Proceedings of Machine Learning Research vol 211:1–15, 2023 5th Annual Conference on Learning for Dynamics and Control

# CLAS: Coordinating Multi-Robot Manipulation with Central Latent Action Spaces

**Elie Aljalbout** ELIE.ALJALBOUT@ARGMAX.AI

**Maximilian Karl** KARLMA@ARGMAX.AI

**Patrick van der Smagt**
*Machine Learning Research Lab, Volkswagen Group, Munich, Germany*

**Editors:** N. Matni, M. Morari, G. J. Pappas

## Abstract

Multi-robot manipulation tasks involve various control entities that can be separated into dynamically independent parts. A typical example of such real-world tasks is dual-arm manipulation. Learning to naively solve such tasks with reinforcement learning is often unfeasible due to the sample complexity and exploration requirements growing with the dimensionality of the action and state spaces. Instead, we would like to handle such environments as multi-agent systems and have several agents control parts of the whole. However, decentralizing the generation of actions requires coordination across agents through a channel limited to information central to the task. This paper proposes an approach to coordinating multi-robot manipulation through learned latent action spaces that are shared across different agents. We validate our method in simulated multi-robot manipulation tasks and demonstrate improvement over previous baselines in terms of sample efficiency and learning performance.

**Keywords:** Multi-robot manipulation, latent action spaces, reinforcement learning

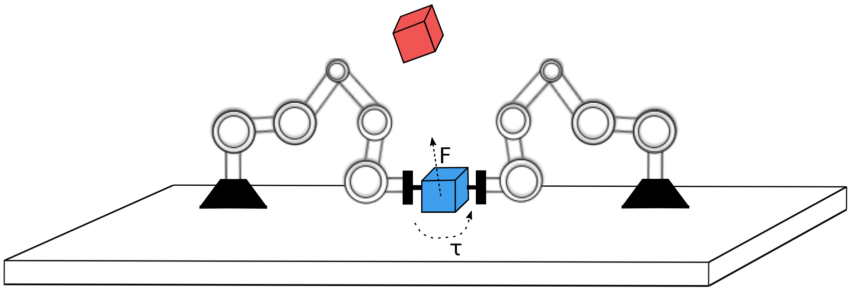


Figure 1: Two robot arms cooperating on an object lifting task. The red cube indicates the target pose. Traditionally, two agents would control the separate robot arms in a control space of the robot such as joint torque control. We explore the option of learning latent central actions spaces which are robot-agnostic and central to the task. In our example, a possible action space would correspond to the force $F$ and torque $\tau$ acting on the center of mass of the cube.

## 1. Introduction

Most recent successes of reinforcement learning (RL) methods have been in single-agent environments. Applications include games (Mnih et al., 2013; Silver et al., 2016), robotics (Kober et al., 2013), and autonomous driving (Kiran et al., 2021). However, many control problems can naturally be distributed to multiple agents. For instance, in robotics, tasks involving multiple robots can be framed as multi-agent systems. In this work, we consider cooperative multi-agent systems. Such environments can also be framed as single-agent RL problems, with policies receiving full observations and outputting a single action responsible for actuating all the different entities. However, such an approach would suffer from great sample complexity due to the difficulty of exploration and fitting a policy under high-dimensional action, state, and observation spaces. Instead, in a multi-agent reinforcement learning (MARL) approach, each agent is responsible for actuating a sub-part of the environment and could have access to either all observations or a subset. This simplifies the exploration and sample requirement for each individual agent.

However, multi-agent methods suffer from the lack of information present to each agent, which results in multiple problems (Graña et al., 2011; Canese et al., 2021). Another challenge for multi-agent systems is the decentralized action generation. This aspect can be ignored for the classical application domains examined by previous MARL research, such as games and particle environments. However, it becomes critical when dealing with physical tasks, such as dual-arm manipulation, where decoupled actions could lead to instabilities and even damage the robots. Hence, in this work, we focus on this problem.

We postulate that there exists an agent-agnostic latent action space that can alternatively be used for solving certain families of cooperative multi-robot manipulation tasks. To illustrate the concept, we take the example of multiple robot manipulators lifting a single object. In the standard multi-agent approach to solving this task, each agent would be responsible for actuating one robot. Each agent's action space would correspond to some control commands suitable for the robot, e. g. joint torques, velocities, Cartesian poses. The goal of this task is to move the object; hence a simpler and more intuitive action space would ideally be expressed with respect to the object and not the robots. For instance, such an action could represent a wrench (force and torque) applied to the object. Task-specific action spaces are not trivial to implement unless the object is rigidly attached to the end-effector of the robot and its physical properties are known. Instead, in this work, we aim to learn such an action space and use it for learning multi-robot manipulation tasks that require coordination. We propose a method for learning latent central action spaces and learning decentralized policies acting independently in these spaces. The previous example is an ideal case where the obtained latent action space has an interpretable physical meaning. However, we restrict ourselves to the general case where this shared latent space could also have a semantic uninterpretable meaning, and study its effect on decentralized learning in cooperative multi-robot manipulation tasks.

