VISUAL REINFORCEMENT LEARNING WITH SELF-SUPERVISED 3D REPRESENTATIONS

Anonymous authors
Paper under double-blind review

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 in simulated manipulation tasks 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*. Videos are available at `https://3d4rl.github.io/`.

1 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 (Lillicrap et al., 2016; Silver et al., 2017; Andrychowicz et al., 2020), or settings where partial observability through 2D image observations (*visual* RL) suffice, e.g., playing video games (Mnih et al., 2013; Vinyals et al., 2019). 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 (Yarats et al., 2019; Srinivas et al., 2020) and generalization (Higgins et al., 2017; Nair et al., 2018; Hansen & Wang, 2021) 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 (Xiao et al., 2022; Parisi et al., 2022).

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 (Dobbins et al., 1998; Cheng et al., 2018a). 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 (Wang et al., 2020a; Tung et al., 2019b). These challenges are particularly pronounced in transfer settings, *e.g.*, sim-to-real, where an agent is trained in simulation using an imperfect model of the world and is expected to transfer to a real robot setup that it has never encountered before (Peng et al., 2018; Yan et al., 2017).

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

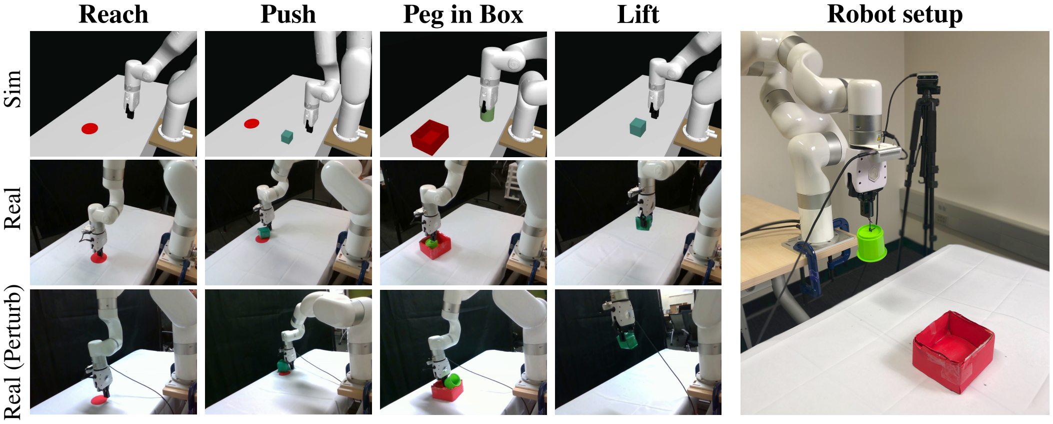


Figure 1: **Overview of sim-to-real tasks.** We consider four tasks for our sim-to-real experiments: *(1) reach*, where the agent needs to position the gripper at the red goal, *(2) push*, where the agent needs to push a green cube to the red goal, *(3) peg in box*, where the agent needs to place a green peg inside a red box, and *(4) lift*, where the agent needs to grasp and lift a green cube into the air. 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 real world.

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* (Lai et al., 2021) 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) (Reizenstein et al., 2021) – 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**, both in simulation and on the real robot.

To validate our method, we consider a set of vision-based Meta-World (Yu et al., 2019) 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. We find empirically that the primary aspects in which our method departs from prior work – 3D pretraining and in-domain joint training of 3D and RL – are crucial to its success.

