Visual Reinforcement Learning with Self-Supervised 3D Representations

Yanjie Ze*[1] Nicklas Hansen*[2] Yinbo Chen[2] Mohit Jain[2] Xiaolong Wang[2]

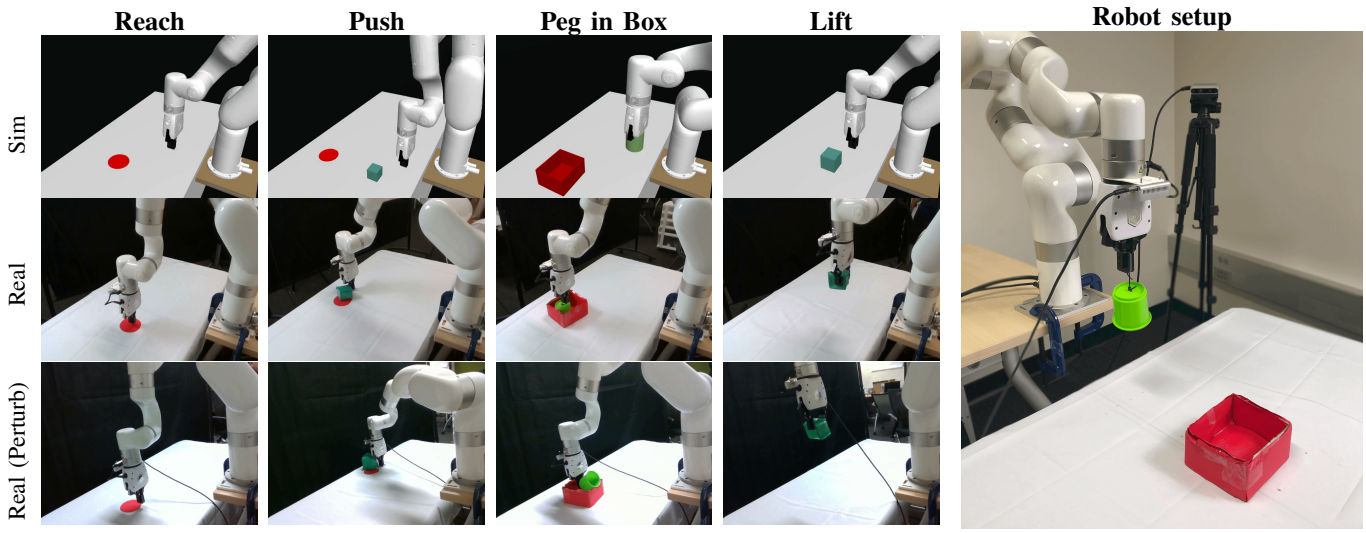

Fig. 1: **Overview of sim-to-real tasks.** We consider four tasks for our sim-to-real experiments: *(1) reach*, *(2) push*, *(3) peg in box*, and *(4) lift*. Observations are captured by a static over-the-shoulder camera (pictured). We visualize the initial configuration of robot and objects in simulation and the success in the real world. We consider transfer to two distinct real-world setups with varying degrees of similarity to the simulation, namely in terms of camera view and lighting.

***Abstract*—A prominent approach to visual Reinforcement Learning (RL) is to learn an internal state representation using self-supervised methods, which has the potential benefit of improved sample-efficiency and generalization through additional learning signal and inductive biases. However, while the real world is inherently 3D, prior efforts have largely been focused on leveraging 2D computer vision techniques as auxiliary self-supervision. In this work, we present a unified framework for self-supervised learning of 3D representations for motor control. Our proposed framework consists of two phases: a *pretraining* phase where a deep voxel-based 3D autoencoder is pretrained on a large object-centric dataset, and a *finetuning* phase where the representation is jointly finetuned together with RL on in-domain data. We empirically show that our method enjoys improved sample efficiency compared to 2D representation learning methods. Additionally, our learned policies transfer zero-shot to a real robot setup with only approximate geometric correspondence, and successfully solve motor control tasks that involve grasping and lifting from *a single, uncalibrated RGB camera*. Code and videos are available at https://yanjieze.com/3d4rl/.**

Manuscript received: December, 04, 2022; Revised February, 26, 2023; Accepted March, 03, 2023.

*Equal contribution. [1]Shanghai Jiao Tong University, Shanghai, China, work done while an intern at University of California San Diego. [2]University of California San Diego, CA, USA. Correspondence at xiw012@ucsd.edu.

This paper was recommended for publication by Editor Jens Kober upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported, in part, by Amazon Research Award and gifts from Qualcomm.

Digital Object Identifier (DOI): see top of this page.

***Index Terms*—Reinforcement Learning; Representation Learning; Deep Learning for Visual Perception**

I. Introduction

While deep Reinforcement Learning (RL) has proven to be a powerful framework for complex and high-dimensional control problems, most notable successes have been in problem settings either with access to fully observable states [1]–[3], or settings where partial observability through 2D image observations (*visual* RL) suffice, e.g., playing video games [4]. While potential applications of visual RL are far broader, it has historically been challenging to deploy in areas such as robotics, in part due to the complexity of controlling from high-dimensional observations.

A prominent approach is to tackle the resulting complexity by learning a good representation of the world, which reduces the information gap that stems from partial observability. Leveraging techniques such as self-supervised objectives for joint representation learning together with RL has been found to improve both sample efficiency [5], [6] and generalization [7]–[9] of RL in control tasks. Recently, researchers also discover training RL from embeddings produced by pretrained frozen visual encoders trained on external datasets can match

the performance of *tabula rasa* (from scratch) representations while requiring less in-domain data [10], [11].

Yet, efforts have largely been focused on applying successful techniques from 2D computer vision to control problems. However, our world is inherently 3D and agents will arguably need to perceive it as such in order to tackle the enormous complexity of real world environments [12]–[14]. For example, a robot manipulating objects may encounter challenges such as partial occlusion and geometric shape understanding, neither of which are easily captured by 2D images without prior knowledge or strong inductive biases [15], [16].

