Spatial Intention Maps for Multi-Agent Mobile Manipulation

Jimmy Wu\textsuperscript{1,2}, Xingyuan Sun\textsuperscript{1}, Andy Zeng\textsuperscript{2}, Shuran Song\textsuperscript{3}, Szymon Rusinkiewicz\textsuperscript{1}, Thomas Funkhouser\textsuperscript{1,2}

Abstract—The ability to communicate intention enables decentralized multi-agent robots to collaborate while performing physical tasks. In this work, we present spatial intention maps, a new intention representation for multi-agent vision-based deep reinforcement learning that improves coordination between decentralized mobile manipulators. In this representation, each agent’s intention is provided to other agents, and rendered into an overhead 2D map aligned with visual observations. This synergizes with the recently proposed spatial action maps framework, in which state and action representations are spatially aligned, providing inductive biases that encourage emergent cooperative behaviors requiring spatial coordination, such as passing objects to each other or avoiding collisions. Experiments across a variety of multi-agent environments, including heterogeneous robot teams with different abilities (lifting, pushing, or throwing), show that incorporating spatial intention maps improves performance for different mobile manipulation tasks while significantly enhancing cooperative behaviors.

I. INTRODUCTION

Multi-agent systems require a mutual understanding of intention in order to coordinate amongst each other and perform collaborative tasks. This is particularly important for decentralized robots that share the same physical space while interacting in complex environments under partial observability and limited-bandwidth communication. Applications of such systems include foraging [1], hazardous waste cleanup [2], object transportation [3], [4], and search and rescue [5]. The capacity to coordinate by conveying intentions can reduce collisions between robots (which can be both costly and disabling) while improving the efficiency with which tasks are completed.

Intention-aware multi-agent learning methods have shown promising success—from hide and seek [6] to Starcraft [7]—albeit largely in simulated domains such as video games. These systems often communicate intentions via sharing high-level state information, or by condensing intentions into a low-dimensional embedding such as the coordinates of a destination. Both of these approaches, however, exclude spatial structure and are thus a poor fit for reinforcement learning (RL) with convolutional neural networks, which has become a dominant strategy for learning with visual input.

In this work we propose spatial intention maps (Fig. 1), a new intention representation for multi-agent deep RL. In our approach, each agent’s intention (i.e., its most recently selected action) is rendered into a 2D map (first inset in Fig. 1) aligned with a state map derived from visual observations. The intention and state maps are input into a fully convolutional network to predict Q-values for a dense set of spatial actions (second inset in Fig. 1). At each step, the best action (e.g., navigate to a location) is selected for execution. Since spatial intention maps are encoded as images pixelwise aligned with state and action maps [8], we expect that fully convolutional networks can leverage them more efficiently than other intention encodings. Indeed, we find during experiments that encoding intentions spatially in this way provides a signal that encourages Q-functions to learn emergent collaborative behaviors, ranging from collision avoidance to spatial coordination and task specialization.

We demonstrate the efficacy of spatial intention maps across a variety of multi-agent environments, including robots with different abilities (lifting, pushing, or throwing) that are trained on different tasks (foraging and search and rescue). We observe that using spatial intention maps leads to better performance and significant gains in collaborative behavior, including fewer collisions, coordination around bottlenecks such as doorways, and improved distribution of agents throughout an environment. We also show that our learned policies (trained in simulation) are able to generalize to a real world setting. For qualitative video results and code, please see the supplementary material at https://spatial-intention-maps.cs.princeton.edu.