## 2. Related Work

**Multi-agent cooperative control** MARL methods assign different agents to different parts of the action and state spaces. This reduces the complexity of learning and exploration of the individual components, but the overall problem remains very challenging. MARL solutions could be simplified using custom policy parametrizations such as finite state controllers (Bernstein et al., 2009; Amato et al., 2010) or transforming the problem to enable tractable planning and search (Dibangoye and Buffet, 2018; Dibangoye et al., 2016). However, decentralized MARL methods fail to achieve the level of coordination, which is needed for physical systems control. Hence, several approaches have been proposed to enable a feasible exchange of information during control. This could either be achieved through explicit communication channels such as in Guestrin et al. (2002); Sukhbaatar et al. (2016); Singh et al. (2018); Das et al. (2019); Pretorius et al. (2020); Niu et al. (2021), or via implicit information exchange as part of the policy architecture or the learning algorithm (Gupta et al., 2017; Lee et al., 2020; Lowe et al., 2017). For instance, multiple methods are based on modeling the other agents' policies (Raileanu et al., 2018; Liu et al., 2020; Yu et al., 2021). This kind of method is usually based on the centralized training decentralized execution (CTDE) paradigm, where at training time, each policy can benefit from the information that is usually exclusive to the other agents at execution time. Others have proposed using CTDE for learning a central dynamics model and use the model for training decentralized policies (Zhang et al., 2021b; Willemsen et al., 2021). Similarly, Lowe et al. (2017) and Foerster et al. (2018) propose training decentralized actors using a centralized critic. Another common approach is to decompose the value function to the different agents (Sunehag et al., 2018; Rashid et al., 2018). Beyond CTDE, multiple other solutions have been proposed for alleviating the non-stationarity of MARL tasks. For instance, Liu et al. (2021) propose engineering the reward function to punish competitive actions taken by individual agents. Gupta et al. (2017) relied on policy parameter sharing across agents, which allows multiple agents to use the same policy network while passing an agent index as part of the observation. A more extensive overview of MARL methods can be found in (Zhang et al., 2021a).

**Latent action representation** Different control settings enable different kinds of behavior. For instance, motion control is ideal for reaching a given goal but not perfectly suited for manipulating objects or applying forces. Similarly, in RL, previous work showed that the choice of action space representation could lead to improvements in sample efficiency, energy consumption, robustness (Martín-Martín et al., 2019; Bogdanovic et al., 2020; Aljalbout et al., 2021; Varin et al., 2019; Ulmer et al., 2021) or learning speed (Peng and van de Panne, 2017). Types of action representation include torque, joint PD-controller, inverse dynamics, muscle activation or variable impedance control (Peng and van de Panne, 2017; Varin et al., 2019; Martín-Martín et al., 2019; Bogdanovic et al., 2020), but also dynamic movement primitives (DMP) can be seen as an action representation (Buchli et al., 2011; Schaal, 2006). Bahl et al. (2020) embed DMPs in the action space by having the policy output the DMP parameters. The studies further show that a relation between the task space and choice of action space is important (Martín-Martín et al., 2019). Ganapathi et al. (2022) integrate a differentiable implementation of forward kinematics in neural networks to combine cartesian and joint space control. More recent publications also show the possibility of learning these action representations from interaction with the environment. Zhou et al. (2020) and Allshire et al. (2021) learn a conditional variational autoencoder in order to obtain a latent action space. Policy search is then performed in this latent action representation. During inference, the decoder part of the auto-encoder is used to transform the latent action back into the original action space. Zhou et al. (2020) additionally emphasize constraining the policy to be within the support of the dataset. Karamcheti et al. (2021) use language embeddings to inform the learning of latent action spaces. Rana et al. (2022) learn a latent skill-based action space where the skills run at a higher frequency than the policy actions.

## 3. Method

We are mainly interested in learning multi-robot manipulation tasks. Typically, such tasks involve multiple robot manipulators (i. e. robot arms) that simultaneously interact with an object to achieve a predefined goal. Having a single actor control all robots would ideally lead to good coordination, but would suffer in exploration due to the large action and state spaces (Alles and Aljalbout, 2022).

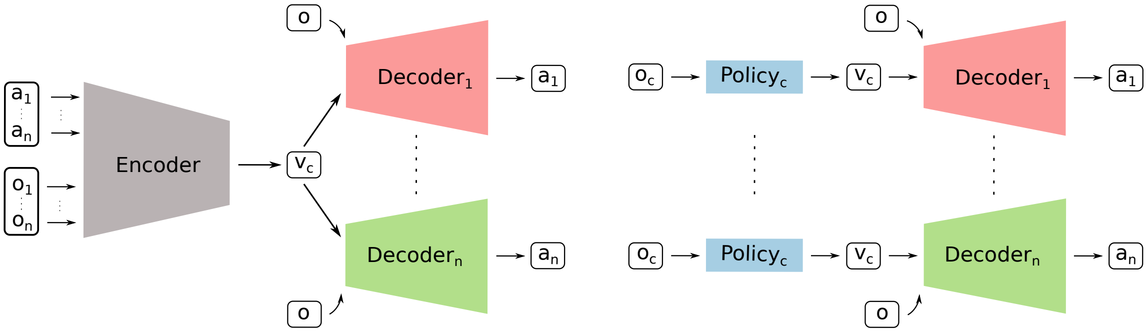


Figure 2: System overview under full agent observability. (Left) we use a conditional autoencoder for learning the central latent action space. The encoder receives all observations and actions from all agents and produces a latent action $v_c$. This latent action together with the full observation is given to the agent-specific decoders together with the observation. Each decoder outputs an action that is in the original action space of the corresponding agent. (Right) All agents share the same policy acting in the latent action space. The learned decoders map the latent action into the original action space.