In this paper, we propose a 3D representation learning framework for RL that includes both a pretraining phase using external data and a joint training phase using in-domain data collected by the RL agent. Figure 2 provides an overview of our method. In the first phase, we learn a generalizable 3D representation using a repurposed *video autoencoder* [17] that performs 3D deep voxel-based novel view synthesis without assuming access to ground-truth cameras. For pretraining, we leverage Common Objects in 3D (CO3D) [18] – a large-scale object-centric 3D dataset – to steer learning towards object-centric scene representations suitable for our downstream manipulation tasks. In the second phase, we finetune the learned representation together with policy learning on in-domain data collected by online interaction. Concretely, a 2D encoder produces a 2D feature map that is shared between the two tasks and the 3D voxel is generated upon this feature map. For the view synthesis task, we apply a random affine transformation to the voxel representation, corresponding to a change of camera pose, and task a 3D decoder with reconstructing the scene from the novel view. This encourages the network to learn the underlying scene geometry. The policy learns to predict actions from the 2D feature map, and we backpropagate gradients from both objectives to the shared encoder for in-domain finetuning. We emphasize the different views are only utilized in training and the learned model **only requires a single view for deployment**.

To validate our method, we consider a set of vision-based Meta-World [19] tasks, as well as four robotic manipulation tasks with camera feedback both in simulation and the real world as shown in Figure 1. For the latter, we train policies in simulated environments, and transfer zero-shot to a real robot setup with only approximate geometric correspondence and an uncalibrated third-person RGB camera. We also demonstrate that our model is more robust to visual changes by using two variations of our real environment with different camera position, camera orientation, and lighting (bottom row in Figure 1). Compared to strong baselines that pretrain representations using 2D computer vision objectives, our method demonstrates improved sample efficiency during policy learning and transfers better to the real world despite environment perturbations. In summary, our contributions are three-fold,

- We propose a novel 3D representation learning framework for RL, using a view synthesis task and including a pretraining stage and a finetuning stage.
- Our method is evaluated on 9 simulation tasks and achieves good sample efficiency compared to 2D representations.
- Our learned policy transfers zero-shot to the real world successfully in both a non-perturbed setting and a perturbed setting, showing the robustness of our 3D representation to visual changes.

II. Related Work

Representation learning for RL. Learning good representations for vision-based RL is a well-studied problem. Prominent approaches include the use of learned dynamics models [20], [21], auxiliary objectives [9], [22], and data augmentation [23]–[25]. Recently, researchers have found that visual backbones pretrained using 2D computer vision objectives on large external datasets can produce useful features for control both in simulation and the real world [10], [11], [26]. Although these pretraining methods that use a *frozen* visual representation have shown initial success, *the domain gap between the RL task and the pretraining data is still non-negligible*. In this work we find that **jointly finetuning the visual backbone on in-domain data produces better representations for RL across different representations** including ours, ImageNet pretrained, and 2D self-supervised pretrained ones, which is a neglected factor in most previous works. Compared to these stronger, finetuned baselines, our method still performs significantly better, especially during sim-to-real transfer.

Learning 3D scene representations. Besides the aforementioned 2D-centric techniques, there are also prior efforts in learning 3D scene representations for RL, e.g. through differentiable 3D keypoints [27], [28], object-centric graphs [29], [30], latent 3D features [16], [17], [31], and neural radiance fields [32]–[34]. Notably, the proposed framework in [34] shares similarities with ours; however, their approach requires multi-view images (with perfect foreground segmentations) as input and only includes experiments in simplistic environments without any actual robots (the agent is simplified as a moving end-effector point), which makes real-world deployment challenging. Our method also learns latent 3D features, but in contrast to prior work we only use ***a single fixed view*** for policy inference, which makes our method both extendable and easy to deploy in the real world.

Sim-to-real transfer. Transferring policies learned in simulation to the real world is a hard problem for which a number of (largely orthogonal) approaches have been proposed. For example, domain randomization [9], [35] improves transfer by artificially widening the training data distribution. Alternatively, the simulation can be iteratively adjusted to match real world data [36], [37], the learned RL policy can be adapted by finetuning in the real world [21], [38], [39], or zero-shot transfer can be improved by learning a better representation [40]. We also consider the problem of sim-to-real transfer from the lens of representation learning due to its generality and not requiring real world data, which often relies on human labor for collection.

III. Background

Problem definition. We model agent and environment as a Markov Decision Process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where $\mathbf{s} \in \mathcal{S}$ are states, $\mathbf{a} \in \mathcal{A}$ are actions, $\mathcal{T} \colon \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$

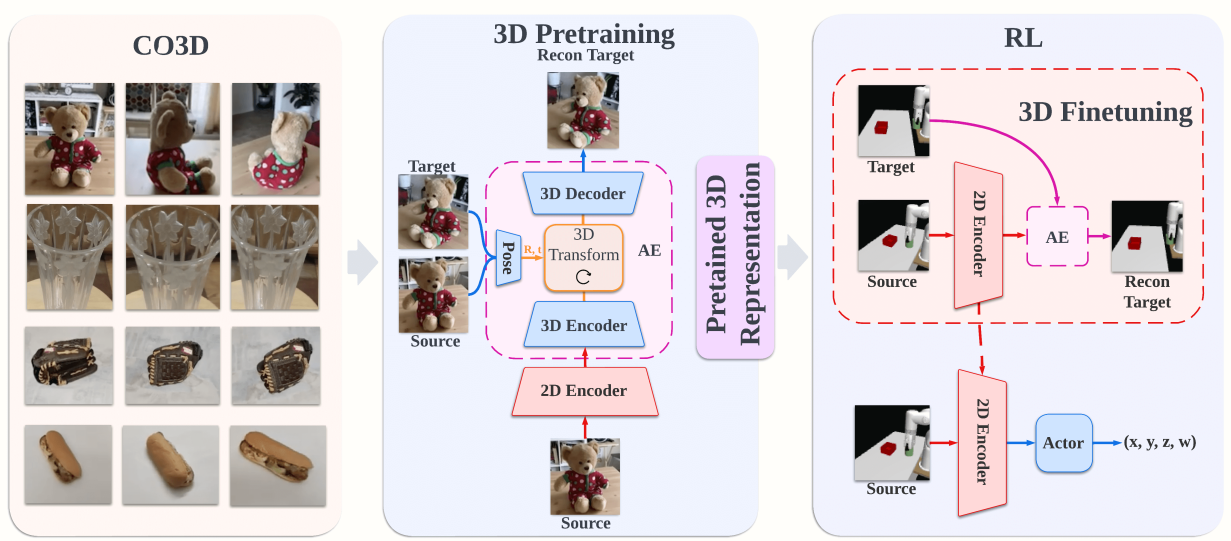

Fig. 2: **Overview of our approach.** We pretrain a 3D deep voxel-based auto-encoder on the *Common Objects in 3D* (CO3D) dataset. Then we train an RL policy in simulation using the learned representation as initialization and jointly finetune the representation with 3D and RL objectives on in-domain data collected by the RL agent.

