Rapid Motor Adaptation for Robotic Manipulator Arms

Yichao Liang[1,2] Kevin Ellis[3] João Henriques[2]

[1]Computational and Biological Learning Lab, University of Cambridge
[2]Visual Geometry Group, University of Oxford
[3]Cornell University

yliang6@gmail.com, kellis@cornell.edu, joao@robots.ox.ac.uk

Abstract

Developing generalizable manipulation skills is a core challenge in embodied AI. This includes generalization across diverse task configurations, encompassing variations in object shape, density, friction coefficient, and external disturbances such as forces applied to the robot. Rapid Motor Adaptation (RMA) offers a promising solution to this challenge. It posits that essential hidden variables influencing an agent's task performance, such as object mass and shape, can be effectively inferred from the agent's action and proprioceptive history. Drawing inspiration from RMA in locomotion and in-hand rotation, we use depth perception to develop agents tailored for rapid motor adaptation in a variety of manipulation tasks. We evaluated our agents on four challenging tasks from the Maniskill2 benchmark, namely pick-and-place operations with hundreds of objects from the YCB and EGAD datasets, peg insertion with precise position and orientation, and operating a variety of faucets and handles, with customized environment variations. Empirical results demonstrate that our agents surpass state-of-the-art methods like automatic domain randomization and vision-based policies, obtaining better generalization performance and sample efficiency.

1. Introduction

With recent advances in computer vision [11, 19, 25] and high-level planning [1, 36], dexterous manipulation of objects (i.e. low-level control skills) remains one of the last major obstacles to the creation of robots that can help in general manipulation tasks. Such an advance would have a wide-ranging impact, allowing robots to take on repetitive tasks in industry and in households.

Classical approaches to robotic manipulation often rely on accurate models of both the robot and the environment [9]. The complexity of creating these models can be a significant hurdle, as they need to account for various physical properties and constraints. On the other hand, many Reinforcement Learning (RL) methods are very sample-inefficient, and fail to generalize robustly [33]. Many efforts have therefore been invested into simulation training for real-world deployment [43]. However, models trained in simulation often fail to perform well in the real world due to the sim-to-real gap – direct deployment (without any domain adaptation) results in decreased performance [3]. More generally, RL agents face the challenge of generalizing to unseen tasks or even tasks with out-of-distribution configurations.

To address these limitations, Kumar et al. [20] proposed Rapid Motor Adaptation (RMA), and demonstrated it for quadruped robot locomotion. The main idea behind RMA is to train a policy that is conditional on environmental factors which are not available in real-world deployment, but are easily randomized and conditioned on during simulation training. A predictor, called the *adaptation module* (adapter for brevity), is then trained to regress these factors from available sensors (such as proprioception). This is possible because environmental factors, such as the density, friction, and ground elevation, can be reasonably inferred based on the dynamic response of the robot (e.g. the difference between desired and actually observed motion). In particular, these factors do not usually have to be precisely predicted for the agent to successfully conduct these tasks, which is why a low-dimensional projection of the environment factors is sufficient (and removes the ambiguity of useless but difficult-to-estimate factors) [20].

While this demonstrates an encouraging path forward, it is not straightforward to bring RMA to general manipulation tasks, which feature diverse objectives and behaviours depending on each object's characteristics. Proprioception alone does not suffice, as it only contains information about the object after touching it – visual reasoning prior to grasp-