Alternatively, control could be split into multiple agents handling one robot each. By doing so, we reduce the dimensionality of the individual agents' action and state spaces, hence reducing the sample complexity and exploration requirements. However, by decentralizing the process of action generation, coordination between the different agents' policies becomes challenging. Another main difference to single-agent approaches is the lack of information present to each agent at execution time. Namely, each agent can only receive a subset of all the observations of the environment. The local agent observations usually correspond to agent-specific observations $\mathbf{o}_i$ (e.g. proprioceptive measurements in a robotics scenario) and task-related observations $\mathbf{o}_c$ (e.g. object poses), that are shared with all agents.

### 3.1. Problem Formulation

We formulate decentralized cooperative control tasks as decentralized partially-observable Markov decision processes (Dec-POMDP), defined by the set $\langle N, \mathcal{X}, \{\mathcal{U}_i\}_{i\in\{1,\ldots,N\}}, \mathcal{T}, \{r_i\}_{i\in\{1,\ldots,N\}}, \gamma, \{\mathcal{O}_i\}_{i\in\{1,\ldots,N\}}, \rho\rangle$, where $N$ is number of agents ($N = 1$ corresponds to the single-agent problem), $\mathcal{X}$ is the state space shared by all agents, $\mathcal{U}_i$ is the action space for agent $i$, $\mathcal{T}$ represents the environment dynamics, $r_i$ is the reward function for agent $i$, $\gamma$ is the discount factor, $\mathcal{O}_i$ is the observation space of agent $i$, and $\rho$ is the initial state distribution. Since we are interested in cooperative tasks, all agents share the same reward $r_1 = r_2 = \cdots = r_N$. Dec-POMDP is a special type of partially observed stochastic games. Optimally solving Dec-POMDPs is a challenging combinatorial problem that is NEXP-complete (Bernstein et al., 2002).

### 3.2. Central Latent Action Spaces

Previous approaches relied on the centralized training and decentralized execution paradigm to allow using full observation and action information at least at training time. We follow this line of work and propose learning a latent central action space $\mathcal{V}$, which is shared across all agents. Controls in this space represent single actions acting on the whole environment and should somehow be

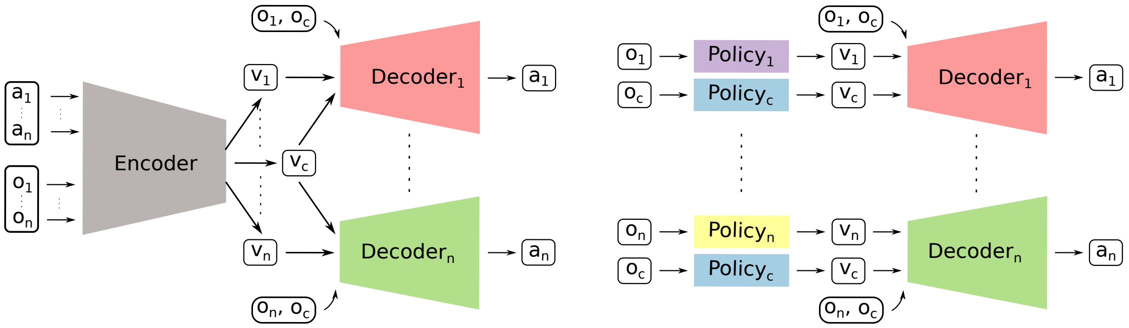


Figure 3: System overview under partial agent observability. (Left) we use a conditional autoencoder for learning the central latent action space. The encoder receives all observations and actions from all agents and produces a latent action $v$. The latent action contains agent-specific actions $v_i$ as well as a central latent action $v_c$. This latent action together with each agent's observation are given to the agent-specific decoders together with the observation. Each decoder outputs an action that is in the original action space of the corresponding agent. (Right) All agents share the same policy acting on the object in the latent action space. They each have a separate policy acting in the latent agent-specific action space. We use the learned decoders to map the latent action into its original action space(s).

translated again into commands to be executed by the individual robots. The motivation behind this method is that cooperative tasks usually involve different agents manipulating the same task-related entities to achieve a high-level goal. The control commands from all the agents are reflected directly on those entities in a simpler way, e. g. multiple robots interacting with an object results in a single force (sum of all force vectors) acting on this object. Our approach is illustrated in figure 2. We learn a central latent action space using a conditional autoencoder. Given this model, all agents can share a single policy that outputs actions in the latent action space. We then map the obtained latent actions to the original action space of each robot based on the learned decoders. The encoder is no longer needed after training.

To learn a latent central action space, we use Stochastic Gradient Variational Bayes (Kingma and Welling, 2013) to overcome the intractable inference distributions involved in learning mappings to this space. First, we look at the case where each agent receives full observations $\mathbf{o} \in \mathcal{O}_1 \times \mathcal{O}_2 \times \cdots \times \mathcal{O}_N$. For that, we introduce the models in figure 2 (left). The generative process (encoding) of each agent's original action $\mathbf{u}_i$ is conditioned on the latent central action $\mathbf{v}$ and the observation $\mathbf{o}$. The latter is also used during the inference process (decoding), as shown in figure 2 (left). Additionally, we encode latent actions based on the actions from all agents $\mathbf{u} = [\mathbf{u}_1, \ldots, \mathbf{u}_N]$. This is possible since the inference/encoder network will not be used at execution time, and will instead be replaced by the policy as shown in figure 2 (right). Based on this model, all agents could share a copy of the same policy, which outputs a latent central action $\mathbf{v}$ based on the full observation $\mathbf{o}$. However, they would each have a different decoder to translate the latent action $\mathbf{v}$ into their original action space. This is illustrated in figure 2 (right). Having a shared policy is feasible in this scenario since the latent action space is supposed to have a lower dimensionality than the aggregated action space of all control agents. To illustrate this, we go back to the lifting example. Controlling the joint velocity of two robots with six degrees of freedom would result in an action space with a dimension of twelve. Instead, controlling the wrench applied to the object only requires an action space with six dimensions. Note that this number does not grow dramatically with the number of agents or robots. We derive a lower bound to the marginal likelihood:

$$\begin{aligned}
p(\mathbf{u} \mid \mathbf{o}) &= \int p_\theta(\mathbf{u} \mid \mathbf{o}, \mathbf{v})\, p_\psi(\mathbf{v} \mid \mathbf{o})\, d\mathbf{v} \\
\ln p(\mathbf{u} \mid \mathbf{o}) &= \ln \int p_\theta(\mathbf{u} \mid \mathbf{o}, \mathbf{v})\, p_\psi(\mathbf{v} \mid \mathbf{o})\, \frac{q_\phi(\mathbf{v} \mid \mathbf{o}, \mathbf{u})}{q_\phi(\mathbf{v} \mid \mathbf{o}, \mathbf{u})}\, d\mathbf{v} \\
&\geq \int q_\phi(\mathbf{v} \mid \mathbf{o}, \mathbf{u}) \ln(p_\theta(\mathbf{u} \mid \mathbf{o}, \mathbf{v})) \frac{p_\psi(\mathbf{v} \mid \mathbf{o})}{q_\phi(\mathbf{v} \mid \mathbf{o}, \mathbf{u})} d\mathbf{v} \\
&= \mathbb{E}_{q_\phi(\mathbf{v} \mid \mathbf{o}, \mathbf{u})}[\ln p_\theta(\mathbf{u} \mid \mathbf{o}, \mathbf{v})] - \mathrm{KL}(q_\phi(\mathbf{v} \mid \mathbf{o}, \mathbf{u}) \mid\mid p_\psi(\mathbf{v} \mid \mathbf{o})) && (1) \\
&= \mathcal{L}(\mathbf{u}, \theta, \phi, \psi \mid \mathbf{o}), && (2)
\end{aligned}$$

where $\mathrm{KL}(||)$ is the Kullback-Leibler divergence, $q(\mathbf{v} \mid \mathbf{o}, \mathbf{u})$ is the approximate posterior distribution:

$$\begin{aligned}
q_\phi(\mathbf{v} \mid \mathbf{o}, \mathbf{u}) &= \mathcal{N}\left(\mathbf{v}; \mu_v, \sigma_v^2\right) \\
[\mu_v, \sigma_v] &= g_\phi(\mathbf{o}, \mathbf{u}). && (3)
\end{aligned}$$

Since the generative process of each agent's action is distributed, the likelihood is composed of multiple terms:

$$p_\theta(\mathbf{u} \mid \mathbf{o}, \mathbf{v}) = [p_{\theta_1}(\mathbf{u}_1 \mid \mathbf{o}, \mathbf{v}), \ldots, p_{\theta_N}(\mathbf{u}_N \mid \mathbf{o}, \mathbf{v})], \tag{4}$$

Where $\theta_i$ refers to decoder parameters for agent $i$, and $\theta = \{\theta_i\}_{i \in N}$. Note that the prior is conditioned on the observations. It is parameterized by $\psi$ and has a policy-like form $p_\psi(\mathbf{v} \mid \mathbf{o})$. We train it simultaneously to the encoder and decoders using the same loss function from equation (2).

As previously mentioned, agents in a Dec-POMDP have only access to a subset of the observations. However, we notice that in most environments, a certain part of the observations is shared across all agents, and that is usually related to either objects in the scene or any kind of other task-specific observations; but not to the agent's embodiment. Even when this condition fails, it could be enforced in the learning process. For instance, in (Liu and Kitani, 2022), the latent space is designed to contain information about the object relevant to the task.

We introduce a new set of models, as seen in figure 3 (left). In this new model, the latent action space is partitioned into $N + 1$ parts. The first $N$ correspond to latent actions $\mathbf{v}_i$, which are specific to each agent. The last part $\mathbf{v}_c$ is central and shared with all agents. The generative process of each agent's action (in the original action space) is now conditioned on the agent's observation $\mathbf{o}_i$, the latent agent-specific action $\mathbf{v}_i$, and the latent central action $\mathbf{v}_c$. As for inference, the whole latent action variable is conditioned on the full observation $\mathbf{o}$ and the full action $\mathbf{u}$. As in the previous case, using the full observation and action for inference is possible because the encoder would not be used during control. Instead, each agent has a set of two policies: one policy producing the latent agent-specific action $\mathbf{v}_i$ based on $\mathbf{o}_i$; and another policy that is shared across all agents, and which generates the latent central action based on the shared observation $\mathbf{o}_c$. These two latent actions are then concatenated and decoded into the original action space of the agent. We show the architecture of the policy in figure 3 (right). Note that the policy updates also affect the decoder. The new lower bound is very similar to the one in equation (2), with the minor difference of:

$$p_\theta(\mathbf{u} \mid \mathbf{o}, \mathbf{v}) = [p_{\theta_1}(\mathbf{u}_1 \mid \mathbf{o}_1, \mathbf{o}_c, \mathbf{v}_1, \mathbf{v}_c), \ldots, p_{\theta_N}(\mathbf{u}_N \mid \mathbf{o}_N, \mathbf{o}_c, \mathbf{v}_N, \mathbf{v}_c)]. \tag{5}$$