II. RELATED WORK

Multi-robot systems. The earliest works in multi-robot systems date back to the 1980s [9]. These works have studied a wide variety of challenges in multi-robot systems, such as architecture, communication, team heterogeneity, and learning. Typical applications areas include foraging [1],

\textsuperscript{1}Princeton University, \textsuperscript{2}Google, \textsuperscript{3}Columbia University
Multi-agent reinforcement learning. There is a wide body of work in multi-agent reinforcement learning (MARL) [11], [12], [13]. One of the earliest works in this area was [14], in which multiple independent Q-learning agents were trained in a cooperative setting. More recently, with the rising popularity of deep reinforcement learning approaches such as DQN [15], many works have proposed ways to extend these methods to multi-agent settings [16]. These include addressing nonstationarity through modifications to experience replay [17], [18] or training with a centralized critic [19], [20], or improving credit assignment through decomposing the value function [21], [22], [23]. Most of these approaches assume access to the full state of the environment, though a few use raw visual data [24], [25]. In our approach, we learn directly from visual data reconstructed from partial observations of the environment.

Learning-based multi-robot systems. While there is a vast amount of literature on multi-agent reinforcement learning, most works focus on games and simulation environments. There is comparatively less work applying multi-agent learning to robotic systems. Some earlier works have applied Q-learning to applications such as box pushing [26], foraging [27], soccer [28], [29], and multi-target observation [30]. More recently, [31] and [32] have used macro-actions [33] with DQN to enable multiple robots to asynchronously reason and execute actions. Other works have investigated multi-agent tasks such as navigation [34] or cooperative manipulation [35]. As with the MARL works previously described, these typically assume access to high-level state information such as the positions of relevant objects. In contrast, we learn directly from visual data, allowing agents to automatically learn to detect relevant visual features.

Multi-robot communication. Many works in multi-robot systems have studied the effect of communication on improving team performance [36], [37], [38]. These comparisons typically span a continuum from no communication (implicit), to passive observation of teammates’ states, to direct communication (explicit). The same continuum of communication has been studied for multi-agent learning [14], [39], [40]. More recent works in multi-agent learning have studied other ways to communicate, such as learning what to communicate [41], [42], [43] or learning to model the intentions of other agents [44], [45], [46], [47], [34]. We similarly explore this communication continuum, but in our approach, communicated intentions are spatially encoded and aligned with the state and action representations. This allows agents to reason about their teammates’ intentions in the same domain as their own observations and actions.

III. Method

We study spatial intention maps in the context of two tasks. The first is a foraging task, in which a team of robots work together to move all objects in an environment to a target receptacle in the upper right corner (see Fig. 2 for an example environment). The second is a search and rescue task, in which a team of robots need to find and “rescue” all objects in an environment. We train our policies in a PyBullet [48] simulation environment and run them on real robots using a sim-to-real mirroring setup (see Sec. IV-D for more details).

A. Reinforcement Learning Formulation

We model our task as a Markov decision process (MDP) from the perspective of each individual agent, using single-agent state and action spaces. We train our policies using double deep Q-learning (DQN) [15], [49], giving individual rewards to agents during training. This formulation of multi-agent RL is similar to independent Q-learning [14], but we share policies between agents of the same type during training. Execution is decentralized—trained policies are run independently and asynchronously on each agent.

As widely discussed in the multi-agent literature, a common challenge with training multiple agents concurrently in the same environment is nonstationarity caused by other agents having policies that are constantly evolving. In our system, we use spatial intention maps (see Fig. 2) to directly encode the intentions of other agents into the state representation given to DQN, which helps to alleviate the uncertainty in other agents’ behaviors. Please see the supplementary material for more training details.

B. State Representation

Our state representation consists of a local overhead map, along with a series of auxiliary local maps with additional information useful to the agent when making its decisions [8].
To construct these local maps, each agent independently builds up its own global map of the environment using online mapping. Each agent has a simulated forward-facing RGB-D camera and captures partial observations of the environment, which are integrated with previous views to gradually build up a global overhead map. Agents must therefore learn policies that can deal with outdated information and seek out unexplored parts of the environment.