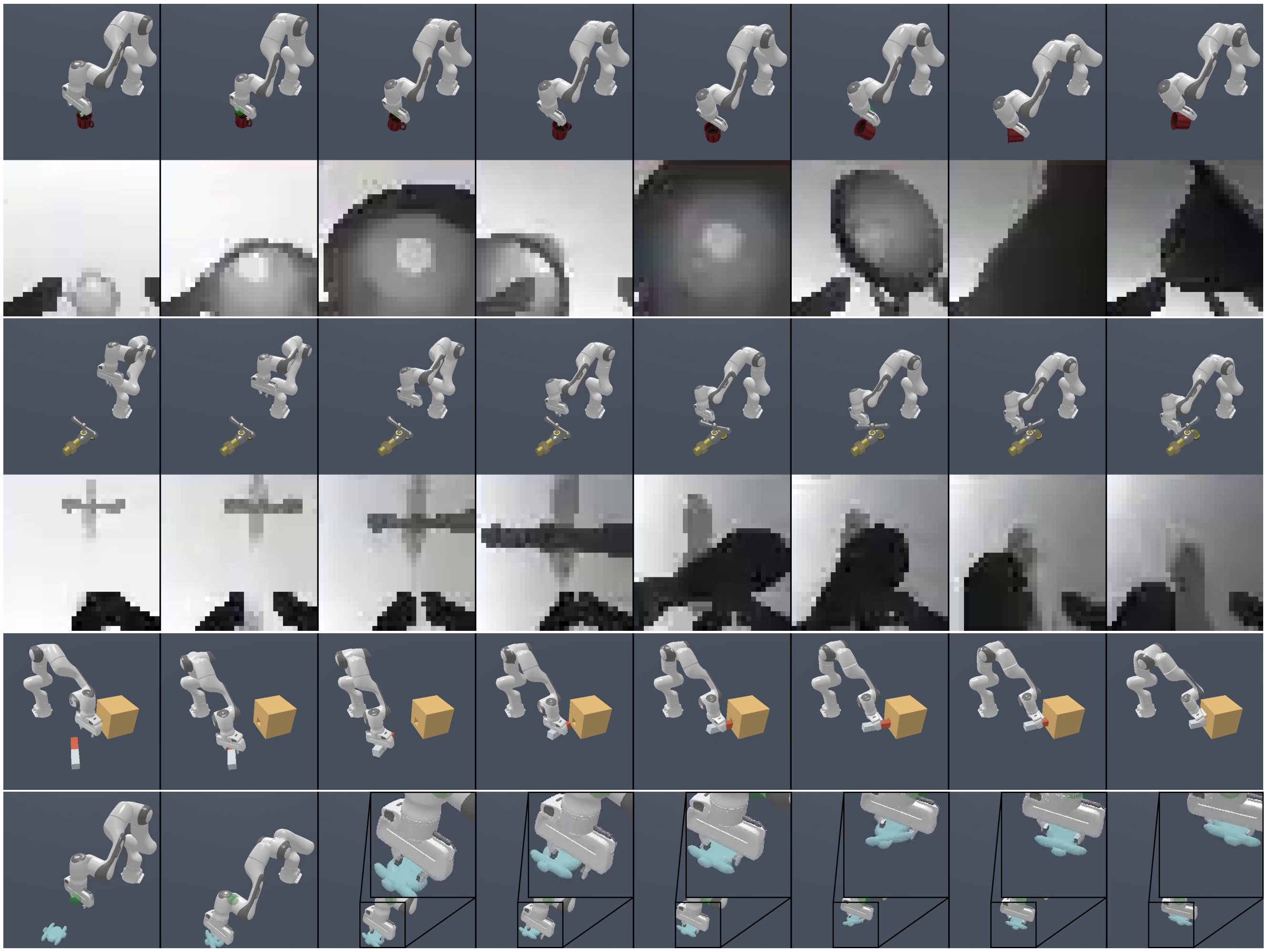


Figure 1. Visualization of an action trajectory by *RMA*^{2} in each of the four tasks. The top two trajectories also depict the corresponding low-resolution depth images as seen by the adapter module. We highlight a few interesting behaviors. In the first trajectory, for the Pick & Place task (YCB dataset), the agent first attempts to pick up a cup by the rim. This fails because the rim, in this instance of randomization, is too wide for its gripper. The agent then reattempted by grasping it by the handle, which succeeded. In the second trajectory, from the Faucet Turning task, we see the agent did not grasp the handle, but only pushed it with one finger to rotate it. The depth image shows the precise positioning of the end effector. In the third trajectory, we see the agent did not aim correctly for insertion on the first attempt. This is due to the external disturbances applied to the peg, and the fact that the hole has a very small clearance at the level of millimeters. But it succeeded after "jiggling" the peg around the correct position, a strategy that mimics human behavior. In the fourth trajectory, Pick & Place (EGAD dataset), the agent attempts to pick up a previously-unseen EGAD object. The object is too wide for the agent to grasp it from the top, as it lays flat on the floor (a zoomed in inset picture is shown). The agent picks up the object by pressing the left side of the object with its left finger and inserting its right finger beneath the object, which is a fair strategy to pick up a flat object.

ing is required. We aim to bring the generalization ability of RMA to a broad spectrum of manipulation tasks involving rigid bodies, such as pick-and-place operations, peg insertion, and faucet or lever turning.

We achieve this through several contributions:

1. We propose category and instance dictionaries as a strong proxy for geometry-aware manipulation (Sec. 3.2.1), which is crucial to learn policies that are not transferable across objects, e.g. grasping handles in different positions.

2. We also propose to use a depth convolutional neural network to estimate part of the privileged information about the environment, which performs object category and instance classification only *implicitly* (Sec. 3.3).

3. As far as we are aware, leveraging these modifications, we are the first to apply rapid motor adaptation to *general* object manipulation tasks with robot arms.

4. As a smaller contribution, we present a unified formalization of the objectives of the two learning phases of rapid

motor adaptation (Eq. (1) and Eq. (3)), which we believe can be useful in future developments based on this framework.

5. Through extensive experiments in four Maniskill2 tasks, we demonstrate that our method outperforms several strong baselines, including state-of-the-art techniques with automatic domain randomization [2, 15] and vision-based policies trained with domain randomization (Sec. 5).

2. Related Work

Classical control methods have long been the foundation for manipulation tasks [34]. These approaches, however, usually demand exacting models of both the robots and their operating environments, where even minor discrepancies can lead to performance degradation or task failure. Moreover, they face limitations in adapting to object variations in size, weight, and texture, requiring manual recalibration – a notable hindrance to scalability and flexibility in dynamic real-world applications.

In response, reinforcement learning (RL) with massive compute has emerged as a powerful alternative for learning manipulation skills [8, 10, 13, 14, 16, 21, 22, 27, 28, 32, 38]. Nonetheless, sample efficient generalization remains challenging. Techniques such as Domain Randomization and Dynamic Randomization [4, 26, 30, 35] have been adopted widely to leverage massive computational resources to train policies across varied environmental parameters, aiming to cultivate robustness to environmental shifts in a model-agnostic manner. Subsequent developments have refined this approach, introducing learning and adaptation mechanisms for randomization to enhance sample efficiency and generalization. For example, Zakharov et al. [42] uses a set of encoder-decoder “deception” modules to apply randomization to make the tasks difficult for the policy. Active Domain Randomization searches for the most informative environment variations within the given randomization ranges, where the informativeness is measured as the discrepancies of policy rollouts in randomized and non-randomized environment instances [23]. Automatic Domain Randomization (ADR) adapts the ranges for the randomization distribution based on the policy performance under the current randomization setting to help improve sample efficiency [2, 15]. We take ADR as one of the baselines for comparison.

Rather than having the policy be independent of the environment parameters, we can condition the policy on privileged parameters in simulation, conceptually related to system identification in control theory [17, 40, 41]. For instance, during deployment, physics parameters can be inferred through a trained module [40] or optimized directly by evolutionary algorithms [41]. However, inferring the exact parameters may not always be feasible or optimal for generalization.