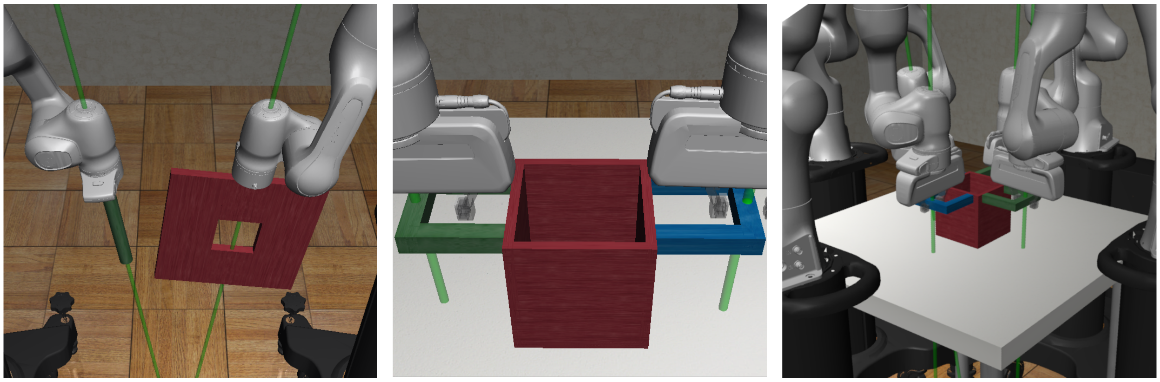


Figure 4: Close-up screenshots from the simulation environments used in our experiments. The environments are provided by robosuite (Zhu et al., 2020). (left) dual-arm-peg-in-hole environment (middle) dual-arm-lift and (right) four-arm-lift environment with modified gripper structure.

### 3.3. Implementation Details:

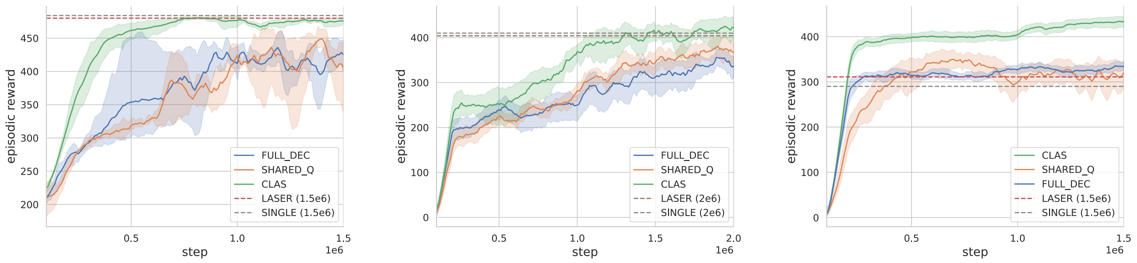
The encoders, decoders, prior distributions, and policies involved in this method are implemented as multi-layer neural networks using PyTorch (Paszke et al., 2019). All distributions are transformed gaussian distributions using a hyperbolic tangent function ($tanh$). Actor, critic, and prior networks have two hidden layers. Encoders and decoders have three hidden layers. For training, we use the Adam optimizer (Kingma and Ba, 2014). Each policy is optimized using soft-actor-critic (SAC) (Haarnoja et al., 2018). All modules are trained using randomly sampled data from the replay buffer. The latter contains trajectories sampled from the previously described multi-agent policy. At the beginning of training, we only update the latent action model using random actions in a warm-up phase that lasts for a hundred thousand steps. We found this step to help the training performance and stability. For a fair comparison, we also implement this step for all the considered baselines.

## 4. Experiments

We designed our experiments to investigate the following questions: (*Q1*) Can central latent action spaces help coordinate action generation in decentralized cooperative robotic manipulation? (*Q2*) Does our method improve sample efficiency with respect to the selected baselines? (*Q3*) Can our method reach or exceed the performance of single-agent approaches with full-state information? (*Q4*) Is our method scalable to more than two-arm manipulation tasks? (*Q5*) How robust is our method to external disturbances?

### 4.1. Environments

We evaluated our method in three simulated environments based on robosuite (Zhu et al., 2020). The environments are selected/built such that they require cooperation between multiple robot arms. Due to the lack of standardized environments that are suitable for our use case (i. e. multi-robot manipulation), we use existing environments from public benchmarks when suitable and build alternatives when needed. Due to the nature of our problem, we select environments that have continuous state and action spaces. In all of the environments, each agent's observations are the corresponding robot's joint position and velocity as well as its end-effector pose.



(a) dual-arm-peg-in-hole (b) dual-arm-lift (c) four-arm-lift

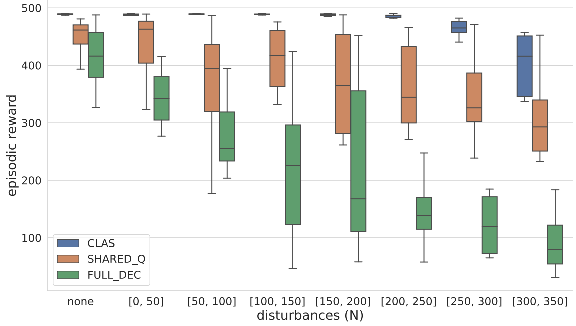
Figure 5: Performance in simulated multi-robot manipulation tasks under Partial-observability. We compare our method (CLAS) to centralized single-agent and decentralized multi-agent approaches. Our approach outperforms the considered decentralized multi-agent approaches in all environments. It also manages to solve the four-arm-lift task in which all the considered baselines fail.