is a transition function, $r \in \mathcal{R}$ are rewards, and $\gamma \in [0, 1)$ is a discount factor. The agent's goal is to learn a policy π that maximizes discounted cumulative rewards on $\mathcal{M}$. In visual RL, states $\mathbf{s}$ are generally unknown, but we can use image observations $\mathbf{o} \in \mathcal{O}$ in lieu of states, rendering it a Partially Observable MDP.

Soft Actor-Critic (SAC) [41] is an off-policy actor-critic algorithm that learns a stochastic policy π_θ and critic Q_θ from an iteratively grown dataset $\mathcal{D}$ collected by interaction. Throughout, we let θ denote the combined parameter vector. The critic is learned by minimizing the Bellman error

$$\mathcal{L}_Q(\theta; \mathcal{D}) = \mathbb{E}_{\mathbf{o},\mathbf{a},r,\mathbf{o}' \sim \mathcal{D}} \left[(Q_\theta(f_\theta(\mathbf{o}), \mathbf{a}) - (r + \gamma \mathcal{V}) \right] , \quad (1)$$

where $\mathcal{V} = Q_{\overline{\theta}}(f_{\overline{\theta}}(\mathbf{o}'), \mathbf{a}') - \alpha \log \pi_\theta(\mathbf{a}'|f_\theta(\mathbf{o}'))$ is the soft Q-target, $\overline{\theta}$ is a slow-moving average of θ, α is a learnable parameter balancing entropy maximization and value function optimization, and $\mathbf{o}' \sim \mathcal{T}(\mathbf{o}, \mathbf{a})$, $\mathbf{a}' \sim \pi_\theta(\cdot|f_\theta(\mathbf{o}'))$. π_θ learns to maximize an entropy-regularized expected return:

$$\mathcal{L}_\pi(\theta; \mathcal{D}) = \mathbb{E}_{\mathbf{o} \sim \mathcal{D}} \left[Q_\theta(f_\theta(\mathbf{o}), \mathbf{a}) - \alpha \log \pi_\theta(\mathbf{a}|f_\theta(\mathbf{o})) \right] , \quad (2)$$

for $\mathbf{a} \sim \pi_\theta(f_\theta(\mathbf{o}))$. Actions are sampled from π using a squashed Gaussian parameterization; see [41] for further details. In this work, we focus on learning a good representation f_θ for SAC, but we emphasize that our framework is fully agnostic to the underlying RL algorithm.

IV. Method

We propose a 3D representation learning framework for RL that includes both a pretraining phase using external data and a finetuning phase using in-domain data collected by an RL agent. An overview of our approach is shown in Figure 2.

A. Object-Centric 3D Pretraining

Our framework is implemented as a deep voxel-based 3D auto-encoder [17] that shares a 2D encoder with an RL policy. Given a view (image) of a 3D scene and an affine camera transformation, we task the 3D auto-encoder with reconstructing a 2D view of the scene after applying a transformation to the deep voxel representation. This task encourages the network to encode geometric scene information, which is beneficial for downstream control tasks.

Architecture. For brevity, we let θ denote the combined parameter vector of our network. A *source* view I_{src} is encoded by a 2D encoder f_θ to produce feature maps $Z = f_\theta(I_{\text{src}})$, $Z \in \mathbb{R}^{C \times H \times W}$. We then reshape Z into a 3D grid of dimensions $(C/D) \times D \times H \times W$ and upsample the reshaped feature maps using strided transposed 3D convolutions g_θ to obtain our final deep voxel representation $V = g_\theta(Z) = g_\theta(f_\theta(I_{\text{src}}))$. Now, let I_{tgt} denote a *target* view (used as reconstruction target) of the same scene as I_{src}. To obtain a camera transformation between I_{src} and I_{tgt} for our 3D reconstruction task, we learn an additional *PoseNet* $\mathcal{F}_{\text{pose}}$ that estimates the rotation between two views. This is necessary because common datasets do not have access to ground-truth cameras. $\mathcal{F}_{\text{pose}}$ takes the concatenation $[I_{\text{src}}, I_{\text{tgt}}]$ as input and predicts relative rotation parameterized by Euler angles $[\alpha, \beta, \gamma]^\top$ (from which we trivially obtain rotation matrix R) as well as translation $t = [x, y, z]^\top$, *i.e.*, $\mathcal{F}_{\text{pose}}(I_{\text{src}}, I_{\text{tgt}}) \in \mathbb{R}^6$. We transform V by R, t and obtain a warped grid $\hat{V} = T_{R,t}(V)$, and predict the target view I_{tgt} from $\hat{V}$ with a 3D decoder h_θ. The 3D network thus predicts I_{tgt} from I_{src} as $\hat{I}_{\text{tgt}} = h_\theta(T_{R,t}(g_\theta(f_\theta(I_{\text{src}}))))$, where R, t are obtained from $\mathcal{F}_{\text{pose}}(I_{\text{src}}, I_{\text{tgt}})$. Intuitively, the source view I_{src} is used to generate a 3D representation, and the target view I_{tgt} is used as reconstruction target to supervise the 3D representation.