Recently, Rapid Motor Adaptation (RMA) has presented a novel approach by learning to predict low-dimensional embeddings of environment parameters, demonstrating remarkably sample-efficient generalization in locomotion and in-hand manipulation tasks [20, 29]. In locomotion, an agent trained entirely in simulation was able to traverse through changing terrains, with changing payloads, and with wear and tear, while using solely proprioception. Building on this, Qi et al. [29] extend RMA to robotic in-hand rotation. They demonstrated that the controller, trained entirely in simulation on only cylindrical objects, can be directly deployed to a real robot hand to rotate dozens of objects with diverse sizes, shapes, and weights over one axis. Despite its potential, RMA’s application to general manipulation, where object states and goals vary from episode to episode, remains non-trivial.

3. Method

Our work extends RMA [20, 29] to perform object manipulation with robot arms. The key novelties are to condition the policy on diverse manipulation goals, and to visually infer object properties from depth images, which requires several modifications.

The main idea of RMA is to train a policy with a high amount of *domain randomization*, which is possible as long as the policy is conditioned on privileged information about the random environment parameters. Then in a second phase, an adapter is learned that estimates the privileged information from readily-available inputs, such as the history of a robot’s joints (proprioception). This allows the policy to exhibit highly-specific behaviours for different environments and situations, such as how to deal with low friction or high masses, without direct access to these factors during deployment. In our work, these factors are extended to include factors relevant to object-manipulation (e.g., what are we manipulating?), and RMA is extended beyond just proprioception, which cannot estimate object-manipulation factors prior to robot-object contact.

3.1. Policy Training Phase

The object manipulation task is formulated as a Markov Decision Process (MDP) [5]. A simulator, parameterized by environment parameters $e \sim \mathcal{E}$ (e.g. robot dimensions and masses), expresses a transition probability P_e that advances the simulation’s state s_t to the next time step s_{t+1}. The reinforcement learning objective is then to maximize the expected future discounted reward r (measuring how well the manipulation goal is attained), when sampling trajectories $(s_0, a_1, s_1, a_2, \ldots)$ by recursive application of P_e, and tak-

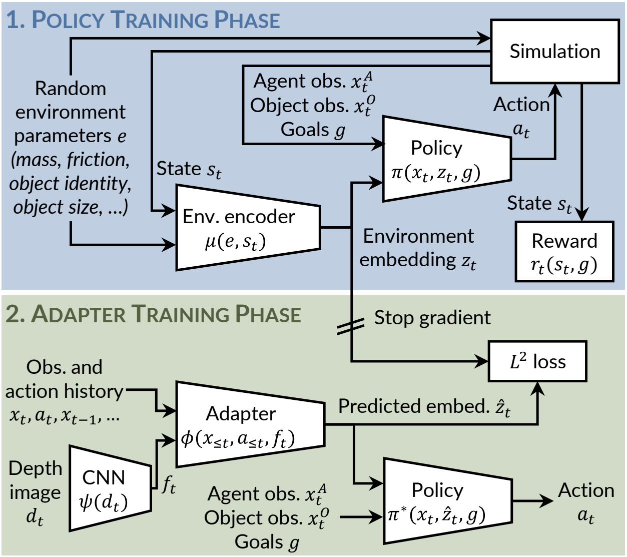

1. Policy Training Phase
Simulation
Random environment parameters e (mass, friction, object identity, object size, ...)
Agent obs. x_t^A
Object obs. x_t^O
Goals g
Action a_t
Policy $\pi(x_t, z_t, g)$
State s_t
Env. encoder $\mu(e, s_t)$
State s_t
Environment embedding z_t
Reward $r_t(s_t, g)$
2. Adapter Training Phase
Stop gradient
L^2 loss
Obs. and action history $x_t, a_t, x_{t-1}, \ldots$
Adapter $\phi(x_{\leq t}, a_{\leq t}, f_t)$
Predicted embed. $\hat{z}_t$
Depth image d_t
CNN $\psi(d_t)$
f_t
Agent obs. x_t^A
Object obs. x_t^O
Goals g
Policy $\pi^*(x_t, \hat{z}_t, g)$
Action a_t

Figure 2. Overview of the proposed training procedure, which consists of 2 phases. In the first phase, a *conditional policy* π is trained to maximize a reward (e.g. move an object to a given position or orientation), given observations x_t (e.g. joint angles), a goal description g and privileged information about the environment e, s_t. The environment is randomized (e.g. varying mass or object identities), so an environment encoder μ is trained jointly to distill this privileged information into an embedding z_t. In the 2nd phase, the policy π and encoder μ are frozen, and an adapter ϕ and CNN ψ are trained with a L^2 loss to predict the privileged information in z_t from just a history of observations (e.g. past dynamic behaviour) and a depth image d_t (e.g. object appearance). The adapter, CNN and policy can be deployed to perform adaptive manipulation directly from observations and depth images.

ing actions a_t that are chosen by the learned policy π:

$$\pi^*, \mu^* = \operatorname*{argmax}_{\pi,\mu} \mathop{\mathbb{E}}_{\substack{e\sim\mathcal{E}\\ g\sim\mathcal{G}}} \left[\mathop{\mathbb{E}}_{s_{t+1}\sim P_e(\cdot|s_t,a_t)} \left[\sum_{t'=0}^{T-1} \gamma^{t'} r(s_{t'}, g) \right] \right]$$

$$\text{with} \quad \begin{aligned} s_0 &\sim P_e(s_0), & x_t &= o(s_t), \\ a_t &= \pi(x_t, z_t, g), & z_t &= \mu(e, s_t), \end{aligned} \tag{1}$$

where T is the length of the simulation, $\mathcal{G}$ is a distribution over goals (e.g. a desired object position or orientation), $\mathcal{E}$ is a distribution over physical parameters (domain randomization), $0 < \gamma < 1$ is a discount factor to stabilize training, and o is an observation function modeling the fact that the policy does not have full access to the hidden state s_t. The policy is not conditioned on the physical parameters e directly, but rather on an *environment embedding* z, which is a (possibly compressed) view of those parameters, output by an environment encoder μ. It may also include other privileged (generally unobserved) information from the simulation state s_t. An illustration is in Fig. 2 (top half).