**Dual-arm-peg-in-hole.** In the first environment, two robot arms cooperate on the peg-in-hole task. A close-up view of the scene and the objects can be seen on the left in figure 4. We are using the original reward from robosuite, which is composed of a reaching and orientation reward. The shared observation $o_c$ corresponds to the poses of the peg and hole and the distance between them.

**Dual-arm-lift.** For the second environment, we decided to use the dual-arm-lift environment. In this environment, a rectangular pot with two handles is placed on a surface between two robot arms. The task for each robot is to reach and grip the handle before cooperatively lifting the pot off the surface. During initial experiments, we noticed that the provided reward does not promote cooperation and can be easily tricked, i. e. the maximum reward per time step can be reached by controlling a single robot to tilt and lift the pot only slightly off the table. This is due to the generous maximum tilt angle of 30° and the successful lift height of 0.10. The other major component of the reward measures the ability to reach and grasp the pot handles. However, we are not interested in assessing the reaching and gripping capabilities but want to rather reward cooperative lifting behavior. Therefore we are considering the following modifications to the reward of the environment. At the start of an episode, we move each robot's end-effector close to its handle and weld the pot handle to the end-effector with a distance constraint in the MuJoCo ((Todorov et al., 2012)) simulator. We chose a distance constraint because it constrains the position but leaves rotational coordinates free. We remove the gripper fingers to avoid unwanted collisions. We visualize the resulting starting condition in the middle of figure 4. We also modify the reward function to enforce success only during high lifts. Additionally, the maximum tilt angle is reduced such that both robots must cooperate to keep the pot level at all times. The shared observation $o_c$ corresponds to the pose of the pot.

**Four-arm-lift.** The third environment is an extension of the dual-arm-lift environment and uses two additional robot arms to lift the pot (i. e. total of four robot arms). Here the pot weight is increased to keep the coordination requirement. We build this environment for the sole purpose of testing the scalability to more than two robots/agents. The pot with four handles and the robot arms' placement can be seen on the right in figure 4.

The changes to the lifting environments were evaluated with manual human control to ensure that tricking the system or solving the task with a single robot arm is not possible. Keeping a high reward was only possible when the pot is lifted vertically for a long period of steps. All environments use a joint velocity controller which receives desired joint velocities from the policy.



Success rates over 10 evaluation runs under external disturbances.

| | FULL_DEC | SHARED_Q | CLAS |
|---|---|---|---|
| none | 70% | **100%** | **100%** |
| $[0, 50]$ | 30% | 80% | **100%** |
| $[50, 100]$ | 20% | 60% | **100%** |
| $[100, 150]$ | 10% | 70% | **100%** |
| $[150, 200]$ | 20% | 40% | **100%** |
| $[200, 250]$ | 0% | 40% | **100%** |
| $[250, 300]$ | 0% | 30% | **90%** |
| $[300, 350]$ | 0% | 20% | **60%** |

Figure 6: Effect of applying disturbances (forces) at the center of mass of the pot in the four-arm-lift environment. (left) Episode reward (right) success rate under different ranges of disturbances. Results are based on 10 evaluation runs. Our method demonstrates robustness against different ranges of disturbances in comparison to the other decentralized baselines, which success rate decreases dramatically as we increase the disturbance.

### 4.2. Baselines:

To validate our method, we compare it to well-established baselines that have been previously applied to continuous control. Our experiments include the following baselines:

- `SINGLE`: refers to having a single agent controlling all robots.
- `LASER`: uses a latent action space on top of a single agent controlling all robots. This is based on the work in (Allshire et al., 2021).
- `FULL_DEC`: refers to having all agents trained with the exact observations and actions they will have access to during execution. The agents are not provided with a communication channel.
- `SHARED_Q`: similar architecture to `FULL_DEC`, but all agents are trained using a central critic. This baseline is based on the work in (Lowe et al., 2017).
- `CLAS`: refers to our method and abbreviates “central latent action spaces.”

The first two single-agent approaches are included as strong baselines and references. They serve us to better understand the different environments and to elaborately analyze our results. Finally, to make the comparison more reliable, we use SAC for training the different agents in all baselines.

### 4.3. Results

**Task Performance.** Figure 5 shows the episodic reward obtained by our method and the baselines on the three considered environments. Looking at the single-agent approaches, we observe that both baselines reach high reward areas for the dual-arm tasks. However, they both fail to solve the four-arm-lift task. At the end of training, the best mean episode reward achieved by a single agent is substantially smaller than the maximum possible reward and has a very large variance. This illustrates the problem of learning multi-robot manipulation tasks with large action and observation spaces with a single-agent RL approach. In contrast to the dual-arm tasks, the four-arm-lift environment features state and action spaces twice the size. Next, we analyze the results from MARL-based methods. `FULL_DEC` and `SHARED_Q` struggle to keep up with single-agent RL methods. Both methods do not explicitly encourage coordination. Hence, this result might indicate that our environments are well-suited for studying Dec-POMDPs, since they require a certain degree of coordination to be solved. The two approaches manage to solve the peg-in-hole task but struggle in the two other environments. They also lead to very similar results. In contrast, our method (CLAS) successfully solves all tasks even under partial observability. In the dual-arm-peg-in-hole environment, it reaches a high episode reward after only 250 thousand environment interaction steps, while the two other MARL approaches fail to do so in triple the number of steps. Furthermore, it achieves a final performance very close to the one achieved by single-agent methods. In the dual-arm-lift environment, our approach outperforms both MARL-based baselines. Additionally, it surpasses the final performance of the two other MARL approaches after only half amount of steps. More importantly, CLAS slightly outperforms the single-agent methods. In the four-arm-lift environment, CLAS is the only studied method that manages to solve the task and achieve a high reward. Even the single-agent baselines which have access to full state information fail in this task. This indicates that acting in the latent central action space enables coordinated control even under partial observability and action decentralization. Finally, we notice that our method leads to significantly lower performance variance, which makes deploying it in real-world scenarios more reliable.