Objective. To optimize the 3D auto-encoder and associated PoseNet, we adopt a ℓ_1-norm reconstruction loss

$$\mathcal{L}_{\text{recon}} \left(\hat{I}_{\text{tgt}}, I_{\text{tgt}} \right) = \lambda_{L1} \left\| \hat{I}_{tgt} - I_{\text{tgt}} \right\|_1 . \quad (3)$$

Training. We implement the 2D encoder f_θ as an ImageNet-

initialized ResNet18 and let the 3D encoder/decoder have relatively fewer parameters, such that the majority of trainable parameters are shared with the RL policy during finetuning. To steer learning of the encoder towards object-centric scene representations, we pretrain our network on 20 object categories from *Common Objects in 3D* (CO3D) [18], a large-scale object-centric 3D dataset. CO3D contains videos that rotate around objects and we only use raw frames. We emphasize that the video for pretraining is not limited to static scenes.

B. In-Domain Joint Training of 3D and RL

After the pretraining phase, we use the learned representation as initialization for training an RL policy, while we continue to jointly optimize the 3D objective together with RL using in-domain data collected by the RL agent. Specifically, we learn a policy network $\pi_\theta \colon \mathbb{R}^{C\times H\times W} \mapsto \mathcal{A}$ that takes feature maps $Z = f_\theta(I)$ from the pretrained 2D encoder f_θ as input (where I is an image observation) and outputs a continuous action. The motivation for our joint finetuning phase is two-fold: *(1)* finetuning with the 3D objective improves the 3D representation on in-domain data, and *(2)* finetuning with the RL objective improves feature extraction relevant for the task at hand. Figure 3 provides an overview of our joint training.

Optimizing 3D. Since our proposed 3D task requires at least two views of a scene, we design a static camera and another dynamic camera. Let I_{src} denote the image from the static (source) view and I_{tgt} denote the image from the dynamic (target) view, respectively. The 3D task is then to reconstruct I_{tgt} from I_{src}. We move the dynamic camera positioned with angle ϕ_d in a circular manner around the scene within an angle ϕ of the static camera, thus $\phi_d \in [0, \phi]$.

Optimizing RL. We train the RL agent by online interaction with a simulation environment, and store observed transitions in a replay buffer for joint optimization together with the 3D objective. To mitigate catastrophical forgetting in the 3D representation due to changes in the data distribution, we optimize the 3D network using a smaller learning rate than for RL. Formally, let λ_{ft} denote the finetuning scale, let lr_{3D} denote the learning rate for the 3D task, and let lr_{RL} denote the learning rate for RL. We then have $\text{lr}_{\text{3D}} = \lambda_{\text{ft}} \times \text{lr}_{\text{RL}}$.

V. Experiments

We validate our method on a set of precision-based robotic manipulation tasks from visual inputs. We report success rates over a set of pre-defined goal and object locations both in simulation and in the real world.

Robot setup is shown in Figure 1 *(right)*. We use an xArm robot equipped with a gripper, and observations are captured by a static third-person camera. The agent operates from 84×84 RGB camera observations, as well as the robot state including end-effector position and gripper aperture. We do *not* calibrate the camera. To estimate the robustness of representations, we consider *two* variants of our real-world setup of varying likeness to the simulation – we refer to these as *perturbed* and *non-perturbed* environments.

Baselines. We implement our method and all baselines using Soft Actor-Critic (SAC; [41]) as the backbone RL algorithm and use the same hyperparameters whenever applicable. Concretely, we consider the following baselines: *(i)* training an image-based SAC with a 4-layer ConvNet encoder from **Scratch**; *(ii)* replacing the encoder with a ResNet18 backbone pretrained by **ImageNet** classification; and *(iii)* a ResNet18 pretrained on ImageNet using the self-supervised **MoCo** [42] objective; *(iv)* **CURL** [6], a strong visual RL method that leverages data augmentation and contrastive learning. All methods use ± 4 random shift [24] and color jitter as data augmentation during RL, except for CURL that uses random crop augmentation as originally proposed.

Tasks. We experiment with **5** image-based tasks from Meta-World, as well as **4** manipulation tasks both in simulation and on physical hardware. We consider the following tasks in our sim-to-real experiments: *(1)* **reach** ($\mathcal{A} \in \mathbb{R}^3$), where the agent needs to position the gripper at the red goal, *(2)* **push** ($\mathcal{A} \in \mathbb{R}^2$), where the agent needs to push a green cube to the red goal, *(3)* **peg in box** ($\mathcal{A} \in \mathbb{R}^3$), where the agent needs to place a green peg inside a red box, and *(4)* **lift** ($\mathcal{A} \in \mathbb{R}^4$), where the agent needs to grasp and lift a green cube into the air. A trial is considered successful only when the goal is reached (*e.g.*, the peg is fully inside the box) within a fixed time limit of 20s (50 decision steps). We conduct an *extensive* set of real-world trials using 5 model seeds per method per task and evaluate each seed over 10 trials (5 for reach) on a set of predefined configurations for a total of **1300** trials: **700** trials for the setup close to the simulated environments and **600** trials for the perturbed real world setup; see Figure 1 *(left)* for the two setups.

A. Sample-Efficiency

We train for 500k environment steps across all xArm tasks and train for 1m environment steps across Meta-World tasks. Results for Meta-World tasks and xArm manipulation tasks are shown in Figure 4, totalling **9** tasks. We summarize our findings as follows:

From scratch training of SAC is generally a strong baseline, but the gap between this baseline and methods that use pretrained representations widens with increasing task difficulty. For example, the success rate of *from scratch* is close to that of our method in *coffee push* (Meta-World), while it fails to solve harder tasks like *coffee pull*.

MoCo vs. ImageNet pretraining. We find that MoCo generally leads to better downstream performance than pretraining with ImageNet classification, which is consistent with observations made in prior work [11], while the performance gap is relatively small for most tasks. we observe MoCo to be better on *basketball* (Meta-World) and *lift* (xArm) which both involve precise object manipulation. This finding suggests that self-supervised pretraining might produce better initializations for in-domain finetuning in precision-based control tasks.

3D vs. 2D representations. Our proposed method that uses a self-supervised 3D representation outperforms from-scratch training, pretrained 2D representations (MoCo), and 2D representations with a SSL auxiliary objective (CURL) across most tasks. Notably, our method enjoys large performance gains on challenging tasks such as *coffee pull* (Meta-World), *hammer* (Meta-World), and *lift* (xArm) that require spatial understanding.