2 Related Work

Representation learning for RL. Learning good representations for vision-based RL is a well-studied problem. Prominent approaches include the use of auto-encoders (Lange & Riedmiller, 2010; Finn et al., 2016; Higgins et al., 2017; Nair et al., 2018; Burgess et al., 2019; Yarats et al., 2019), learned dynamics models (Ha & Schmidhuber, 2018; Ebert et al., 2018; Dasari et al., 2019; Janner et al., 2019; Hafner et al., 2020; Hansen et al., 2021a), auxiliary objectives (Jaderberg et al., 2016; Srinivas et al., 2020; Yan et al., 2020; Schwarzer et al., 2021; Hansen & Wang, 2021; Stooke et al., 2021; Ye et al., 2021), abstractions (Battaglia et al., 2016; Kulkarni et al., 2019; Veerapaneni et al., 2019; Minderer et al., 2019; Deng et al., 2020), and data augmentation (Tobin et al., 2017; Peng et al., 2018; Laskin et al., 2020; Kostrikov et al., 2020; Hansen et al., 2021b). For example, Yarats et al. and Srinivas et al. show that jointly optimizing a visual encoder using a 2D autoencoder or contrastive objective, respectively, together with RL can improve sample-efficiency of visual RL. Recently, researchers have also 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 (Xiao et al., 2022; Parisi et al., 2022; Nair et al., 2022; Radosavovic et al.). In particular, Parisi et al. explores a variety of pretrained representations for control and find that the choice of pretraining objective is important to downstream performance. 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 propose a 3D object-centric pretraining recipe for control. We also 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 baseline approaches, 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 (Chen et al., 2021; Jaritz et al., 2019), object-centric graphs (Chang et al., 2017; Tung et al., 2020; Locatello et al., 2020; Qi et al., 2021), latent 3D features (Burgess et al., 2019; Tung et al., 2019a; Lai et al., 2021), and neural radiance fields (Li et al., 2021; Ichnowski et al., 2021). For example, Tung et al. learn a graph-based forward dynamics model of scene objects, and Lai et al. learn latent 3D scene features from passive indoor navigation videos with known camera trajectories. 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 (Tobin et al., 2017; Pinto et al., 2018; James et al., 2019; Lee et al., 2019; Ramos et al., 2019; Wang et al., 2020b; Hansen & Wang, 2021; Zhang et al., 2022) improves transfer by artificially widening the training data distribution. Alternatively, the simulation can be iteratively adjusted to match real world data (Chebotar et al., 2019; Hanna et al., 2021; Tsai et al., 2021; Du et al., 2021), the learned RL policy can be adapted by finetuning in the real world (Traoré et al., 2019; Julian et al., 2020; Hansen et al., 2021a; Kumar et al., 2021), or zero-shot transfer can be improved by learning a better representation (Yan et al., 2017; James et al., 2019; Jangir et al., 2022). 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.

3 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, $\mathcal{T} \colon \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ 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 (Kaelbling et al., 1998). By learning a good internal representation f, we can hope to increase mutual information between encoded observations and ground-truth states of the system $\mathcal{I}(f(\mathbf{o}); \mathbf{s}),\ \forall \mathbf{o} \in \mathcal{O}, \mathbf{s} \in \mathcal{S}$. While f can be, e.g., an off-the-shelf visual backbone pretrained on external data, prior knowledge about the distribution of $\mathcal{O}, \mathcal{S}$ (such as in-domain samples) may aid in capturing information of the state.

Soft Actor-Critic (SAC) (Haarnoja et al., 2018) 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

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

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:

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

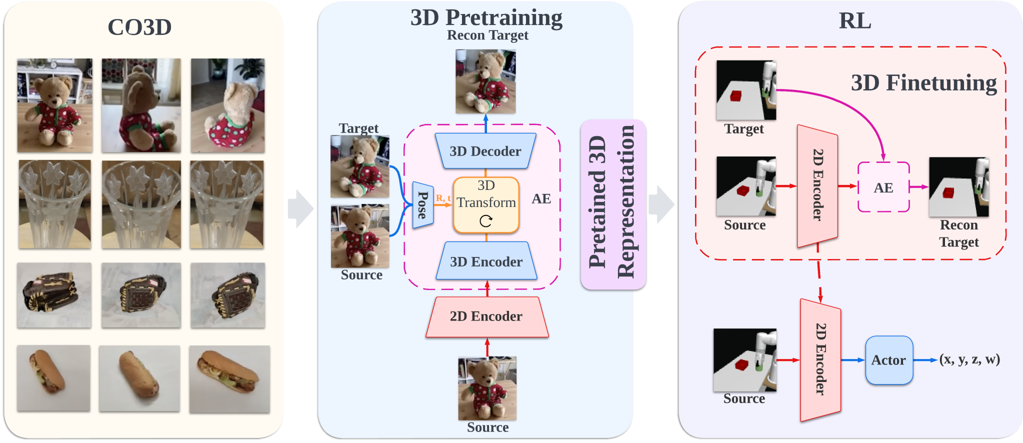


Figure 2: **Overview of our approach.** *(left)* We pretrain a 3D deep voxel-based auto-encoder on the *Common Objects in 3D* (CO3D) dataset, a large object-centric dataset. *(right)* We train an RL policy in simulation using the learned representation as initialization, and jointly finetune the representation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

for $\mathbf{a} \sim \pi_\theta(f_\theta(\mathbf{o}))$. Actions are sampled from π using a squashed Gaussian parameterization; see (Haarnoja et al., 2018) 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.

4 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. By leveraging a large-scale external dataset for 3D pretraining, our learned representation can quickly be finetuned together with RL to produce quality deep voxel-based scene representations for single-camera visual control. Figure 2 provides an overview of our approach. We now describe an instantiation of our framework with SAC as the backbone learning algorithm.