**Robustness analysis.** We aim to evaluate the coordination capability of our method by quantifying its robustness to external disturbances. We perform this experiment on the four-arm-lift environment and compare the different decentralized baselines to our method. For each method, we pick the model from the training run with the best-achieved performance. We then evaluate the corresponding agents in the same environment as before, however, when additionally applying an external force to the pot. The force is applied during the steps in the interval $[10, 100]$, and the values of the force vector are uniformly sampled at each step to be in a certain range. We experimented with multiple ranges. The results can be seen in figure 6. Under no disturbances ("none"), all methods achieve a high reward and a decent success rate. After applying disturbances in the range $[200 - 250]$, FULL_DEC fails in all evaluation runs to solve the task. The success rate of SHARED_Q goes down to $40\%$, but its reward remains relatively high as the agent manages to lift the pot a bit but not always to the target height. On the other hand, our method CLAS is almost not affected by this level of disturbance. As expected, when increasing the magnitude of the forces, all methods start to fail more often at solving the task, but CLAS appears to remain reasonably robust.

We perform additional ablations and experiments to validate the coordination behavior of our method, and study its performance under full agent observability and asymmetric action spaces with different robots per agent. We also attempt to interpret the learned actions spaces. The results can be seen in our extended technical report (Aljalbout et al., 2022).

## 5. Conclusion

We propose latent central action spaces for decentralized multi-agent control in cooperative tasks. The main idea behind our method is to enable coordinated control of multiple robot manipulators based on sharing a latent action space that is agent-agnostic. During training time, our approach benefits from central access to all observations and actions and uses this data to train the latent action space model. During execution, each agent benefits from the latent central action model to produce control commands that are coordinated with other agents. We compare our approach to different baselines and show that latent central action spaces improve the overall performance and efficiency of learning. Interestingly, our method solves a task in which centralized baselines struggle. Finally, we show that our approach improves robustness against external disturbances.

## References

Elie Aljalbout, Ji Chen, Konstantin Ritt, Maximilian Ulmer, and Sami Haddadin. Learning vision-based reactive policies for obstacle avoidance. In Conference on Robot Learning, pages 2040–2054. PMLR, 2021.

Elie Aljalbout, Maximilian Karl, and Patrick van der Smagt. Clas: Coordinating multi-robot manipulation with central latent action spaces, 2022. URL https://arxiv.org/abs/2211.15824.

Marvin Alles and Elie Aljalbout. Learning to centralize dual-arm assembly. Frontiers in Robotics and AI, 9, 2022.

Arthur Allshire, Roberto Martín-Martín, Charles Lin, Shawn Manuel, Silvio Savarese, and Animesh Garg. Laser: Learning a latent action space for efficient reinforcement learning. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 6650–6656. IEEE, 2021.

Christopher Amato, Daniel S Bernstein, and Shlomo Zilberstein. Optimizing fixed-size stochastic controllers for pomdps and decentralized pomdps. Autonomous Agents and Multi-Agent Systems, 21(3):293–320, 2010.

Shikhar Bahl, Mustafa Mukadam, Abhinav Gupta, and Deepak Pathak. Neural dynamic policies for end-to-end sensorimotor learning. Advances in Neural Information Processing Systems, 33: 5058–5069, 2020.

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. Mathematics of operations research, 27(4): 819–840, 2002.

Daniel S Bernstein, Christopher Amato, Eric A Hansen, and Shlomo Zilberstein. Policy iteration for decentralized control of markov decision processes. Journal of Artificial Intelligence Research, 34:89–132, 2009.

Miroslav Bogdanovic, Majid Khadiv, and Ludovic Righetti. Learning variable impedance control for contact sensitive tasks. IEEE Robotics and Automation Letters, 5(4):6129–6136, 2020. ISSN 2377-3766. doi: 10.1109/LRA.2020.3011379.

Jonas Buchli, Freek Stulp, Evangelos Theodorou, and Stefan Schaal. Learning variable impedance control. The International Journal of Robotics Research, 30(7):820–833, 2011. ISSN 0278-3649. doi: 10.1177/0278364911402527. URL https://doi.org/10.1177/0278364911402527. Publisher: SAGE Publications Ltd STM.

Lorenzo Canese, Gian Carlo Cardarilli, Luca Di Nunzio, Rocco Fazzolari, Daniele Giardino, Marco Re, and Sergio Spanò. Multi-agent reinforcement learning: A review of challenges and applications. Applied Sciences, 11(11):4948, 2021.

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

Jilles Dibangoye and Olivier Buffet. Learning to act in decentralized partially observable MDPs. In Jennifer Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 1233–1242. PMLR, 2018.

Jilles Steeve Dibangoye, Christopher Amato, Olivier Buffet, and François Charpillet. Optimally solving dec-pomdps as continuous-state mdps. Journal of Artificial Intelligence Research, 55: 443–497, 2016.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018.