Eq. (1) is optimized using Proximal Policy Optimization (PPO) [31]. Both the policy π and the environment encoder μ are multi-layer perceptrons (MLPs). Note that at this stage we have obtained a policy π^* that can cope with different environment conditions, but the requirement for the environment parameters e and privileged state information s_t prevents its direct application in practice.

3.2. Privileged and Observable Information

Before moving on to the next training phase, it is worth discussing the observations x_t and privileged information (e, s_t) that the policy is conditioned on.

1. The *observations* x_t are the angle of each degree-of-freedom of the robot arm, and the position (but not orientation) of the object. Both represent measurements during deployment – the output of proprioception (e.g. wheel encoders) and a standard object detector (e.g. vision-based).

2. The *environment parameters* e represent generally unknown or hard-to-estimate quantities that are used to initialize the simulation, and are constant throughout: the manipulated object's shape, scale, mass and friction coefficient.

3. The *privileged state information* s_t contains all physical variables, some of which could be useful for learning, and so we would like to encourage the model to estimate them. In our setting, we condition μ on the object's rotation in 3D, and whether there is contact on each finger (binary variables). Both are only available in simulation.

However, these are still not enough for successful grasping, which also depends on the exact geometry of the object (Sec. 4). We will address this in the following section.

3.2.1 Category and Instance Dictionaries

We propose to encode geometry only implicitly, with instance and category dictionaries of learnable embeddings as proxies for geometry knowledge. We train a dictionary of learnable embeddings (similar to word embedding vocabularies in language models [6]), with a vector u_i for the ith object instance, and a vector c_j for the jth object category. These are initialized randomly, and concatenated with the physical environmental parameters e_{phys} (Sec. 3.2):

$$e(i) = \big(e_{\text{phys}}(i), u_i, c_{\text{cat}(i)}\big) \tag{2}$$

where $\text{cat}(i)$ retrieves the index of the category of object instance i. While this information is not available during deployment, it is no different from the other privileged physical parameters e_{phys}. This encoding allows us to estimate both kinds of privileged information with the same method, presented in the next section.

3.3. Adapter Training Phase

In order to estimate the environment embedding z with readily-available information, instead of privileged environment parameters e, the second phase aims to train an *adapter* ϕ that is conditioned on past observations $x_{\leq t}$ and actions $a_{\leq t}$. Note that it is unlikely that observations based purely on proprioception (joint angles) will carry information about objects' identities prior to manipulating them (Eq. (2)). This necessitates another input modality, to allow conditioning on object categories and instances (albeit indirectly). We choose depth images d_t from an arm-mounted camera, which should reveal properties such as an object's size or the orientation of graspable features, such as handles. These are processed by a convolutional neural network (CNN) ψ before being passed to the adapter ϕ. The overall objective then becomes:

$$\begin{aligned} \phi^*,\psi^* = \underset{\pi,\psi}{\operatorname{argmin}} \underset{\substack{e\sim\mathcal{E} \\ g\sim\mathcal{G}}}{\mathbb{E}} & \left[\underset{\substack{s_{t+1}\sim \\ P_e(\cdot|s_t,a_t)}}{\mathbb{E}} \left[\sum_{t'=0}^{T-1} \|\mu^*(e,s_{t'}) - \hat{z}_{t'}\|^2 \right] \right] \\ \text{with} \quad s_0 \sim P_e(s_0), \quad & a_t = \pi^*(x_t,\hat{z}_t,g), \qquad x_t = o(s_t), \\ f_t = \psi(d_t), \quad & \hat{z}_t = \phi(x_{\leq t},a_{\leq t},f_t), \end{aligned} \tag{3}$$

where $\hat{z}_t$ is the estimated environment embedding. Note that the optimal policy π^* and environment encoder μ^* from the first phase are used but kept frozen (i.e. not minimized over). An illustration is in Fig. 2 (bottom half).

Eq. (3) is optimized using standard back-propagation (namely Adam [18]). During deployment, only the trained depth CNN ψ^*, adapter ϕ^* and policy π^* are used.

3.4. Environments

We train our agents in a customized variant of *ManiSkill2* environments [12], with additional environmental randomization (Sec. 3.2). We show an illustration of each task in Fig. 1. These tasks are:

1. Pick and Place YCB and EGAD objects. The agent picks up a random object from the YCB dataset [7] (78 objects), or the EGAD dataset [24] (2281 objects), and places it at a point uniformly sampled from the reachable 3D space.
2. Peg Insertion. The agent picks up a cuboid-shaped peg on the table, and inserts at least 50% of it into a gap.
3. Faucet Turning. The agent turns a faucet handle by a variable angle, with a random faucet from the 60-object PartNet-Mobility dataset [37].

The selected tasks exemplify a broad spectrum of goal specifications. The first two have only a positional target, while Peg Insertion has both positional and rotational specifications. It also has partial constraints on the moving trajectory for the peg to be successfully inserted into the hole, rather than simply matching a target pose. Faucet Turning requires rotating (to varied angles) a faucet handle (of varied shapes; see Appendix B for examples). This skill is representative of other useful "twisting" motions, such as rotating screwdrivers, or unscrewing caps to open containers.