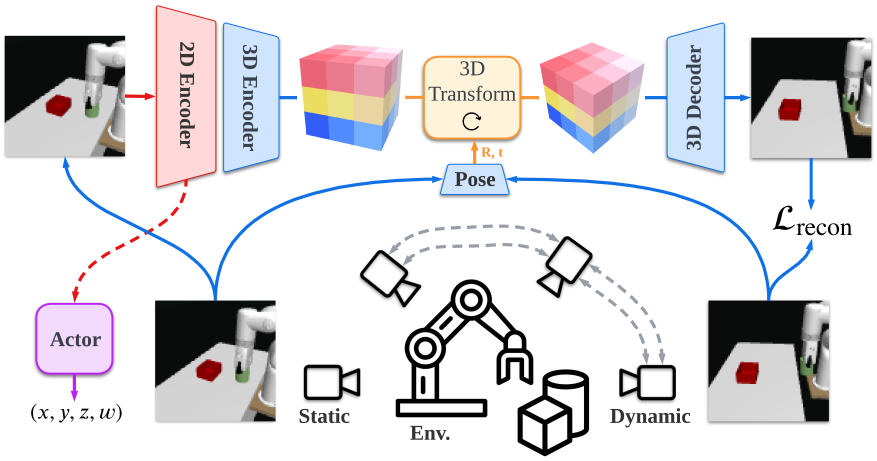

Fig. 3: **In-domain joint training of 3D and RL.** A static view is used as input to both 3D and RL and is encoded using a shared 2D encoder. The 3D autoencoder takes 2D features as input and reconstructs observations from a dynamic view.

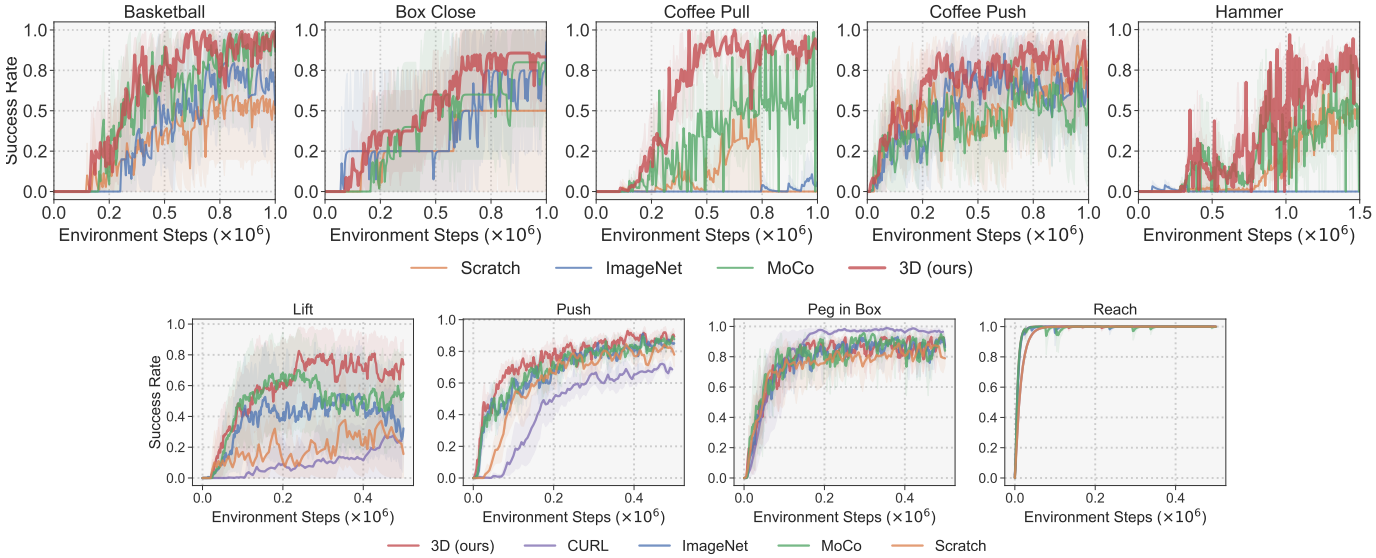

Fig. 4: **Learning curves (*Meta-World and xArm*).** Success rate of our method and baselines on *five* diverse image-based Meta-World tasks and four simulated xArm manipulation tasks. Mean of 5 seeds, shaded areas are 95% CIs. Our method achieves non-trivial success rates faster than other methods.

B. Sim-to-Real Transfer

We evaluate policies trained in simulation on physical hardware following the previously outlined evaluation procedure. For the *lift* task, we additionally report the grasping success rate in real. Results are shown in Table I. We observe a drop in success rates across the board when transferring learned policies to the real world relative to their simulation performance. However, the gap between simulation and real performances is generally lower for our 3D method than for baselines. For example, our method achieves a **46**% success rate on *lift* (vs. 64% in sim), whereas MoCo – the second-best method in sim – achieves only 20% success rate (vs. 51% in sim). While baseline performances differ in simulation, we do not find any single 2D method to consistently transfer better than the others. We thus attribute the sizable difference in transfer results between our method and the baselines to the learned 3D representation.

C. Robustness

We provide a more challenging evaluation in both simulation and the real world by adding further perturbations to the environments. Perturbations added to the simulation includes camera position and orientation, lighting, and the texture of objects and robot. Perturbation added to the real world include camera position and orientation, as well as lighting. We refer to the latter setting as *Real (Perturb)* and visualize it in Figure 1. Results are shown in Table II. We observe a drop in success rates across all methods due to the perturbation, while the perturbation effects are alleviated in our method. For example, our method still achieves **95**% success rate in perturbed simulation and **60**% success rate in perturbed real on *reach* whereas MoCo achieves only 86% in sim and 27% in real respectively. We also find that for 2D baselines there is no single method that outperforms others consistently. For example, ImageNet pretraining leads to better generalization

TABLE I: **Robotic manipulation results (*xArm*).** Success rate (in %) of our method and baselines. *(left)* results in simulation. *(right)* results when transferred zero-shot to physical hardware. We report mean and std. err. across 5 model seeds for all evaluations. Initial configurations are randomized.