Each time an agent chooses a new action, it crops out a local map from its global map to generate a new state representation. These local maps are oriented such that the agent itself is in the center and facing up (see Fig. 2 for an example). The state representation consists of the following local maps, each of which is represented as an overhead image: (i) an environment map, (ii) an agent map encoding the agent’s own state and the observed states of other agents, (iii) maps encoding shortest path distances to the receptacle and from the agent, and (iv) the spatial intention map, which we describe in Sec. III-D. The agent map spatially encodes the state of every agent, which includes each agent’s pose, as well as whether they are carrying an object (if applicable).

C. Action Representation

We use spatial action maps [8] as our action representation. The action space is represented as a pixel map, spatially aligned with the state representation, in which every pixel represents the action that navigates the agent to the corresponding location in the environment (see Q-value map in Fig. 2). For agents that can execute an end effector action (such as lifting or throwing), we augment the action space with a second spatial channel, which represents navigating to the corresponding location and then attempting to perform the end effector action. (Note that figures in this paper only show one relevant action channel for clarity.) We execute the action corresponding to the argmax across all channels in the action space, employing high-level motion primitives implemented with low-level control. For movement, the primitive attempts to proceed to the specified location along the shortest path, computed using the agent’s own occupancy map. For end effector actions, the primitives attempt to lock onto an object before operating the end effector.

D. Spatial Intention Maps

Spatial intention maps provide an encoding of other agents’ intentions in the form of a map aligned with the state and action representations. This map-based encoding of intentions is key to our approach, as it allows us to train a fully convolutional deep residual Q-network [50], [51] that maps pixels representing states (and intentions) to a Q-value map that is pixelwise aligned with the state. This type of computation (dense pixelwise prediction using fully convolutional networks) has been shown to be effective for many visual tasks (e.g., semantic segmentation). In our case, it is a good fit because encoding intentions into the spatial domain allows deep Q-networks to reason about intentions in the same domain as the state representation.

Agents in our setup use decentralized, asynchronous execution, which means that whenever an agent is choosing a new action, all of the other agents in the environment will be in motion, executing their most recently selected action. We encode these in-progress actions spatially as rasterized paths in a spatial intention map. Intended paths are encoded using a linear ramp function, with a value of 1 at the executing agent’s current location, and dropping off linearly along the path (see Fig. 2 for an example). A lower value at a point on a path indicates a longer period of time before the executing agent will reach that point. This information enables more fine-grained reasoning about time and distance (e.g., the agent over there intends to come here, but is still quite a distance away).

The bandwidth requirements for spatial intention maps during decentralized execution are low, as agents do not communicate maps (images) directly. Instead, they broadcast intentions as lists of \((x, y)\) coordinates (waypoints of intended path). Then, whenever an agent is selecting a new action, it locally renders the most recently received intentions (paths) into an up-to-date spatial intention map.

IV. EXPERIMENTS

We evaluate spatial intention maps with decentralized multi-robot teams on both foraging and search and rescue tasks across a wide range of simulation environments. The goals of these experiments are to investigate (i) whether learning with spatial intention maps is better than without, and (ii) how learning with spatial intention maps compares with alternative representations of intention. We also test our policies, which were trained in simulation, on real robots.

A. Experimental Setup

We investigate two tasks: (i) foraging: where a team of robots must collaborate to move a set of objects into a receptacle (objects are removed from the environment when they enter the receptacle), and (ii) search and rescue: where a team of robots must navigate to objects scattered in an environment and “rescue” them (objects are removed after contact by a robot). The two tasks are trained in the same way for the most part. Robots receive a success reward (+1.0) whenever an object is removed from the environment, and receive penalties for collisions with obstacles (-0.25) or other agents (-1.0). In the foraging task, they receive a distance-based partial reward/penalty for moving an object closer to or further from the receptacle. The lifting robot is also penalized (-0.25) for dropping objects outside the receptacle.