We use the default task-specific dense rewards offered by *Maniskill2* environments [12], which are composed of simple metrics such as distances between entities or whether the object is grasped for training the agents (see Appendix C for an overview).

4. Experiment Design

Simulation Setup. We use the Franka Emika Panda robot arm, a widely used 7-DOF manipulator with torque sensors in each joint known for its dexterity and precision. The arm is controlled using position control at a frequency of 20 Hz. Complementing the arm is a two-finger gripper, which serves as the end-effector for object manipulation tasks. To convert target position commands into actuator torques, we utilize a Proportional-Derivative (PD) controller with stiffness and damping coefficients $K_p = 4.0$ and $K_d = 0.2$, respectively. These can also be varied and added to the list of environment variation parameters in future work.

We utilize the ManiSkill2 environments [12] constructed atop the Sapien simulator [37]. During training, 50 independent environments run concurrently, where each episode has a length of 50 control steps; in testing, each episode has a maximum length of 200 control steps. The simulation operates at a frequency of 120 Hz while the control policy operates at 20 Hz. This setup provides a robust and versatile platform for evaluating the performance of our algorithms, and for extending it to mobile robots and to real-world manipulation tasks in future work.

Environment Setup. As introduced on a high level in Sec. 3, we incorporate three types of randomization into each environment for learning a generalizable policy. In each run, the parameters are sampled from a uniform distribution parameterized by the boundary values (see Appendix A). For evaluation of agent generalization, the ranges of environmental variations, observation noise, and external disturbances are widened during testing. Specifically, we increased the low and high values of the environment variations and external disturbance distribution by 0.8 and 1.2 during testing, respectively. We scale both the boundary parameters for the observation noise distribution by 1.2.

External disturbances are forces applied onto an object's center of mass when it is grasped by the robot. Following [4, 29], we implement it as follows. At each control step, we sample from a Bernoulli distribution with probability p whether to apply such a force to the object. If true, we apply a randomly sampled force to the object, which is then decayed by 0.8 at each control step. To sample the force to be applied, we first sample a direction vector from a 3-dimensional Gaussian distribution with mean 0, standard deviation 0.1, and scaled it to have an L2 norm of 1. The force is then scaled by the object mass and a force scale parameter sampled (see Appendix A). When a new force is sampled, the residual force from the previous time step is overwritten.

Differing tasks naturally yield variations in the privileged information e_t and the goal state g_t, while the object and agent state shapes are the same across tasks. Specifically, all three tasks share the agent state $x_t^a \in \mathbb{R}^{32}$ which includes the 9-dimensional position and velocity of its joints, and the 7-dimensional pose of its base and Tool Center Point (TCP).

The object state $x_t^o \in \mathbb{R}^6$ is a concatenation of the object position, and $\|{}^{\text{tcp}}x_t^{\text{obj}}\|$ – the distance between the TCP and the object center. We assume the object position is output by an off-the-shelf perception module with imperfect accuracy. Alternatively, this could also be part of the privileged information whose embeddings can be estimated from the depth-based perception in phase 2 of the training.

The privileged environment information $e_t \in \mathbb{R}^{71}$ is a concatenation of object dimension, $e_t^{\text{dim}} \in \mathbb{R}^3$; object density, $e_t^{\text{dens}} \in \mathbb{R}$; friction coefficient, $e_t^{\text{fric}} \in \mathbb{R}$; the magnitude of the impulse applied by the left and right finger of the gripper $e_t^{\text{impl}} \in \mathbb{R}^2$; and a 64-dimensional embedding for the type and token variable for the object identity, $e_t^{\text{typ}}, e_t^{\text{tok}} \in \mathbb{R}^{32}$. For Faucet Turning, the object dimension is replaced with the rotational axis of the handle that is targeted by the task to rotate $e_t^{\text{axis}} \in \mathbb{R}^3$.

The goal state representation naturally varies by task. In Pick and Place with YCB and EGAD objects, $g_t \in \mathbb{R}^9$ consists of the target 3-dimensional position of the object $g_t^{\text{pos}} \in \mathbb{R}^3$ and ${}^{\text{tcp}}x_t^{\text{goal}}, {}^{\text{obj}}x_t^{\text{goal}} \in \mathbb{R}^3$ which refers to the distance in position between the TCP and the goal, and the object and the goal, respectively. The derived variables assist the policy by extracting useful information that guides actions, which simplifies the task of learning. For Peg Insertion, the goal $g_t \in \mathbb{R}^{13}$ includes the pose of the target hole $g_t^{\text{pos}} \in \mathbb{R}^7$. And it also contains the ${}^{\text{tcp}}x_t^{\text{goal}}, {}^{\text{obj}}x_t^{\text{goal}} \in \mathbb{R}^3$ as in Pick and Place. For Faucet Turning, the goal $g_t \in \mathbb{R}^2$ specifies the 1-dimensional angle to rotate the handle with.

We train separate policies for each of the tasks to explore the feasibility and generalization of single-task agents. In future works, we wish to explore the possibility of multi-task agents with the hope that knowledge about dexterous movements can be shared across different tasks, accelerating the learning process.

