, vol. 29, 2016.
- [12] Y. Lee, J. Yang, and J. J. Lim, “Learning to coordinate manipulation skills via skill behavior diversification,” in *International conference on learning representations*, 2019.
- [13] F. De Vincenti and S. Coros, “Centralized model predictive control for collaborative loco-manipulation,” *Proceedings of Robotics: Science and System XIX*, p. 050, 2023.
- [14] K. A. Hamed, V. R. Kamidi, A. Pandala, W.-L. Ma, and A. D. Ames, “Distributed feedback controllers for stable cooperative locomotion of quadrupedal robots: A virtual constraint approach,” in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 5314–5321.
- [15] Y. Ji, B. Zhang, and K. Sreenath, “Reinforcement learning for collaborative quadrupedal manipulation of a payload over challenging terrain,” in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 899–904.
- [16] O. Nachum, M. Ahn, H. Ponte, S. Gu, and V. Kumar, “Multi-agent manipulation via locomotion using hierarchical sim2real,” *arXiv preprint arXiv:1908.05224*, 2019.
- [17] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, “Emergent tool use from multi-agent autocurricula,” *arXiv preprint arXiv:1909.07528*, 2019.
- [18] H. Ha, J. Xu, and S. Song, “Learning a decentralized multi-arm motion planner,” *arXiv preprint arXiv:2011.02608*, 2020.
- [19] J. Yang, A. Nakhaei, D. Isele, K. Fujimura, and H. Zha, “Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning,” *arXiv preprint arXiv:1809.05188*, 2018.
- [20] Q. Xu, J. Li, S. Koenig, and H. Ma, “Multi-goal multi-agent pickup and delivery,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 9964–9971.
- [21] S. Zimmermann, G. Hakimifard, M. Zamora, R. Poranne, and S. Coros, “A multi-level optimization framework for simultaneous grasping and motion planning,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2966–2972, 2020.
- [22] J. Envall, R. Poranne, and S. Coros, “Differentiable task assignment and motion planning,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 2049–2056.
- [23] F. Kennel-Maushart, R. Poranne, and S. Coros, “Interacting with multi-robot systems via mixed reality,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 633–11 639.
- [24] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.
- [25] M. Bjelonic, C. D. Bellilcoso, Y. de Viragh, D. Sako, F. D. Tresoldi, F. Jenelten, and M. Hutter, “Keep rollin’—whole-body motion control and planning for wheeled quadrupedal robots,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2116–2123, 2019.
- [26] gRPC, “About grpc,” <https://grpc.io/about/>, 2023.
- [27] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph cnn for learning on point clouds,” *ACM Transactions on Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.
- [28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [29] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh,