4.1 Object-Centric 3D Pretraining

Our framework is implemented as a deep voxel-based 3D auto-encoder (Lai et al., 2021) 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. We first detail training of the 3D auto-encoder, and then discuss the RL policy and finetuning phase to Section 4.2.

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. In summary, 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 $\mathcal{F}_{\text{pose}}(I_{\text{src}}, I_{\text{tgt}})$.

Objective. To optimize the 3D auto-encoder and associated PoseNet, we adopt an objective similar to that of (Lai et al., 2021). Specifically, our objective is a ℓ_1-norm reconstruction loss

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

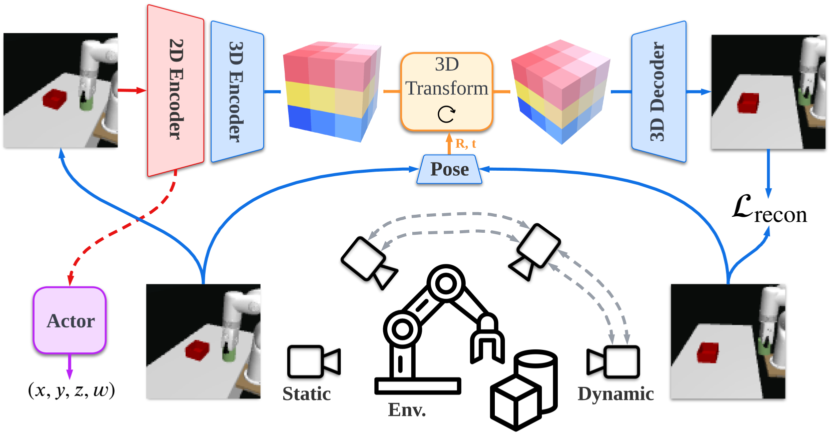


Figure 3: **In-domain joint training of 3D and RL.** A static view is used as input to both 3D and RL and is encoded using a shared 2D encoder. The 3D autoencoder takes 2D features as input and reconstructs observations from a dynamic view that moves around the scene in a circular manner.

with additional losses as in Lai et al. (2021) that serve to improve reconstructions in pretraining, *e.g.*, an adversarial loss. We do not find it necessary to use these additional losses during policy learning.

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 suitable for our downstream manipulation tasks, we choose to pretrain our network on 20 object categories from *Common Objects in 3D* (CO3D) (Reizenstein et al., 2021), a large-scale object-centric 3D dataset. CO3D contains videos that rotate around objects and we only use raw frames for training. We emphasize that the video for pretraining is not limited to static scene videos. Since our training only takes a few frames close in time as inputs, for most videos in the real world we can assume the structure does not change drastically in a short time, thus the pretraining can be applied to general videos as well. During training, we sample two views $I_{\text{src}}, I_{\text{tgt}}$ of a scene and jointly optimize the 3D auto-encoder and PoseNet using the objective in Equation 3. To encourage geometrically plausible latent interpolations between views, we constrain the interval between any two sampled views by a bound b, such that the number of skipped frames are randomized but bounded.

4.2 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 $\pi_\theta : \mathbb{R}^{C\times H\times W} \mapsto \mathcal{A}$ 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. During this phase, we optimize f_θ using gradients from both the 3D objective and RL, but use a reduced learning rate to mitigate catastrophical forgetting. 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. In the following, we describe the training in more detail.

Optimizing 3D. Since our proposed 3D task requires at least two views of a scene, we design a static camera and another dynamic camera which moves flexibly in a predefined manner. 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 $\phi_d \in [0, \phi]$. The concrete design is remained in supplementary materials.

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 $\text{lr}_{\text{3D}} = \lambda_{\text{ft}} \times \text{lr}_{\text{RL}}$.

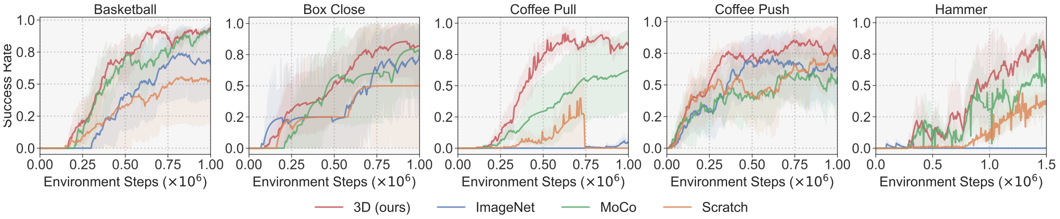


Figure 4: **Learning curves (*Meta-World*)**. Success rate of our method and baselines on *five* diverse image-based Meta-World tasks. Mean of 5 seeds, shaded areas are 95% CIs. Our method achieves non-trivial success rates faster than other methods. See supplementary material for samples.