Baselines and Ablations. We compare our model, dubbed *RMA*^{2}, against the following ablations and baselines. Each comparison is designed to highlight a different aspect of our design. The alternative models include:

1. **Oracle Adaptation** (*Oracle*). This model uses the ground truth extrinsic vector z_t generated by the environment encoder as opposed to the estimated $\hat{z}_t$. Because it relies on ground-truth access to privileged information, this alternative model could never actually run in the real world, but serves as an upper bound on adaptation performance.
2. **Domain Randomization with state-based policy** (*DR*). This baseline implements basic domain randomization, trained using the same randomization scheme but without the privileged information [35]. This comparison serves to test the value of adaptation, by replacing it with a policy that does not adapt but aims to be robust across environment variations.
3. **Domain Randomization with vision-based policy** (*DR+Vi*). This baseline uses depth-based perception rather than state-based info (as for *DR*), similar to [4][1].
4. **Automatic Domain Randomization** (*ADR*). This baseline uses ADR to generate learning curricula for improved efficiency, as done in a number of recent works [2, 15].
5. **Without Object Embedding** (*NoOE*). This model omits the two-part object embedding during training and, as a result, remains unaware of the identity of the object being manipulated. This variation assesses the benefit of incorporating object type-token identity into the privileged information.
6. **No Vision in Adaptation** (*NoVA*). This ablation removes depth vision when predicting extrinsics $\hat{z}_t$, similar to the adaptation in previous RMA works [20, 29].

Metrics. We adopt the following metrics to evaluate the models, each giving a different lens on model performance. The results, based on these metrics, are averaged across three random seeds, with each seed's result averaged across 5000 episodes:

1. **Success Rate** (SR). This gauges the proficiency of the agent in performing the assigned task. An episode is deemed successful if the agent meets the task's objective as defined in ManiSkill2 [12]. For example, for the Pick and Place task, success is attained when the object is positioned within 2.5 cm of the target location, with the robot remaining static.
2. **Episode Length** (EL). This measures the time taken by the agent to complete the task, with a cap set at 200 steps. Shorter length signifies a more efficient task completion.

Architecture and Training Details. In policy training, we use a 3-layer MLP for Environment encoder and a 4-layer MLP for Policy. In adapter training, we use a CNN with 3 2D convolutional layers and 4 1D convolutional layers for the depth image and state-action history, respectively. This sensory information is then integrated by a 2-layer MLP in Adapter. The parameters are optimized with the Adam optimizer [18].

[1] We also experimented with DR+Vi+Proprioception but achieved similar performance as DR+Vi.

We use a curriculum learning approach to facilitate the learning process. This curriculum linearly amplifies the magnitude of three types of randomization in our environment–environment variations, external disturbances, and observation noise, up to a threshold.

We train each agent on an Nvidia A100 GPU and 16 CPUs until convergence, or a maximum of 7 days.

5. Experiment Results and Analysis

We show example trajectory of *RMA*^{2} in each task in Fig. 1 and the experiment results of it and the baselines in Tab. 1. We see that *Oracle* consistently achieves the highest success rate and lowest episode length across the tasks, which is expected given its privileged access to the simulation's parameters and state. Our method, *RMA*^{2}, is consistently the best performing agent in the evaluation while being real-world deployable. The two ablations, *NoOE* and *NoVa* closely follow *RMA*^{2}'s performance. This highlights the significance of each of the design choices. Their performance is sometimes better than the agents trained with domain randomization, but not always – both object dictionaries and depth conditioning are necessary to achieve the best result.

5.1. Pick & Place task – YCB objects dataset

At the task of picking and placing objects sampled from the YCB dataset (see Fig. 1, row 1 for an example), *DR* and *ADR* exhibit a negligible success rate in the allotted time, underscoring our method's proficiency in reducing sample complexity and enhancing practical task learnability (see Tab. 1, columns 2 and 3). The vision-based *DR+Vi* achieves better results than *DR* but is about 4 times slower than the state-based method to complete the same number of training steps, such as *RMA*^{2} and *DR*. The ablation *NoVA* outperforms *NoOE*, showing the value of vision-based adaptation.

5.2. Faucet Turning task

In the Faucet Turning task (see Fig. 1 row 2 for an example trajectory), *ADR* outperforms *DR*, benefiting from the dynamically generated curriculum (Tab. 1, column 4 and 5). *DR+Vi* did not converge and scored lower than *DR* after 7 days of training, as it was much more computationally-expensive to train. There is a larger gap between *RMA*^{2} and *Oracle* than in Pick and Place with YCB objects (1.6% vs 13.5%), which is reflected in the larger adaptation loss (.08 vs .04). We hypothesize that the movement of the gripper camera increases training complexity, which could potentially be addressed with a fixed camera, but this might bring additional challenges in "hand-eye" coordination.

5.3. Peg Insertion task

Overall, Peg Insertion is the most challenging task as the hole has only a 3 mm clearance on the box and the task is successful only if half of the peg is inserted, while the equivalent task in other benchmarks [39] only requires the peg head to approach the surface of the hole (refer to the third row of Fig. 1 for a sample trajectory). In this task, we removed the *NoOE* ablation as there is only one object shape, the cuboid-shaped peg, in the task.

ADR performs worse than *DR*, which likely indicates a suboptimal hyperparameter setting for this task (Tab. 1, column 6 and 7). *DR+Vi* achieves 0.0% accuracy again due to it reaching the timeout before making any progress in the task, indicating that direct visual policy learning may be too difficult when the task requires high precision to even receive a reward.

5.4. Extrapolation of Policies from YCB to EGAD Dataset

EGAD is a collection of more than 2000 geometrically unique object generated using evolutionary algorithms specifically for evaluating robotic grasping and manipulation [24] (see Fig. 3 (a) for an illustration and Appendix B). We evaluated the agents trained on the YCB dataset directly on this to evaluate the effect of shift in object shape distribution, with an example trajectory shown in Fig. 1 row 4.

The *Oracle* agent is not applicable here because the trained object type-token embeddings are available only for the objects it has been trained on. Notably, as tabulated in the last two columns of Tab. 1, *RMA*^{2}'s success rate is higher than its performance on the YCB dataset by a 16.7% margin, while *DR+Vi*'s performance is only 0.3% higher than its counterpart on the YCB objects. This highlights the greater generalization performance for our method compared to domain randomization methods, which require the randomization distribution to be well-tuned to the distribution of the task that the policy is ultimately deployed to.