- “Set transformer: A framework for attention-based permutation-invariant neural networks,” in *International conference on machine learning*. PMLR, 2019, pp. 3744–3753.
- [30] Y. Tang and D. Ha, “The sensory neuron as a transformer: Permutation-invariant neural networks for reinforcement learning,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 22 574–22 587, 2021.
- [31] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, “Mean field multi-agent reinforcement learning,” in *International conference on machine learning*. PMLR, 2018, pp. 5571–5580.
- [32] Y. Li, L. Wang, J. Yang, E. Wang, Z. Wang, T. Zhao, and H. Zha, “Permutation invariant policy optimization for mean-field multi-agent reinforcement learning: A principled approach,” *arXiv preprint arXiv:2105.08268*, 2021.
- [33] R. Zhang, G. Chen, J. Hou, Z. Li, and A. Knoll, “Pipo: Policy optimization with permutation-invariant constraint for distributed multi-robot navigation,” in *2022 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 2022, pp. 1–7.
- [34] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, “Tossingbot: Learning to throw arbitrary objects with residual physics,” *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1307–1319, 2020.
- [35] Y. Huang, A. Conkey, and T. Hermans, “Planning for multi-object manipulation with graph neural network relational classifiers,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1822–1829.
- [36] S. Shaw, E. Wenzel, A. Walker, and G. Sartoretti, “Formic: Foraging via multiagent rl with implicit communication,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4877–4884, 2022.
- [37] J. Wu, X. Sun, A. Zeng, S. Song, S. Rusinkiewicz, and T. Funkhouser, “Spatial intention maps for multi-agent mobile manipulation,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8749–8756.
- [38] F. Schmalstieg, D. Honerkamp, T. Welschehold, and A. Valada, “Learning long-horizon robot exploration strategies for multi-object search in continuous action spaces,” in *The International Symposium of Robotics Research*. Springer, 2022, pp. 52–66.
- [39] J. Jiang and Z. Lu, “Learning attentional communication for multi-agent cooperation,” *Advances in neural information processing systems*, vol. 31, 2018.
- [40] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, “Mastering the game of go without human knowledge,” *nature*, vol. 550, no. 7676, pp. 354–359, 2017.
- [41] S. Liu, G. Lever, Z. Wang, J. Merel, S. Eslami, D. Hennes, W. M. Czarnecki, Y. Tassa, S. Omidshafiei, A. Abdolmaleki *et al.*, “From motor control to team play in simulated humanoid football,” *arXiv preprint arXiv:2105.12196*, 2021.
- [42] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, M. Wulfmeier, J. Humplik, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner *et al.*, “Learning agile soccer skills for a bipedal robot with deep reinforcement learning,” *arXiv preprint arXiv:2304.13653*, 2023.
- [43] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, “Emergent complexity via multi-agent competition,” *arXiv preprint arXiv:1710.03748*, 2017.
- [44] T. Ming, “Multi-agent reinforcement learning: Independent versus cooperative agents,” in *Proceedings of the Tenth International Conference on Machine Learning (ICML 1993), San Francisco, CA, USA*, 1993, pp. 330–337.
- [45] T. Chu, J. Wang, L. Codecà, and Z. Li, “Multi-agent deep reinforcement learning for large-scale traffic signal control,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1086–1095, 2019.
- [46] C. S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. Torr, M. Sun, and S. Whiteson, “Is independent learning all you need in the starcraft multi-agent challenge?” *arXiv preprint arXiv:2011.09533*, 2020.
- [47] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, “The surprising effectiveness of ppo in cooperative multi-agent games,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 611–24 624, 2022.
- [48] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” *Advances in neural information processing systems*, vol. 30, 2017.
- [49] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls *et al.*, “Value-decomposition networks for cooperative multi-agent learning,” *arXiv preprint arXiv:1706.05296*, 2017.
- [50] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, “Monotonic value function factorisation for deep multi-agent reinforcement learning,” *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 7234–7284, 2020.
- [51] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, “Qplex: Duplex dueling multi-agent q-learning,” *arXiv preprint arXiv:2008.01062*, 2020.
- [52] J. Wang, D. Ye, and Z. Lu, “More centralized training, still decentralized execution: Multi-agent conditional policy factorization,” *arXiv preprint arXiv:2209.12681*, 2022.
- [53] Y. Zhou, S. Liu, Y. Qing, K. Chen, T. Zheng, Y. Huang, J. Song, and M. Song, “Is centralized training with decentralized execution framework centralized enough for marl?” *arXiv preprint arXiv:2305.17352*, 2023.
- [54] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu *et al.*, “Swarm of micro flying robots in the wild,” *Science Robotics*, vol. 7, no. 66, p. eabm5954, 2022.
- [55] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, “Learning to communicate with deep multi-agent reinforcement learning,” *Advances in neural information processing systems*, vol. 29, 2016.
- [56] A. Singh, T. Jain, and S. Sukhbaatar, “Learning when to communicate at scale in multiagent cooperative and competitive tasks,” *arXiv preprint arXiv:1812.09755*, 2018.
- [57] Y. Niu, R. R. Paleja, and M. C. Gombolay, “Multi-agent graph-attention communication and teaming.” in *AAMAS*, 2021, pp. 964–973.
- [58] A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau, “Tarmac: Targeted multi-agent communication,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 1538–1546.
- [59] C. Yu, X. Yang, J. Gao, J. Chen, Y. Li, J. Liu, Y. Xiang, R. Huang, H. Yang, Y. Wu *et al.*, “Asynchronous multi-agent reinforcement learning for efficient real-time multi-robot cooperative exploration,” *arXiv preprint arXiv:2301.03398*, 2023.
- [60] J. Lee, M. Bjelonic, and M. Hutter, “Control of wheeled-legged quadrupeds using deep reinforcement learning,” in *Climbing and Walking Robots Conference*. Springer, 2022, pp. 119–127.
- [61] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart *et al.*, “Cerberus in the darpa subterranean challenge,” *Science Robotics*, vol. 7, no. 66, p. eabp9742, 2022.
- [62] P.-W. Chou, D. Maturana, and S. Scherer, “Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution,” in *International conference on machine learning*. PMLR, 2017, pp. 834–843.
- [63] I. G. Petrazzini and E. A. Antonelo, “Proximal policy optimization with continuous bounded action space via the beta distribution,” in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2021, pp. 1–8.
- [64] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [65] F. De Vincenti and S. Coros, “Ungar - a c++ framework for