5 Experiments

We validate our method on a set of precision-based robotic manipulation tasks from visual inputs. Our 3D method and baselines are trained entirely in simulation using dense rewards and randomized robot poses, goals and object positions. Evaluation of trained policies is conducted both in simulation and on a real robot setup. In our real-world evaluation, policies are transferred zero-shot, *i.e.*, without finetuning nor access to real world data prior to deployment. We report success rates over a set of pre-defined goal and object locations both in simulation and in the real world. *We are committed to releasing our code.* We now describe our experimental setup and discuss our findings.

Robot setup. Our real robot setup is shown in Figure 1 *(right)*. We use an xArm robot equipped with a gripper in our real-world experiments, and observations are captured by a static third-person camera. Simulated environments correspond only approximately to the real setup. The agent operates from 84×84 RGB camera observations, as well as the robot state including end-effector position and gripper aperture. We use position control for end-effector motion and torque control for the gripper and do *not* calibrate the camera. Agents are trained by online RL in simulation using dense rewards and are transferred zero-shot to the real setup, *i.e.*, no finetuning nor access to real data. 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; (Haarnoja et al., 2018)) 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 (Russakovsky et al., 2015); and *(iii)* a ResNet18 pretrained on ImageNet using the self-supervised **MoCo** objective. All methods use ±4 random shift (Kostrikov et al., 2020) and color jitter as data augmentation during RL.

Tasks. We experiment with **5** image-based tasks from Meta-World, as well as **4** manipulation tasks both in simulation and on physical hardware. Visualization of all tasks are provided in Appendix F. 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 and supplementary material for videos and further details.

5.1 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 are shown in Figure 4 and results for xArm manipulation tasks are shown in Figure 5, 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*.

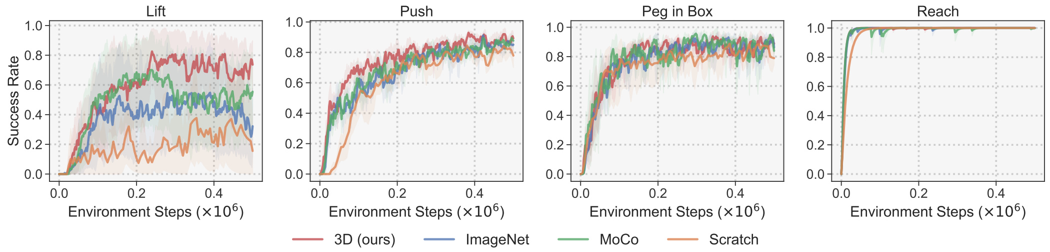


Figure 5: **Learning curves (*xArm*).** Success rate of our method and baselines on the four simulated xArm manipulation tasks shown in Figure 1. Mean of 5 seeds, shaded areas are 95% CIs.

Table 1: **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

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 (Parisi et al., 2022), 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 both from scratch training and pretrained 2D representations 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.

5.2 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 1. 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.

5.3 Robustness

We provide a more challenging evaluation in both the simulation and the real world, by adding more perturbation into the environment to make the observation much more *out-of-distribution*. The perturbation added to the simulation includes the camera position, the camera orientation, the lighting, the texture of objects, and the texture of the robot arm, as visualized in Figure 14. The perturbation added to the real world includes the camera position, the camera orientation, the lighting, and the background, as visualized in Figure 1. The results are shown in Table 2. 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 on *reach* while MoCo performs well on *lift*. The overall experiments demonstrate that our 3D visual representation is more robust to distribution shift in the observation space and better in generalization.

Table 2: **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**

Table 3: **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↑
0.00	8.69	0.22
0.01	11.34	**0.37**
0.10	11.75	0.35
1.00	**11.93**	**0.37**

λ_{ft} (`R`)	SSIM↑	PSNR↑
0.00	8.28	0.28
0.01	10.49	0.31
0.10	11.20	0.34
1.00	**11.29**	**0.38**

ϕ_d (`S`)	SSIM↑	PSNR↑
15	**12.26**	**0.42**
30	11.34	0.37
45	11.68	0.37
60	10.13	0.33

ϕ_d (`R`)	SSIM↑	PSNR↑
15	**11.78**	**0.36**
30	10.49	0.31
45	9.94	0.24
60	8.72	0.20

5.4 Ablations

Frozen 3D visual representation. We compare our *frozen* 3D visual representation with the following pretrain methods for motor control: *(i)* **MVP** (Xiao et al., 2022) which provides a pretrained vision transformer using masked auto-encoder on a joint Human-Object Interaction dataset; *(ii)* **PVR** (Parisi et al., 2022) which utilizes a ResNet50 pretrained with MoCo on ImageNet; and *(iii)* **R3M** (Nair et al., 2022) which pretrains a ResNet50 with time contrastive learning and video-language alignment on the Ego4D human video dataset. The results are shown in Figure 6.