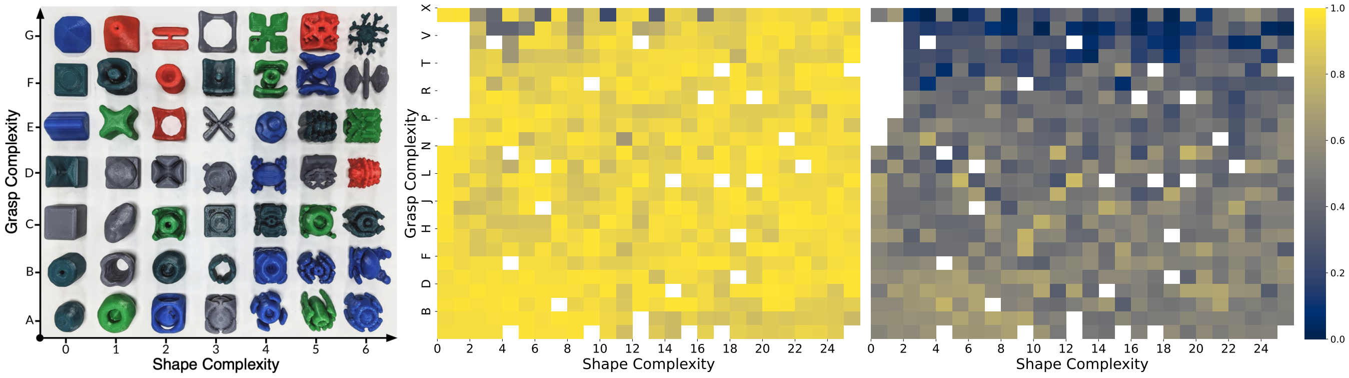
In Fig. 3 (b), we present the per object success rate for *RMA*^{2} and *DR+Vi*. Overall, the heatmap for *RMA*^{2} exhibits a consistently brighter tone, indicating a generally higher success rate. Notably, the *RMA*^{2} heatmap shows a pronounced darkness in areas where shape complexity is low yet grasp complexity is high, which is not as apparent when both complexities are elevated. In contrast, *DR+Vi* demonstrates a darker region across the spectrum of high grasp complexity, indicating a more uniform challenge in these conditions. Despite both approaches utilizing a CNN, we hypothesize that the deliberate inductive bias in *RMA*^{2}, which only attempts to predict an environment embedding that is useful for conditioning a successful policy, allows it to generalize better to geometrically more complex shapes.

6. Conclusion

In this work, we presented Rapid Motor Adaptation for Robot Manipulator Arms (*RMA*^{2}). By incorporating a

	Pick & Place task (YCB)		Faucet Turning task		Peg Insertion task		Pick & Place task (EGAD)	
Method	SR ↑	EL ↓	SR ↑	EL ↓	SR ↑	EL ↓	SR ↑	EL ↓
Oracle	75.4 ± 0.6	64.2 ± 1.4	76.2 ± 0.4	70.0 ± 0.3	55.4 ± 5.6	111.9 ± 10.4	–	–
DR	0.3 ± 0.1	199.7 ± 0.0	48.9 ± 3.1	176.1 ± 4.5	45.6 ± 21.6	142.2 ± 34.3	1.0 ± 0.0	199.9 ± 0.0
DR+Vi	35.2 ± 3.7	138.1 ± 6.6	14.7 ± 3.2	169.6 ± 3.8	0.0 ± 0.0	200.0 ± 0.0	35.5 ± 4.5	138.6 ± 7.2
ADR	1.6 ± 2.3	198.1 ± 2.1	51.8 ± 1.8	110.1 ± 2.5	14.0 ± 19.0	177.6 ± 30.5	3.0 ± 3.5	196.4 ± 4.3
NoOE	70.4 ± 0.4	77.7 ± 2.9	57.6 ± 0.4	108.3 ± 1.0	–	–	88.1 ± 0.1	44.3 ± 0.9
NoVA	68.1 ± 0.8	84.4 ± 0.5	47.0 ± 0.7	182.6 ± 1.8	48.4 ± 9.8	133.6 ± 43.9	87.5 ± 3.4	52.6 ± 8.8
RMA^2	$\mathbf{73.8 \pm 4.5}$	$\mathbf{72.1 \pm 0.8}$	$\mathbf{62.7 \pm 0.5}$	$\mathbf{88.1 \pm 0.8}$	$\mathbf{51.6 \pm 6.7}$	$\mathbf{127.3 \pm 10.8}$	$\mathbf{90.5 \pm 2.8}$	$\mathbf{40.1 \pm 6.1}$

Table 1. Evaluation results of our model and the baselines in simulation for the 4 tasks that we evaluate. The *Oracle* agent is highlighted in gray, as it is not real-world applicable due to its reliance on privileged information. The methods are Domain Randomization (DR) [35], a reactive vision-based RL method (DR+Vi) [4], Automatic Domain Randomization (ADR) [2, 15], an ablation of our method without object embeddings (NoOE), and our method without depth vision in the adapter (NoVA), i.e. simple RMA [20, 29]. The best performance in each column is bolded. See Sec. 5 for more details.