- real-time optimal control using template metaprogramming,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 6297–6303.
- [66] W. Jakob, J. Rhinelander, and D. Moldovan, “pybind11 – seamless operability between c++11 and python,” 2017, <https://github.com/pybind/pybind11>.
- [67] protobuf.dev, “Protocol buffers documentation,” <https://protobuf.dev/>, 2023.

## APPENDIX

### A. Hierarchical Policy

Hierarchical policies have been widely applied on complex robotic tasks [10, 12, 16], as modularized policies provide easier control over desired behaviors at each task level. We use a similar way to train hierarchical policies as in [16]. First, a locomotion policy is trained according to the work by Lee et al. [60]. This policy takes body velocity commands and executes joint-level control on individual robots. The locomotion policy is treated as the low-level policy and fixed when training higher level policies. In this project, both the box packing task and the soccer task use two-level policy structure: the low-level policy is the locomotion policy, and the high-level policies output body velocity commands to complete specific multi-agent tasks. The MRMG navigation task uses three-level policy structure, where the low-level policy is the same locomotion policy. Similar to the high-level policies of the other two tasks, the mid-level policy in MRMG navigation outputs body velocity commands to control individual robots towards a shorter goal point (0–2 m from ego robot) without observing neighboring robots, and the high-level policy outputs these shorter goal points to navigate each robot to flexible numbers of far goal points (15–30 m) in a MAS. In fact, the low-level policy is the same for all three tasks, as the desired locomotion behaviors are the same in such low level. This is the advantage of modularized hierarchical policy structure, as some parts of the policy can be shared among different tasks, thus greatly reducing the time to train robots from scratch.

### B. Bounded Action Space using Beta Distribution

For stochastic policy gradient methods such as PPO [64], a practical limitation is that the Gaussian distribution has infinite support over the action space. This creates challenges to constrain policy outputs for robots’ physical limitations. For example, the maximum speed of our robots is 2 m/s in longitudinal directions, 1 m/s in lateral directions and 1.5 rad/s about vertical axis. Therefore, the body velocity commands as action outputs from our high-level policies must be within these ranges, which cannot be satisfied by Gaussian distribution. Inspired by [62] and [63], we replace Gaussian distribution with Beta distribution as the stochastic action distribution function in our PPO algorithm [64]. The benefits of Beta distribution is that the output action distribution is bounded between 0 and 1, so it can be linearly scaled to any bounded action range without suffering the loss of sampling accuracy. The policy network outputs parameters  $\alpha$  and  $\beta$  for Beta action distribution:

TABLE II  
MPC PARAMETERS

$N$	30
$\Delta t$	0.03
$\rho$	0.8
$\mathbf{W}_c$	diag(10, 10)
$\mathbf{W}_{\dot{c}}$	diag(0.6, 0.6)
$\mathbf{W}_x$	$10^{-2}$ diag(0, 0, 1000, 4, 4, 4)
$\mathbf{W}_u$	$10^{-5}$ diag(1, 1, 1000)
$v_{\max}^x$	2
$v_{\max}^y$	1
$\omega_{\max}$	1.5

$$\text{Beta}(x; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1} \quad (4)$$

The shape of the distribution function depends only on the two parameters  $\alpha$  and  $\beta$ , and  $B(\alpha, \beta)$  is called the beta function, which also depends on  $\alpha$  and  $\beta$  and ensure the total probability equals to 1.