Aditya Ganapathi, Pete Florence, Jake Varley, Kaylee Burns, Ken Goldberg, and Andy Zeng. Implicit kinematic policies: Unifying joint and cartesian action spaces in end-to-end robot learning. arXiv preprint arXiv:2203.01983, 2022.

Manuel Graña, Borja Fernandez-Gauna, and Jose Manuel Lopez-Guede. Cooperative multi-agent reinforcement learning for multi-component robotic systems: guidelines for future research. Paladyn, 2(2):71–81, 2011.

Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In ICML, volume 2, pages 227–234. Citeseer, 2002.

Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In International conference on autonomous agents and multiagent systems, pages 66–83. Springer, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning, pages 1861–1870. PMLR, 2018.

Siddharth Karamcheti, Megha Srivastava, Percy Liang, and Dorsa Sadigh. LILA: Language-informed latent actions. In 5th Annual Conference on Robot Learning, 2021. URL https://openreview.net/forum?id=_lkBGOctkip.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems, 2021.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. The International Journal of Robotics Research, 32(11):1238–1274, 2013.

Youngwoon Lee, Jingyun Yang, and Joseph J. Lim. Learning to coordinate manipulation skills via skill behavior diversification. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=ryxB2lBtvH.

Luyu Liu, Qianyuan Liu, Yong Song, Bao Pang, Xianfeng Yuan, and Qingyang Xu. A collaborative control method of dual-arm robots based on deep reinforcement learning. Applied Sciences, 11 (4):1816, 2021.

Minghuan Liu, Ming Zhou, Weinan Zhang, Yuzheng Zhuang, Jun Wang, Wulong Liu, and Yong Yu. Multi-agent interactions modeling with correlated policies. arXiv preprint arXiv:2001.03415, 2020.

Xingyu Liu and Kris M. Kitani. V-mao: Generative modeling for multi-arm manipulation of articulated objects. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, Proceedings of the 5th Conference on Robot Learning, volume 164 of Proceedings of Machine Learning Research, pages 287–296. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/liu22a.html.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, pages 6382–6393. Curran Associates Inc., 2017. ISBN 978-1-5108-6096-4.

Roberto Martín-Martín, Michelle A. Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, and Animesh Garg. Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1010–1017, 2019. doi: 10.1109/IROS40897.2019.8968201. ISSN: 2153-0866.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

Yaru Niu, Rohan R Paleja, and Matthew C Gombolay. Multi-agent graph-attention communication and teaming. In AAMAS, pages 964–973, 2021.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019.

Xue Bin Peng and Michiel van de Panne. Learning locomotion skills using DeepRL: does the choice of action space matter? In Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation, SCA '17, pages 1–13. Association for Computing Machinery, 2017. ISBN 978-1-4503-5091-4. doi: 10.1145/3099564.3099567. URL https://doi.org/10.1145/3099564.3099567.

Arnu Pretorius, Scott Cameron, Andries Petrus Smit, Elan van Biljon, Lawrence Francis, Femi Azeez, Alexandre Laterre, and Karim Beguir. Learning to communicate through imagination with model-based deep multi-agent reinforcement learning. 2020.

Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In International conference on machine learning, pages 4257–4266. PMLR, 2018.

Krishan Rana, Ming Xu, Brendan Tidd, Michael Milford, and Niko Suenderhauf. Residual skill policies: Learning an adaptable skill-based action space for reinforcement learning for robotics. In 6th Annual Conference on Robot Learning, 2022. URL https://openreview.net/forum?id=0nb97NQypbK.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In International Conference on Machine Learning, pages 4295–4304. PMLR, 2018.

Stefan Schaal. Dynamic movement primitives -a framework for motor control in humans and humanoid robotics. In Adaptive Motion of Animals and Machines. Springer, 2006.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587):484–489, 2016. ISSN 0028-0836. doi: 10.1038/nature16961.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In International Conference on Learning Representations, 2018.

Sainbayar Sukhbaatar, arthur szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 29. Curran Associates, Inc., 2016.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In AAMAS, pages 2085–2087, 2018. URL http://dl.acm.org/citation.cfm?id=3238080.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5026–5033. IEEE, 2012.

Maximilian Ulmer, Elie Aljalbout, Sascha Schwarz, and Sami Haddadin. Learning robotic manipulation skills using an adaptive force-impedance action space. arXiv preprint arXiv:2110.09904, 2021.

Patrick Varin, Lev Grossman, and Scott Kuindersma. A comparison of action spaces for learning manipulation tasks. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6015–6021, 2019. doi: 10.1109/IROS40897.2019.8967946. ISSN: 2153-0866.

Daniël Willemsen, Mario Coppola, and Guido CHE de Croon. Mambpo: Sample-efficient multi-robot reinforcement learning using learned world models. In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5635–5640. IEEE, 2021.

Xiaopeng Yu, Jiechuan Jiang, Haobin Jiang, and Zongqing Lu. Model-based opponent modeling. arXiv preprint arXiv:2108.01843, 2021.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. Handbook of Reinforcement Learning and Control, pages 321–384, 2021a.

Qizhen Zhang, Chris Lu, Animesh Garg, and Jakob Foerster. Centralized model and exploration policy for multi-agent rl. arXiv preprint arXiv:2107.06434, 2021b.

Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. In Conference on Robot Learning, 2020.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. In arXiv preprint arXiv:2009.12293, 2020.