(a)

(b)

Figure 3. (a) Example objects from the EGAD dataset, sorted by grasp and shape complexity. This illustrates the array of diverse shapes. The horizontal axis indicates ascending shape complexity, while the vertical axis corresponds to increasing grasp complexity. (b) Fine-grained evaluation of the performance of *RMA*^{2} (left) and *DR+Vi* (right) on Maniskills2's Pick & Place task, with EGAD objects. The color coding reflects the success rate (bright yellow for 100%, dark blue for 0%), averaged over 500 runs. The white cells corresponds to objects that are not in the dataset for this task.

category-instance dictionary, paying deliberate attention to environmental parameters in base policy training and utilizing low-resolution depth vision during adaptation training, our policy demonstrated superior generalization performance and sample efficiency across four challenging ManiSkill2 tasks compared to the baselines. We believe these principles can be leveraged for efficient learning of other complex manipulation skills.

Looking ahead, we see several promising avenues for further research. 1) A natural next step is to learn a multitask motor skill policy that encourages knowledge sharing across an even broader range of tasks, which could further improve adaptability and learning efficiency. 2) Building on the state-based observations, an interesting extension would be to support variable numbers of objects in the environment. 3) We observe in tasks such as faucet turning that there is a performance gap between *Oracle* and *RMA*^{2}, which suggests that there is room for improving the adapter's estimate of the environment embedding, potentially by including other modalities or more sophisticated visual networks. 4) Finally, low-level skills could seamlessly interoperate with high-level task planners, or hierarchical RL methods, to develop more versatile and adept embodied AI agents capable of achieving long-horizon tasks in the real world.

Acknowledgements The authors would like to express their gratitude to Doug Morrison for the permission to reuse their figure. We acknowledge the generous support of the Royal Academy of Engineering (RF\201819\18\163). YL is particularly thankful for the funding provided by the Cambridge Trust and an unrestricted gift from Huawei.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 1

[2] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. 3, 6, 8

[3] Peter Anderson, Ayush Shrivastava, Joanne Truong, Arjun Majumdar, Devi Parikh, Dhruv Batra, and Stefan Lee. Sim-to-real transfer for vision-and-language navigation. In *Conference on Robot Learning*, pages 671–681. PMLR, 2021. 1

[4] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020. 3, 5, 6, 8

[5] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957. 3

[6] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003. 4

[7] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015. 5, 2

[8] Chen Chen, Hsieh-Yu Li, Xuewen Zhang, Xiang Liu, and U-Xuan Tan. Towards robotic picking of targets with background distractors using deep reinforcement learning. In *2019 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*, pages 166–171. IEEE, 2019. 3

[9] Peter I Corke, Witold Jachimczyk, and Remo Pillat. *Robotics, vision and control: fundamental algorithms in MATLAB*. Springer, 2011. 1

[10] Justin Fu, Sergey Levine, and Pieter Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4019–4026. IEEE, 2016. 3

[11] Nishad Gothoskar, Marco Cusumano-Towner, Ben Zinberg, Matin Ghavamizadeh, Falk Pollok, Austin Garrett, Josh Tenenbaum, Dan Gutfreund, and Vikash Mansinghka. 3dp3: 3d scene perception via probabilistic programming. *Advances in Neural Information Processing Systems*, 34:9600–9612, 2021. 1

[12] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023. 5, 6, 2

[13] Marcus Gualtieri and Robert Platt. Learning 6-dof grasping and pick-place using attention focus. In *Conference on Robot Learning*, pages 477–486. PMLR, 2018. 3

[14] Marcus Gualtieri, Andreas Ten Pas, and Robert Platt. Pick and place without geometric object models. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 7433–7440. IEEE, 2018. 3

[15] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, et al. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5977–5984. IEEE, 2023. 3, 6, 8

[16] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR, 2018. 3

[17] Yiannis Karayiannidis, Christian Smith, Danica Kragic, et al. Adaptive control for pivoting with visual and tactile feedback. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 399–406. IEEE, 2016. 3

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5, 6

[19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 1

[20] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021. 1, 3, 6, 8

[21] Boyao Li, Tao Lu, Jiayi Li, Ning Lu, Yinghao Cai, and Shuo Wang. Acder: Augmented curiosity-driven experience replay. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4218–4224. IEEE, 2020. 3

[22] Jeffrey Mahler and Ken Goldberg. Learning deep policies for robot bin picking by simulating robust grasping sequences. In *Conference on robot learning*, pages 515–524. PMLR, 2017. 3

[23] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020. 3

[24] Douglas Morrison, Peter Corke, and Jürgen Leitner. Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation. *IEEE Robotics and Automation Letters*, 5(3):4368–4375, 2020. 5, 7

[25] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 1

[26] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control

with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018. 3

[27] Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017. 3

[28] Ameya Pore and Gerardo Aragon-Camarasa. On simple reactive neural networks for behaviour-based reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7477–7483. IEEE, 2020. 3

[29] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning*, pages 1722–1732. PMLR, 2023. 3, 5, 6, 8

[30] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016. 3

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4

[32] Adarsh Sehgal, Hung La, Sushil Louis, and Hai Nguyen. Deep reinforcement learning using genetic algorithm for parameter optimization. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 596–601. IEEE, 2019. 3

[33] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 1

[34] Russ Tedrake. *Robotic Manipulation*. 2023. 3

[35] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017. 3, 6, 8

[36] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. 1

[37] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020. 5, 2

[38] Yuchen Xiao, Sammie Katt, Andreas ten Pas, Shengjian Chen, and Christopher Amato. Online planning for target object search in clutter under partial observability. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8241–8247. IEEE, 2019. 3

[39] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020. 7

[40] Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017. 3

[41] Wenhao Yu, C Karen Liu, and Greg Turk. Policy transfer with strategy optimization. *arXiv preprint arXiv:1810.05751*, 2018. 3

[42] Sergey Zakharov, Wadim Kehl, and Slobodan Ilic. Deceptionnet: Network-driven domain randomization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 532–541, 2019. 3

[43] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pages 737–744. IEEE, 2020. 1