We experiment with four types of robots (Fig. 3): (i) a lifting robot that can pick up objects and carry them, (ii) a
pushing robot that can push objects, (iii) a throwing robot that can throw objects backwards, and (iv) a rescue robot that can mark an object as “rescued” upon contact. Teams can be homogeneous (4 of the same) or heterogeneous combinations (2 robot types with 2 per type). For all robots, an action consists of moving to a target location and optionally executing an end effector action (lifting or throwing).

We test the robot teams in six different environments (see Fig. 4). In empty environments with no obstacles, robots need to coordinate to avoid colliding while going towards the receptacle. In environments with obstacles, there are doorways and tunnels wide enough for only one robot to pass through, requiring coordination to avoid obstructing one another. We initialize robots, objects, and obstacles (dividers and walls) in random configurations at the start of each episode. Small environments contain 10 objects, and large ones contain 20. For the SmallDivider, LargeDoors, and LargeTunnels environments, robots and objects are initialized in opposite sides of the room, requiring the robots to bring all objects through doorways or tunnels to reach the receptacle.

For each episode, our evaluation metric is the total number of objects gathered after a fixed time cutoff, which measures the efficiency of a team (higher is better). The cutoff is the time at which the most efficient policy places the last object into the receptacle, and is different for every unique combination of robot team and environment, but is kept consistent between methods. We evaluate a trained policy by averaging its performance over 20 test episodes. For each method, we train five policies and report the mean and standard deviation of those averages across all five. Between evaluation runs, randomly initialized environments are kept consistent using random seeds (training is not seeded).

B. Experimental Results

In this section, we investigate whether spatial intention maps are useful by comparing the performance of multi-robot teams trained with and without them.

Foraging task. We first investigate performance on the foraging task with teams of 4 lifting robots (4L) or 4 pushing robots (4P). The corresponding rows in Tab. I show a comparison of teams trained with and without spatial intention maps. The results suggest that teams trained with spatial intention maps (Ours column) are able to learn the foraging task better. The biggest quantitative differences occur in the larger and more complex environments—e.g., the teams of lifting robots pick up an average of over 19 objects in the large environments with spatial intention maps, and only 11 to 17 without. However, the differences are significant in all settings.

We conjecture one reason for the differences is that robots without intention maps tend to move conservatively back and forth to minimize collisions with nearby robots. This is particularly evident when multiple robots are trying to approach the same object, or trying to put objects into the receptacle at the same time. For example, they may all approach the receptacle, observe that others are also moving forward, then all preemptively back away. In contrast, with intention maps, each robot is able to consider the intentions of other robots in order to choose actions that do not conflict.

By inspecting output Q-value maps, we can validate that the spatial intention maps are being used in this way. Fig. 5 examines a pair of scenarios requiring coordination between robots to pass through a shared doorway. We find that the predicted Q-value maps for each robot assign higher Q-values to actions that are compatible with what the other robot intends to do.

We observe several interesting emergent strategies when training with intention maps. For example, in the SmallDivider environment, both lifting and pushing robots consistently learn to move single file in a clockwise circle around the center divider (Fig. 6). In this emergent pattern, the robots move leftward through the bottom opening to find more objects, and rightward through the top opening to carry objects towards the receptacle. By maintaining oneway traffic in each opening, the robots avoid having to pause and coordinate passing through an opening from opposite directions. Without intention maps, we observe that this pattern still occasionally emerges, but is difficult to sustain.