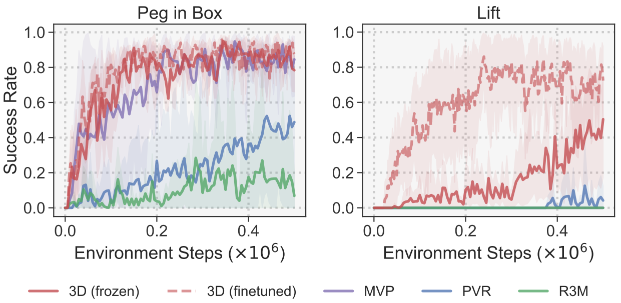


Figure 6: **Success rate of different frozen visual representations.** We compare our 3D visual representation with MVP (Xiao et al., 2022), PVR (Parisi et al., 2022), and R3M (Nair et al., 2022) on *peg in box* and *lift*.

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). The results are shown in Figure 6. 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 (Xiao et al., 2022; Nair et al., 2022; Parisi et al., 2022) only focus on the frozen visual representation, which might neglect the power of end-to-end policy learning (Levine et al., 2016).

Finetuning with the 3D objective. We then unfreeze the 3D visual representation and show the necessity of finetuning the visual representation with in-domain data and our 3D objective, *i.e.*, the reconstruction loss. As shown in Figure 8 *(left)*, we observe that our method without 3D finetuning initially keeps the similar converge rate but converges to a lower accuracy in 500k steps. The finetuning scale used for the 3D objective also matters as shown in Figure 8 *(mid)*, where a smaller scale could stable the learning process. In addition, we provide the quantitative and qualitative evaluation on the view synthesis results with 3D finetuning in Table 3 and Appendix E.6. We could observe that with 3D finetuning the view synthesis is more realistic and closer to the original images, and the quantitative evaluation in Table 3 also shows that 3D finetuning leads to better reconstruction.

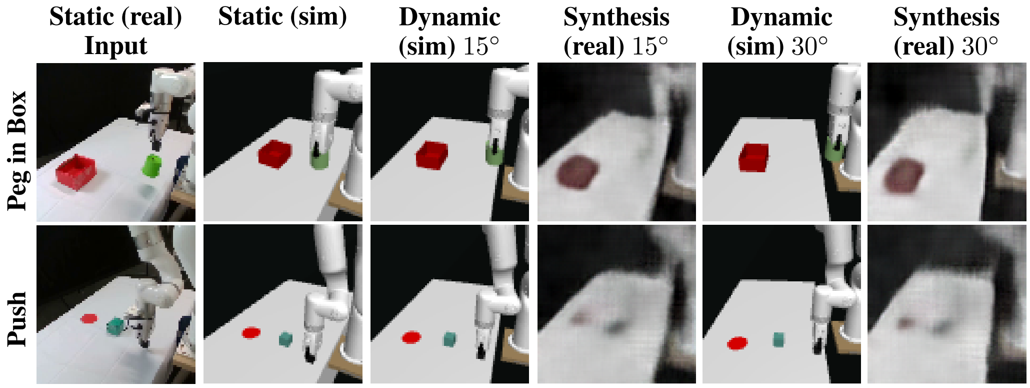


Figure 7: **Novel view synthesis in real.** We use images in the real world to generate the deep voxel and use the static view and the dynamic in the simulation to predict the transformation and then reconstruct the novel view. We display the reconstruction results for $\phi_d = 15^\circ, 30^\circ$ in two tasks.