Sim	Scratch	ImageNet	MoCo	3D (*ours*)
Reach	**100±0**	**100±0**	**100±0**	**100±0**
Push	65±16	74±15	74±14	**80±14**
Peg in Box	77±22	**82±18**	**82±17**	**82±17**
Grasp	–	–	–	–
Lift	20±34	40±40	51±40	**64±32**

Real	Scratch	ImageNet	MoCo	3D (*ours*)
Reach	84±12	**96±4**	80±11	**96±4**
Push	2±2	22±10	22±7	**48±9**
Peg in Box	40±14	62±20	50±15	**76±19**
Grasp	44±14	20±10	38±10	**62±14**
Lift	30±15	2±2	20±5	**46±19**

TABLE II: **Robotic manipulation results evaluated in perturbed environments (*xArm*).** Success rate (in %) of our method and baselines. *(left)* results in *perturbed* (P) simulation environments. *(right)* results when transferred zero-shot to perturbed real environments. We report mean and std. err. across 5 model seeds for all evaluations. Initial configurations are randomized.

Sim(P)	Scratch	ImageNet	MoCo	3D (*ours*)
Reach	76±10	96±8	86±14	**96±5**
Push	12±7	12±10	14±14	**24±21**
Peg in Box	20±20	22±13	24±7	**34±20**
Grasp	–	–	–	–
Lift	0±0	10±15	10±10	**16±8**

Real(P)	Scratch	ImageNet	MoCo	3D (*ours*)
Reach	26±12	48±12	27±12	**60±12**
Push	10±7	10±7	0±0	**33±17**
Peg in Box	18±11	28±14	20±6	**52±14**
Grasp	25±11	10±10	35±19	**40±15**
Lift	10±10	0±0	10±10	**25±11**

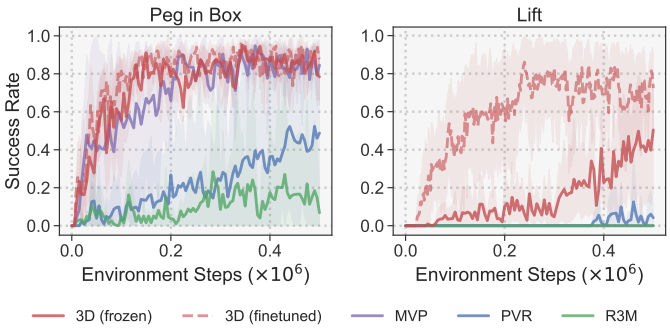

Fig. 5: **Success rate of different frozen visual representations.** We compare our 3D visual representation with MVP [10], PVR [11], and R3M [26] on *peg in box* and *lift*.

on *reach* while MoCo performs well on *lift*. The experiments demonstrate that our 3D visual representation is more robust to distribution shifts in the observation space.

D. Ablations

Frozen 3D visual representation. We compare our *frozen* 3D visual representation with the following pretrain methods for motor control: *(i)* **MVP** [10] which provides a pretrained vision transformer using masked auto-encoder on a joint Human-Object Interaction dataset; *(ii)* **PVR** [11] which utilizes a ResNet50 pretrained with MoCo on ImageNet; and *(iii)* **R3M** [26] which pretrains a ResNet50 with time contrastive learning and video-language alignment on the Ego4D human video dataset. The results are shown in Figure 5. We directly apply the public pretrained encoders provided by these works and all the methods are equipped with the same RL backbone. Our encoder uses fewer parameters (**11.47m**) than MVP (21.67m), PVR (23.51m), and R3M (23.51m). On *peg in box* our method could achieve high success rates in 500k steps and is comparable to MVP, while another two baselines learn much slower. On a more challenging task *lift*, only our method achieves meaningful accuracy and all other three methods have not yet. Thus our visual representation with much fewer parameters is very competitive to recent methods. In addition, we compare between the frozen 3D representation and the finetuned 3D representation and find that unfreezing the representation could give a more promising result. It is not surprising, but recent works [10], [11], [26] only focus on the frozen visual representation, which might neglect the power of end-to-end policy learning [43].

Finetuning with the 3D objective. We demonstrate the necessity of finetuning the 3D visual representation with in-domain data and our 3D objective, specifically the reconstruction loss. Figure 7 *(left)* shows that our method without 3D finetuning initially converges at a similar rate but to a lower accuracy in 500k steps. The finetuning scale for the 3D objective also matters, as shown in Figure 7 *(mid)*, where a smaller scale stabilizes the learning process. Table III provides quantitative and qualitative evaluation on the view synthesis results with 3D finetuning, demonstrating that it leads to more realistic and closer-to-original images.

3D Pretraining. We are also curious about whether 3D pretraining really helps if the visual representation has been trained with the 3D objective and the RL objective jointly. As shown in Figure 7 *(right)*, it is observed that without 3D pretraining the representation achieves much lower accuracy on *lift*. Compared to Figure 7 *(left)*, we could also find that the lack of 3D pretraining leads to more degradation of the success rates, showing that 3D pretraining is necessary.

Novel view synthesis in sim and real. We evaluate our method's 3D representation using qualitative and quantitative methods: *(i)* novel view synthesis results from simulated and real observations (Figure 6) and *(ii)* quantitative ablations

TABLE III: **Quantitative evaluation of novel view synthesis** in sim (S) and real (R) on *peg in box*. *(left two)* Different λ_{ft} for $\phi = 30°$. *(right two)* Different dynamic camera angle ϕ_d when $\phi = 30°$ and $\lambda_{\text{ft}} = 0.01$. Our method generalizes well to out-of-distribution camera angles.