<table>
<thead>
<tr>
<th>Robots</th>
<th>Environment</th>
<th>Ours</th>
<th>No intention maps</th>
</tr>
</thead>
<tbody>
<tr>
<td>4L</td>
<td>SmallEmpty</td>
<td>9.54 ± 0.16</td>
<td>7.92 ± 0.86</td>
</tr>
<tr>
<td></td>
<td>SmallDivider</td>
<td>9.44 ± 0.46</td>
<td>8.07 ± 0.75</td>
</tr>
<tr>
<td></td>
<td>LargeEmpty</td>
<td>18.86 ± 0.85</td>
<td>15.58 ± 3.80</td>
</tr>
<tr>
<td></td>
<td>LargeDoors</td>
<td>19.57 ± 0.25</td>
<td>13.96 ± 2.32</td>
</tr>
<tr>
<td></td>
<td>LargeTunnels</td>
<td>19.00 ± 0.47</td>
<td>11.89 ± 5.96</td>
</tr>
<tr>
<td></td>
<td>LargeRooms</td>
<td>19.52 ± 0.38</td>
<td>16.56 ± 1.53</td>
</tr>
<tr>
<td>4P</td>
<td>SmallEmpty</td>
<td>9.51 ± 0.20</td>
<td>8.73 ± 0.54</td>
</tr>
<tr>
<td></td>
<td>SmallDivider</td>
<td>9.50 ± 0.24</td>
<td>8.40 ± 0.78</td>
</tr>
<tr>
<td></td>
<td>LargeEmpty</td>
<td>19.51 ± 0.53</td>
<td>18.86 ± 0.72</td>
</tr>
<tr>
<td>2L+2P</td>
<td>LargeEmpty</td>
<td>19.52 ± 0.17</td>
<td>16.51 ± 4.27</td>
</tr>
<tr>
<td></td>
<td>LargeDoors</td>
<td>19.55 ± 0.18</td>
<td>17.44 ± 0.63</td>
</tr>
<tr>
<td></td>
<td>LargeRooms</td>
<td>19.51 ± 0.24</td>
<td>18.51 ± 0.75</td>
</tr>
<tr>
<td>2L+2T</td>
<td>LargeEmpty</td>
<td>19.51 ± 0.67</td>
<td>12.46 ± 4.34</td>
</tr>
<tr>
<td></td>
<td>LargeDoors</td>
<td>19.50 ± 0.45</td>
<td>6.21 ± 4.12</td>
</tr>
</tbody>
</table>

L = lifting, P = pushing, T = throwing
Fig. 5. **Coordinating to go through doorways.** In these scenarios, the current robot (center) is choosing a new action, while the other robot (top) is already moving. The other robot is headed towards the further (right) doorway in scenario 1 and the closer (left) doorway in scenario 2. In both scenarios, the Q-value map suggests the current robot should go through the unoccupied doorway.

Fig. 6. **Emergent foraging strategy.** With intention maps, both lifting and pushing teams learn an effective and efficient strategy in which they move single file in a clockwise circle around the center divider.

![Q-value map](image)

**TABLE II**

<table>
<thead>
<tr>
<th>Environment</th>
<th>Ours</th>
<th>No intention maps</th>
</tr>
</thead>
<tbody>
<tr>
<td>SmallEmpty</td>
<td>9.56 ± 0.28</td>
<td>9.08 ± 0.45</td>
</tr>
<tr>
<td>LargeEmpty</td>
<td>19.52 ± 0.21</td>
<td>18.49 ± 0.72</td>
</tr>
</tbody>
</table>

**Search and rescue task.** We then investigate whether spatial intention maps are also helpful for the search and rescue task. Tab. II shows results for teams of 4 rescue robots in the SmallEmpty and LargeEmpty environments. Both quantitative and qualitative results indicate that robots without intention maps are less efficient. They tend to move back and forth unproductively as multiple robots try to rescue the same object, especially towards the end of an episode (when there are only a few objects left). Since the search and rescue task by its nature finishes much more quickly than the foraging task, we are able to see this behavior very clearly in the movement trajectories (Fig. 8). By contrast, robots that use intention maps know the intentions of other robots, and can choose actions that avoid overlapping with others, leading to better distribution of robots throughout the environment. This emerging behavior can be seen clearly in visualizations of Q-value maps of trained policies. For example, in Fig. 7, we see that the locations corresponding to objects another robot already intends to rescue are generally assigned lower Q-values. These results indicate that spatial intention maps can lead to more efficient execution of the rescue task by helping agents avoid overlapping in their efforts.