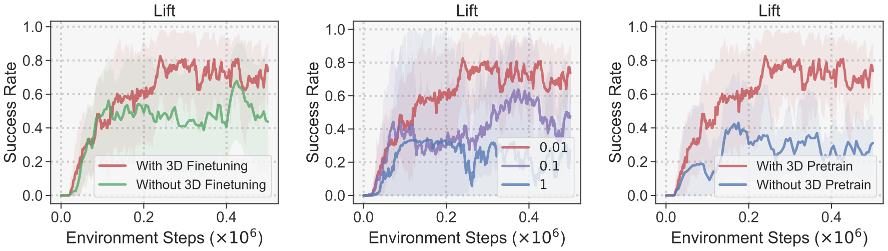


Figure 8: *(left)* Success rate of our 3D method with and without 3D finetuning on *lift*. *(mid)* Success rate of different lr_{ft} on *lift*, where lr_{ft} is the finetuning scale for 3D auxiliary task such that the learning rate would be $\text{lr} \times \text{lr}_{\text{ft}}$. *(right)* Success rate of our 3D method with and without 3D pretraining.

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 8 *(right)*, it is observed that without 3D pretraining the representation achieves much lower accuracy on *lift*. Compared to Figure 8 *(left)*, we could also find that the lack of 3D pretraining leads to more degradation of the success rates, showing that 3D pretraining is a necessary component.

Novel view synthesis in sim and real. We evaluate the 3D representation learned by our method by *(i)* qualitative novel view synthesis results from both simulated and real observations (shown in Figure 7), and *(ii)* quantitative ablations that provide insight into the effect of important hyperparameters on reconstruction quality (shown in Table 3). We find that *our method can synthesize meaningful reconstructions when provided with real camera observations* (and camera transformations predicted from simulated observations by $\mathcal{F}_{\text{pose}}$), despite not having seen our real robot setup before. For quantitative evaluation, we consider two standard metrics that measure the quality of synthesized images: Structural Similarity (SSIM) and Peak Signal-to-Noise Ratio (PSNR), and evaluate the impact of different finetuning rates λ_{ft} and camera angles ϕ_d, for $\phi = 30^\circ$. We observe that *(1)* using a reduced learning rate does not hurt reconstruction, and *(2)* our method trained with $\phi = 30^\circ$ generalizes well to out-of-distribution angles, up to 60° (2× that of training).

6 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 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. One limitation of our work is that it is not easy to train the agent in real directly during the era that training in real is feasible, which might be an interesting direction to explore in future.

Reproducibility

The main details of our 3D method have been stated in Section 4 and more details are remained in Appendix A. More information about baseline methods is given in Appendix B. We also give the used hyperparameters in Table 5 for reproducibility. We are committed to **releasing our code, environment, and the pretrained model**.

References

OpenAI Marcin Andrychowicz, Bowen Baker, Maciek Chociej, R. Józefowicz, Bob McGrew, Jakub W. Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, S. Sidor, Joshua Tobin, P. Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39:20 – 3, 2020. 1

Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *ArXiv*, abs/1612.00222, 2016. 2

Christopher P. Burgess, Loïc Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matthew M. Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *ArXiv*, abs/1901.11390, 2019. 2, 3

Michael Chang, Tomer David Ullman, Antonio Torralba, and Joshua B. Tenenbaum. A compositional object-based approach to learning physical dynamics. *ArXiv*, abs/1612.00341, 2017. 3

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan D. Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8973–8979, 2019. 3

Boyuan Chen, P. Abbeel, and Deepak Pathak. Unsupervised learning of visual 3d keypoints for control. *ArXiv*, abs/2106.07643, 2021. 3

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 15

Ricson Cheng, Arpit Agarwal, and Katerina Fragkiadaki. Reinforcement learning of active vision for manipulating objects under occlusions. In *Conference on Robot Learning*, pp. 422–431. PMLR, 2018a. 1

Xinjing Cheng, Peng Wang, and Ruigang Yang. Depth estimation via affinity learned with convolutional spatial propagation network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 103–119, 2018b. 17

Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *ArXiv*, abs/1910.11215, 2019. 2

Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl, and Dieter Fox. Self-supervised 6d object pose estimation for robot manipulation. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3665–3671, 2020. 2

Allan Dobbins, Richard Jeo, Jozsef Fiser, and John Allman. Distance modulation of neural activity in the visual cortex. *Science (New York, N.Y.)*, 281:552–5, 08 1998. doi: 10.1126/science.281.5376.552. 1

Yuqing Du, Olivia Watkins, Trevor Darrell, P. Abbeel, and Deepak Pathak. Auto-tuned sim-to-real transfer. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1290–1296, 2021. 3

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex X. Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *ArXiv*, abs/1812.00568, 2018. 2

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 512–519, 2016. 2

David R Ha and Jürgen Schmidhuber. World models. *ArXiv*, abs/1803.10122, 2018. 2

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018. 3, 4, 6

Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *ArXiv*, abs/1912.01603, 2020. 2