λ_{ft} (S)	SSIM↑	PSNR↑	λ_{ft} (R)	SSIM↑	PSNR↑	ϕ_d (S)	SSIM↑	PSNR↑	ϕ_d (R)	SSIM↑	PSNR↑
0.00	8.69	0.22	0.00	8.28	0.28	15	**12.26**	**0.42**	15	**11.78**	**0.36**
0.01	11.34	**0.37**	0.01	10.49	0.31	30	11.34	0.37	30	10.49	0.31
0.10	11.75	0.35	0.10	11.20	0.34	45	11.68	0.37	45	9.94	0.24
1.00	**11.93**	**0.37**	1.00	**11.29**	**0.38**	60	10.13	0.33	60	8.72	0.20

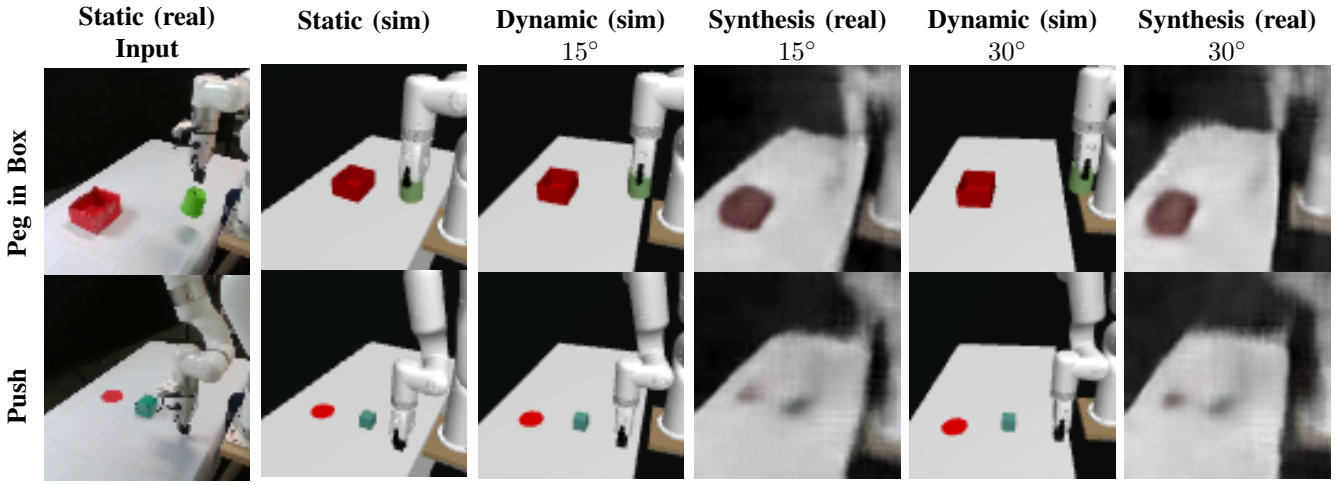

Fig. 6: **Novel view synthesis in real.** We use images in the real world to generate the deep voxel and use the static view and the dynamic in the simulation to predict the transformation and then reconstruct the novel view. We display the reconstruction results for $\phi_d = 15°, 30°$ in two tasks.

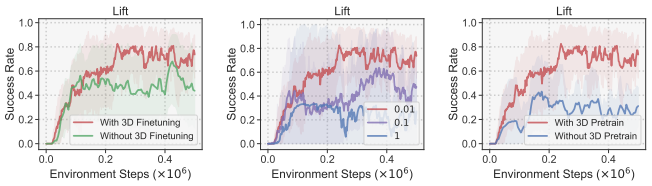

Fig. 7: *(left)* Success rate of our 3D method with and without 3D finetuning on *lift*. *(mid)* Success rate of different lr_{ft} on *lift*, where lr_{ft} is the finetuning scale for 3D auxiliary task such that the learning rate would be $\text{lr} \times \text{lr}_{\text{ft}}$. *(right)* Success rate of our 3D method with and without 3D pretraining.

to investigate the effect of important hyperparameters on reconstruction quality (Table III). Our method can synthesize meaningful reconstructions using real camera observations, even without prior exposure to our robot setup. For quantitative evaluation, we use Structural Similarity (SSIM) and Peak Signal-to-Noise Ratio (PSNR) metrics to evaluate different finetuning rates (λ_{ft}) and camera angles (ϕ_d) at $\phi = 30°$. Results show that a reduced learning rate does not harm reconstruction, and our method trained at $\phi = 30°$ generalizes well to out-of-distribution angles, up to 60°.

Camera pose estimation with PoseNet. Our PoseNet estimates the relative pose between two frames. We quantitatively evaluate such pose estimation results in the following. For a whole trajectory, we estimate the relative pose between the dynamic camera and the static camera for each timestep.

TABLE IV: **Quality of pose estimation for the *peg in box* task.** The table shows the root mean squared error (RMSE) of camera pose estimation given varying dynamic camera angles ϕ_d and varying finetuning coefficients λ_{ft}. We only train with $\phi_d = 30°$ and evaluate both in-domain (ID) and out-of-domain (OOD) performance. We observe that finetuning leads to smaller errors.

Pose Estimation Error (RMSE↓)				
$\phi_d/\lambda_{\text{ft}}$	Pretrain	0.01	0.10	1.00
15 (ID)	0.041	0.041	**0.040**	0.046
30 (ID)	0.066	0.059	**0.033**	0.041
45 (OOD)	0.120	0.064	0.046	**0.044**
60 (OOD)	0.174	**0.130**	0.132	0.133
avg	0.100	0.073	**0.063**	0.066

To align predicted camera pose trajectories with the ground truth [17], we apply Umeyama alignment [44] in deep voxel coordinate space. Our method is tested with in-domain and out-of-domain dynamic camera angles and various finetuning scales. Results (Table IV) show that our method reduces pose estimation error compared to CO3D-only pretrained networks. Our method could also generalize to 45 degrees with a small error equal to 0.064, nearly half of the one with only pretraining. Larger finetuning scales generally reduce error, but even small scale finetuning can improve results over the pretrained model.

VI. Conclusion

Our proposed 3D framework for pretraining and joint learning improves sample efficiency of reinforcement learning (RL) in simulation and successfully transfers to a real robot setup. This is, to the best of our knowledge, the first positive sim-to-real transfer result using pretrained 3D representations with RL. We find that learning 3D representations leads to significant gain in real robot performance and our representation is much more robust to the visual environment changes in the real world. We also compare to settings on RL with frozen features and show frozen 3D representation consistently outperforms state-of-the-art methods with frozen 2D representations. A possible limitation of our work is the requirement of multi-view inputs *during training*. While these are easy to obtain in simulation, it could be difficult to learn a multi-view representation in the real world. Exploring single-view training with a 3D-aware objective could therefore be an interesting direction for future research.