**Heterogeneous teams.** We also investigate whether spatial intention maps are useful for heterogeneous teams of robots. The bottom five rows of Tab. I show results on the foraging task for teams of 2 lifting and 2 pushing robots (2L+2P), and 2 lifting and 2 throwing robots (2L+2T). While the lifting robot is generally versatile, the pushing robot is particularly adept at pushing objects along walls, while the throwing robot can throw objects long distances backwards. The unique capabilities of the different robot types are complementary to each other, and indeed, we find that when training these heterogeneous teams, a natural division of labor emerges, as shown in Fig. 9. This specialization generally happens with or without spatial intention maps, but we find that when training without intention maps, some of the robots in the team may learn to perpetually wander around without doing anything useful, or the team may never learn to fully complete the task. They likely develop this conservative behavior since they are unable to coordinate and avoid collisions without knowing each other’s intentions. By contrast, when training with intention maps, heterogeneous teams are more efficient and productive.
C. Comparisons and Ablations

In this section, we perform comparisons and ablations across a wide range of communication variants. For all variants, we run experiments on the foraging task using homogeneous teams of 4 lifting robots in all six environments.

Comparison to nonspatial intentions. We compare spatial intention maps to a nonspatial encoding of intentions, in which we add two channels per robot (sorted closest robot to furthest) and tile them with the $x$ and $y$ values of each robot’s intended target location. This nonspatial encoding conveys to the network the $(x, y)$ target of other robots’ intended destination in the best way we can imagine without a spatially varying map. As shown in Tab. III, we find that providing the intentions encoded in a nonspatial format results in much worse performance, suggesting that it is important to use a spatial encoding for agent intentions.

Encoding of spatial intentions. Our method encodes intended paths (with linearly ramped values) in a 2D map. Here we investigate four alternative encodings: (i) a binary encoding of the intention map where the linear ramp of values is replaced with a binarion-path off-path map, (ii) a line encoding that replaces the path encoding with a simple straight line between each agent to its intended target location (which may not necessarily reflect the agent’s true trajectory), (iii) a circle encoding, where a circle marks each agent’s target location (this variant does not associate each agent with its target location), and (iv) an encoding that decomposes the circle intention map into multiple channels (“spatial intention channels” in Tab. III), one per robot, sorted in order from closest robot to furthest (thus allowing a way to associate agents with intentions). Results in Tab. III show that the binary and line variants generally perform on par with our method, while both circle variants are worse overall (but still better than without intention maps). These results suggest that providing any spatial encoding of intention is highly beneficial to multi-agent coordination, as long as there is a clear visual association between agents and their intentions.

Predicting spatial intention maps. We further explore whether agents can learn to predict useful intention maps, either with heuristics or deep networks. We explore three possible methods: (i) predicted intention, which involves training an additional fully convolutional network to predict intention maps from the state representation so that they can be used during execution with no communication, (ii) history maps, where instead of encoding intention, the recent trajectory history of other agents are encoded into a map (assumes robots can track each other’s poses without communicating), and (iii) a combination of (i) and (ii), where an additional network is trained to predict the intention map from a state representation augmented with the history map. Results in Tab. III show that in isolation, (i) and (ii) do not improve performance as much as spatial intention maps do. However, we find that the last variant (iii) which combines history maps with predicted intention achieves performance (Tab. III) almost on par with explicit communication using spatial intention maps. This result is significant in that it provides a method for robot teams to leverage the benefits of spatial intention maps to coordinate without explicit communication, albeit at the cost of extra computation.

D. Real Robot Experiments

As a final test, we take the best policies from our simulation experiments and run them directly on the real robots by building upon the physical setup used in [8]. We use fiducial markers to estimate the poses of robots and objects, which allows us to mirror the real setup inside the simulation. This means we can execute the trained policies (without fine-tuning) on the real robots as if they were moving around in the simulation. During tests with 4 lifting robots in a mirror of the SmallEmpty environment, the robots are able to gather all 10 objects within 1 minute and 56 seconds (on average over 5 episodes). For video results across a wide range of environments, please see the supplementary material at https://spatial-intention-maps.cs.princeton.edu.