Josiah P. Hanna, Siddharth Desai, Haresh Karnan, Garrett Warnell, and Peter Stone. Grounded action transformation for sim-to-real reinforcement learning. *Special Issue on Reinforcement Learning for Real Life, Machine Learning, 2021*, May 2021. 3

Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *International Conference on Robotics and Automation*, 2021. 1, 2, 3

Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A. Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. In *International Conference on Learning Representations*, 2021a. 2, 3

Nicklas Hansen, H. Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. *ArXiv*, abs/2107.00644, 2021b. 2

Irina Higgins, Arka Pal, Andrei A. Rusu, Loïc Matthey, Christopher P. Burgess, Alexander Pritzel, Matthew M. Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. *ArXiv*, abs/1707.08475, 2017. 1, 2

Jeffrey Ichnowski, Yahav Avigal, Justin Kerr, and Ken Goldberg. Dex-neRF: Using a neural radiance field to grasp transparent objects. In *5th Annual Conference on Robot Learning*, 2021. 3

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks, 2016. 2

Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12619–12629, 2019. 3

Rishabh Jangir, Nicklas Hansen, Sambaran Ghosal, Mohit Jain, and Xiaolong Wang. Look closer: Bridging egocentric and third-person views with transformers for robotic manipulation. *IEEE Robotics and Automation Letters*, pp. 1–1, 2022. doi: 10.1109/LRA.2022.3144512. 3, 15

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *ArXiv*, abs/1906.08253, 2019. 2

Maximilian Jaritz, Jiayuan Gu, and Hao Su. Multi-view PointNet for 3D Scene Understanding. *arXiv preprint*, 2019. 3

R. Julian, B. Swanson, G. Sukhatme, Sergey Levine, Chelsea Finn, and Karol Hausman. Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning. *arXiv: Learning*, 2020. 3

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998. 3

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *International Conference on Learning Representations*, 2020. 2, 6, 15

Tejas D. Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *ArXiv*, abs/1906.11883, 2019. 2

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *ArXiv*, abs/2107.04034, 2021. 3

Zihang Lai, Sifei Liu, Alexei A. Efros, and Xiaolong Wang. Video autoencoder: self-supervised disentanglement of static 3d structure and motion. *ArXiv*, abs/2110.02951, 2021. 2, 3, 4, 5

Sascha Lange and Martin A. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2010. 2

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020. 2

Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. A simple randomization technique for generalization in deep reinforcement learning. *ArXiv*, abs/1910.05396, 2019. 3

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016. 8

Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. *ArXiv*, abs/2107.04004, 2021. 3

T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2016. 1

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *ArXiv*, abs/2006.15055, 2020. 3

Matthias Minderer, Chen Sun, Ruben Villegas, Forrester Cole, Kevin P. Murphy, and Honglak Lee. Unsupervised learning of object structure and dynamics from videos. *ArXiv*, abs/1906.07889, 2019. 2

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. 1

Ashvin Nair, Vitchyr H. Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *NeurIPS*, 2018. 1, 2

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022. 3, 8, 16

Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Kumar Gupta. The unsurprising effectiveness of pre-trained vision models for control. *ArXiv*, abs/2203.03580, 2022. 1, 3, 7, 8, 16

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018. 1, 2

Lerrel Pinto, Marcin Andrychowicz, P. Welinder, Wojciech Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. *ArXiv*, abs/1710.06542, 2018. 3

Haozhi Qi, Xiaolong Wang, Deepak Pathak, Yi Ma, and Jitendra Malik. Learning long-term visual dynamics with region proposal interaction networks. In *ICLR*, 2021. 3

Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real world robot learning with masked visual pre-training. In *6th Annual Conference on Robot Learning*. 3

Fabio Ramos, Rafael Possas, and Dieter Fox. Bayessim: Adaptive domain randomization via probabilistic inference for robotics simulators. *Robotics: Science and Systems XV*, Jun 2019. 3

Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *International Conference on Computer Vision*, 2021. 2, 5

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y. 6

Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron C. Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *ICLR*, 2021. 2

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017. 1

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020. 1, 2

Adam Stooke, Kimin Lee, P. Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. *ArXiv*, abs/2009.08319, 2021. 2

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. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep 2017. 2, 3

Kalifou René Traoré, Hugo Caselles-Dupré, Timothée Lesort, Te Sun, Natalia Díaz Rodríguez, and David Filliat. Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer. *ArXiv*, abs/1906.04452, 2019. 3

Ya-Yen Tsai, Hui Xu, Zihan Ding, Chong Zhang, Edward Johns, and Bidan Huang. Droid: Minimizing the reality gap using single-shot human demonstration. *IEEE Robotics and Automation Letters*, 6:3168–3175, 2021. 3