References

[1] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv*, 2016. 1

[2] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv*, 2017. 1

[3] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *IJRR*, 2020. 1

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv*, 2013. 1

[5] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," 2019. 1

[6] M. Laskin, A. Srinivas, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," in *ICML*, 2020. 1, 4

[7] I. Higgins, A. Pal, A. A. Rusu, L. Matthey, C. P. Burgess, A. Pritzel, M. M. Botvinick, C. Blundell, and A. Lerchner, "Darla: Improving zero-shot transfer in reinforcement learning," *arXiv*, 2017. 1

[8] A. Nair, V. H. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," in *NeurIPS*, 2018. 1

[9] N. Hansen and X. Wang, "Generalization in reinforcement learning by soft data augmentation," in *ICRA*, 2021. 1, 2

[10] T. Xiao, I. Radosavovic, T. Darrell, and J. Malik, "Masked visual pre-training for motor control," *arXiv*, 2022. 2, 6

[11] S. Parisi, A. Rajeswaran, S. Purushwalkam, and A. K. Gupta, "The unsurprising effectiveness of pre-trained vision models for control," *arXiv*, 2022. 2, 4, 6

[12] A. Dobbins, R. Jeo, J. Fiser, and J. Allman, "Distance modulation of neural activity in the visual cortex," *Science*, 1998. 2

[13] R. Cheng, A. Agarwal, and K. Fragkiadaki, "Reinforcement learning of active vision for manipulating objects under occlusions," in *CoRL*, 2018. 2

[14] I. Akinola, J. Varley, and D. Kalashnikov, "Learning precise 3d manipulation from multiple uncalibrated cameras," in *ICRA*. IEEE, 2020, pp. 4616–4622. 2

[15] C. Wang, R. Martín-Martín, D. Xu, J. Lv, C. Lu, L. Fei-Fei, S. Savarese, and Y. Zhu, "6-pack: Category-level 6d pose tracker with anchor-based keypoints," in *ICRA*. IEEE, 2020, pp. 10 059–10 066. 2

[16] H.-Y. F. Tung, R. Cheng, and K. Fragkiadaki, "Learning spatial common sense with geometry-aware recurrent networks," in *CVPR*, 2019. 2

[17] Z. Lai, S. Liu, A. A. Efros, and X. Wang, "Video autoencoder: self-supervised disentanglement of static 3d structure and motion," *arXiv*, 2021. 2, 3, 7

[18] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut, and D. Novotny, "Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction," in *ICCV*, 2021. 2, 4

[19] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *CoRL*, 2019. 2

[20] D. Hafner, T. P. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *arXiv*, 2020. 2

[21] N. Hansen, R. Jangir, Y. Sun, G. Alenyà, P. Abbeel, A. A. Efros, L. Pinto, and X. Wang, "Self-supervised policy adaptation during deployment," in *ICLR*, 2021. 2

[22] W. Ye, S. Liu, T. Kurutach, P. Abbeel, and Y. Gao, "Mastering atari games with limited data," *arXiv*, 2021. 2

[23] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *arXiv*, 2020. 2

[24] I. Kostrikov, D. Yarats, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," *ICLR*, 2020. 2, 4

[25] N. Hansen, H. Su, and X. Wang, "Stabilizing deep q-learning with convnets and vision transformers under data augmentation," *arXiv*, 2021. 2

[26] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3m: A universal visual representation for robot manipulation," *arXiv*, 2022. 2, 6

[27] B. Chen, P. Abbeel, and D. Pathak, "Unsupervised learning of visual 3d keypoints for control," *arXiv*, vol. abs/2106.07643, 2021. 2

[28] M. Jaritz, J. Gu, and H. Su, "Multi-view PointNet for 3D Scene Understanding," *arXiv*, 2019. 2

[29] H.-Y. F. Tung, X. Zhou, M. Prabhudesai, S. Lal, and K. Fragkiadaki, "3d-oes: Viewpoint-invariant object-factorized environment simulators," in *CoRL*, 2020. 2

[30] H. Qi, X. Wang, D. Pathak, Y. Ma, and J. Malik, "Learning long-term visual dynamics with region proposal interaction networks," in *ICLR*, 2021. 2

[31] C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. M. Botvinick, and A. Lerchner, "Monet: Unsupervised scene decomposition and representation," *arXiv*, 2019. 2

[32] Y. Li, S. Li, V. Sitzmann, P. Agrawal, and A. Torralba, "3d neural scene representations for visuomotor control," *arXiv*, 2021. 2

[33] J. Ichnowski, Y. Avigal, J. Kerr, and K. Goldberg, "Dex-neRF: Using a neural radiance field to grasp transparent objects," in *CoRL*, 2021. 2

[34] D. Driess, I. Schubert, P. Florence, Y. Li, and M. Toussaint, "Reinforcement learning with neural radiance fields," *arXiv*, 2022. 2

[35] K. Wang, B. Kang, J. Shao, and J. Feng, "Improving generalization in reinforcement learning with mixture regularization," *arXiv*, 2020. 2

[36] Y.-Y. Tsai, H. Xu, Z. Ding, C. Zhang, E. Johns, and B. Huang, "Droid: Minimizing the reality gap using single-shot human demonstration," *RA-L*, 2021. 2

[37] Y. Du, O. Watkins, T. Darrell, P. Abbeel, and D. Pathak, "Auto-tuned sim-to-real transfer," *ICRA*, 2021. 2

[38] R. Julian, B. Swanson, G. Sukhatme, S. Levine, C. Finn, and K. Hausman, "Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning," *arXiv*, 2020. 2

[39] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv*, 2021. 2

[40] R. Jangir, N. Hansen, S. Ghosal, M. Jain, and X. Wang, "Look closer: Bridging egocentric and third-person views with transformers for robotic manipulation," *RA-L*, 2022. 2

[41] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv*, 2018. 3, 4

[42] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *arXiv*, 2020. 4

[43] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *JMLR*, 2016. 6

[44] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, no. 04, pp. 376–380, 1991. 7