V. Conclusion

In this work, we propose spatial intention maps as a new way of communicating intention for improved collaboration in multi-agent RL. In our framework, intentions are spatially encoded in a 2D map, allowing vision-based RL agents to spatially reason about intentions in the same domain as the states and actions. During experiments, we observe that spatial intention maps provide significant gains in performance across a broad range of multi-agent environments, and help robot teams learn emergent collaborative behaviors such as avoiding collisions, coordinating passage through bottlenecks, and distributing through an environment.

Acknowledgments

We thank Naomi Leonard, Anirudha Majumdar, Naveen Verma, and Anish Singhani for fruitful technical discussions. This work was supported by Princeton School of Engineering, as well as NSF under IIS-1815070 and DGE-1656466.


A. Additional Reinforcement Learning Details

We model our task as a Markov decision process (MDP) from the perspective of each individual agent. Specifically, given state $s_t$ at time $t$, the agent takes an action $a_t$ following policy $\pi(s_t)$, and arrives at a new state $s_{t+1}$ while receiving reward $r_t$. In Q-learning, the goal is to find some optimal policy $\pi^*$ that selects actions maximizing the Q-function $Q(s_t, a_t) = \sum_{t'=t}^{\infty} \gamma^{t-t'}r_t$, which represents the discounted sum of future rewards. We train our policies using deep Q-learning (DQN) [15], which approximates the Q-function using a neural network and uses a policy that greedily selects actions maximizing the Q-function: $\pi(s_t) = \arg \max_{a_t} Q_\theta(s_t, a_t)$, where $\theta$ refers to the parameters of the neural network. We train using the double DQN learning objective [49] with smooth L1 loss. Formally, at each training iteration $i$, we minimize $L_i = r_t + \gamma Q_\theta^-(s_{t+1}, \arg \max_{a_t} Q_\theta(s_{t+1}, a_{t+1})) - Q_\theta(s_t, a_t)$, where $(s_t, a_t, r_t, s_{t+1})$ is a transition uniformly sampled from the replay buffer, and $\theta^-$ refers to the parameters of the DQN target network.

B. Additional Training Details

We train DQN using SGD for either 160k or 240k timesteps (depending on the robot types in the team), with batch size 32, learning rate 0.01, momentum 0.9, and weight decay 0.0001. Norms of gradients are clipped to 100 during training. We use a replay buffer of size 10k, constant discount factor $\gamma = 0.85$, train frequency of 4 (train the policy network every 4 timesteps), and we update the target network every 1,000 timesteps. Episodes end after all objects have been removed from the environment, or after 400 consecutive steps in which no object has been removed. Training takes approximately 6 hours on a single Nvidia Titan Xp GPU (for the lifting team).

Network architecture. We model our Q-function $Q_\theta$ with a ResNet-18 [51] backbone, which we transform into a fully convolutional network by removing the AvgPool and fully connected layers and adding three 1x1 convolution layers interleaved with bilinear upsampling layers. Unlike in [8], we apply BatchNorm after convolutional layers.

Exploration. We run a random policy for a small number of timesteps ($1/40$ of the total number of timesteps) in order to fill the replay buffer with some initial experience before we start training the policy network. We use $\epsilon$-greedy exploration, in which we linearly anneal the exploration fraction from 1 to 0.01 during the first 1/10 of training.

Multi-agent. We use one buffer/policy per robot type during training. This means that homogeneous teams pool all transitions into one replay buffer and train one shared policy, while heterogeneous teams share a buffer/policy within each group of robots (of one robot type). For example, the lifting and pushing team trains with two buffers/policies. For decentralized execution, trained policies are independently and asynchronously run on each individual agent.