Stochastic actions are sampled from the Beta distribution during training to encourage exploration. During policy execution, we take the distribution mean  $\frac{\alpha}{\alpha+\beta}$  as the deterministic action.

### C. MPC Baseline Implementation Details

We implement a simplified MPC planner based on the work by De Vincenti and Coros [13] and delineate its formulation in Section IV-D1. In this section, we lay out more implementation details for our MPC controller to complete the multi-agent problem in Section IV-D.

To prevent robot collisions, we enforce a minimum distance of  $2\rho$  between all pairs of robots ( $\rho > 0$ ). The quantities  $b\dot{\mathbf{p}}_{i,k}^x$  and  $b\dot{\mathbf{p}}_{i,k}^y$  represent the  $x$ - and  $y$ -components of the  $i$ th robot’s linear velocity at time step  $k$  expressed in its base frame. Finally,  $v_{\max}^x$  and  $v_{\max}^y$  denote the maximum forward and sideways linear velocities, while  $\omega_{\max}$  is the maximum yaw rate. We remark that we adopted a 1-norm for the centroid tracking cost in (3a) to prevent steep gradients for distant goal positions  $\mathbf{c}^*$  that would cause numerical issues.

We implement our MPC in C++ using Ungar [65] and send the optimized trajectories to our low-level PD controller through Python bindings [66]. Specifically, the output planar linear velocities and yaw rates are converted into force and torque signals to drive each individual cuboid in the simulator. We use the parameters listed in Table II for all our experiments.

### D. gRPC Communication framework

We implemented a multi-robot communication framework in ROS using the gRPC framework [26]. gRPC uses Protocol Buffers [67] to serialize messages between a server and its clients distributed in different machines. In our case, the server collects the most recent messages from all clients periodically and sends them to any client that makes the request. Each robot initiates a client at the beginning, which constantly sends out its own message to the server and requests other robots’

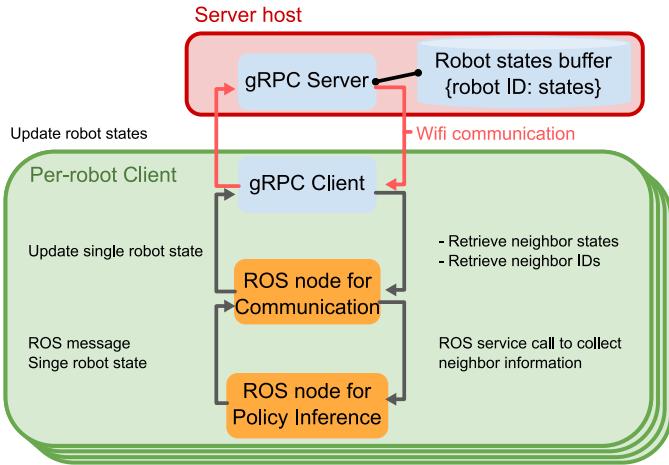


Fig. 9. gRPC communication framework

TABLE III  
REWARDS

## MRMG Navigation

Reward	Description	Scale
Termination	+1.0 if game success	10.0
Distance to Goal	$+ \exp(-(distance to unreached goals)^2)$	5.0
Motion bonus	$+ \text{clip}(\text{moving speed}, 0.0, 1.0)$	1.0
Neighbor distance	-1.0 if too close (<1 m)	1.0
Collision	-1.0 if robot collides	2.0

## Box Packing

Reward	Description	Scale
Termination	+1.0 if game success	5.0
Progress	$+1.0 (N_{\text{completed boxes}} / N_{\text{total box count}})$	0.25
Box velocity	$+1.0 \sum_{\text{boxes}} (v_{\text{box}} \cdot \text{direction to goal})$	0.1
Box position	$+1.0 \sum_{\text{boxes}} \exp(-(p_{\text{box}} - p_{\text{goal}})^2)$	0.5
Neighbor distance	-1.0 if too close (<1 m)	1.0

## Soccer

Reward	Description	Scale
Termination	+1.0 if wins, -1.0 if loses	5.0
Ball velocity	$+1.0 (v_{\text{ball}} \cdot \text{direction to goal})$	1.0
Ball position	$+1.0 \exp(-(p_{\text{ball}} - \text{goal position})^2)$	0.25
Neighbor distance	-1.0 if too close (<1 m)	1.0

messages when needed by the policy net. The server-client communication framework is illustrated in Fig. 9. A gRPC server contains a message buffer with robot ID-message pairs. A robot ID is a unique number for each robot and a message is the agent state information passed to the GEE. Every gRPC client subscribes to the server host at the beginning and creates gRPC Stubs to call service methods for message exchange with other clients. Inside each robot, a gRPC client is instantiated by the robot's ROS communication node, which is also responsible for transferring messages between the gRPC client and the navigation planner through standard ROS communication protocol.

The gRPC communication pipeline is simple to use in real-world applications and can connect a flexible number of robots. When a new robot is activated and subscribed to the server host, its unique ID is saved in the message buffer in the server. When a robot requires communication with other robots, it can request other robots' messages with their IDs. After the robot executes its action and arrives at a new state,

its message gets automatically updated in the message buffer.

It is worth noting that the server host can be any computer with the same internet connection as the clients, including any one of the clients. In our real-world experiments, we use a separated computer as the server host.

TABLE IV  
ROBOT OBSERVATION IN ALL POLICIES

Policy	Type	Observation	Dimension
Locomotion Low-Level	Proprioceptive	Velocity command Projected gravity Body linear velocity Body angular velocity Joint position (excluding 4 wheel joints) Joint position error (excluding 4 wheel joints) for 3 half-LL time steps Joint velocity for 3 half-LL time steps Joint position target for 2 LL time steps	3 3 3 3 12 $3 * 12$ $3 * 16$ $2 * 16$
	Exteroceptive	foot height scan around 4 feet	$4 * 52$
MRMG Navigation Mid-Level	Proprioceptive	Same as Low-Level	140
	Exteroceptive	Body height scan	600
	Short Navigation State	Short goal position for 10 ML time steps	$10 * 6$
MRMG Navigation High-Level	Proprioceptive	Same as Low-Level	140
	Exteroceptive	Same as Mid-Level	600
	Short Navigation State	Same as Mid-Level	60
	Common Neighbor States (GEE)	Neighbor velocity command for 20 HL time steps Neighbor position for 20 HL time steps Neighbor orientation for 20 HL time steps Neighbor linear velocity for 20 HL time steps Neighbor angular velocity for 20 HL time steps Neighbor projected gravity for 20 HL time steps	$20 * 3 * \#N$ $20 * 3 * \#N$ $20 * 4 * \#N$ $20 * 3 * \#N$ $20 * 3 * \#N$ $20 * 3 * \#N$
	Task Neighbor State (GEE)	Neighbor short goal for 5 HL time steps	$5 * 6 * \#N$
	Long Navigation State (GEE)	Long Goal position for 20 HL time steps Long Goal reach mask (0: not reached; 1: reached)	$20 * 3 * \#G$ $1 * \#G$
Box Packing High-Level	Proprioceptive	Same as Low-Level	140
	Exteroceptive	Same as Mid-Level	430
	Common Neighbor States (GEE)	Same as above	$20 * 19 * \#N$
	Box State (GEE)	Box position for 20 HL time steps Box in-range mask (0: outside packing range; 1: inside packing range)	$20 * 3 * \#B$ $1 * \#B$
Soccer High-Level	Proprioceptive	Same as Low-Level	140
	Exteroceptive	Same as Mid-Level	430
	Teammate State (GEE)	Common neighbor states team mask: 0: teammate; 1: opponent	$20 * 19 * \#T$ $1 * \#N$
	Opponent State (GEE)	Common neighbor states team mask: 0: teammate; 1: opponent	$20 * 19 * \#O$ $1 * \#N$

#N: number of neighbors. #G: number of goals. #B: number of boxes. #T: number of teammates. #O: number of opponents.  
GEE: observation goes through the GEE