Hsiao-Yu Fish Tung, Ricson Cheng, and Katerina Fragkiadaki. Learning spatial common sense with geometry-aware recurrent networks. *CoRR*, 2019a. 3

Hsiao-Yu Fish Tung, Ricson Cheng, and Katerina Fragkiadaki. Learning spatial common sense with geometry-aware recurrent networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2595–2603, 2019b. 1

Hsiao-Yu Fish Tung, Xian Zhou, Mihir Prabhudesai, Shamit Lal, and Katerina Fragkiadaki. 3d-oes: Viewpoint-invariant object-factorized environment simulators. In *CoRL*, 2020. 3

Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380, 1991. 17

Rishi Veerapaneni, John D. Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua B. Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning. *ArXiv*, abs/1910.12827, 2019. 2

Oriol Vinyals, I. Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, J. Chung, David H. Choi, Richard Powell, Timo Ewalds, P. Georgiev, Junhyuk Oh, Dan Horgan, M. Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, J. Agapiou, Max Jaderberg, A. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, D. Budden, Yury Sulsky, James Molloy, T. Paine, Caglar Gulcehre, Ziyun Wang, T. Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pp. 1–5, 2019. 1

Chen Wang, Roberto Martín-Martín, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke Zhu. 6-pack: Category-level 6d pose tracker with anchor-based keypoints. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10059–10066. IEEE, 2020a. 1

K. Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving generalization in reinforcement learning with mixture regularization. *ArXiv*, abs/2010.10814, 2020b. 3

Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv:2203.06173*, 2022. 1, 3, 8, 16

Mengyuan Yan, Iuri Frosio, Stephen Tyree, and Jan Kautz. Sim-to-real transfer of accurate grasping with eye-in-hand observations and continuous control. *ArXiv*, abs/1712.03303, 2017. 1, 3

Wilson Yan, Ashwin Vangipuram, P. Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. *ArXiv*, abs/2003.05436, 2020. 2

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. 2019. 1, 2

Weirui Ye, Shaohuai Liu, Thanard Kurutach, P. Abbeel, and Yang Gao. Mastering atari games with limited data. *ArXiv*, abs/2111.00210, 2021. 2

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 (CoRL)*, 2019. URL https://arxiv.org/abs/1910.10897. 2

Xiaoshuai Zhang, Rui Chen, Fanbo Xiang, Yuzhe Qin, Jiayuan Gu, Zhan Ling, Minghua Liu, Peiyu Zeng, Songfang Han, Zhiao Huang, Tongzhou Mu, Jing Xu, and Hao Su. Close the Visual Domain Gap by Physics-Grounded Active Stereovision Depth Sensor Simulation. *arXiv preprint*, 2022. 3

A Implementation of Our 3D Method

2D encoder. We use ResNet18 as the encoder.

3D encoder consists of two 3D transposed convolutions activated by Leaky ReLU function.

3D decoder first reshapes the deep voxel $(C/D) \times D \times H \times W$ back to $C \times H \times W$, and consists of one 2D 1×1 convolution and three transposed convolutions to reconstruct the RGB images.

PoseNet concatenates two images as the input and reduces the dimension to 6 for predicting Euler angles $[\alpha, \beta, \gamma]^\top$ and the translation $[x, y, z]^\top$, by seven 2D convolutions.

3D transformation. We warp the grid such that the voxel at location $p = [i, j, k]^\top$ will be warped to $\hat{p}$, which is computed as

$$\hat{p} = Rp + t \tag{4}$$

where R, t is the 3×3 rotation matrix and translation vector corresponding to the camera pose. In our implementation, the warp is performed inversely and the value at fractional grid location is trilinearly sampled. In addition, since there exists misaligned voxels during the sampling procedure caused by the coarse deep voxel representation, we apply two 3D convolutions to refine and correct these mismatches.

3D pretraining. We randomly select 20 classes in CO3D and sample I_{src} and I_{tgt} with a bounded interval $b = 9$. We train $\sim 250k$ iterations with batch size 32.

3D finetuning. In the finetuning phase, we apply a less frequent update when doing the 3D task, *i.e.*, performing λ_{up} 3D update every 1 RL update. In practice, we set $\lambda_{\text{ft}} = 10^{-2}$, $\text{lr}_{\text{RL}} = 10^{-3}$, and $\lambda_{\text{up}} = 0.5$. In addition, we only apply the reconstruction loss.

B Baselines

From scratch, also called vanilla SAC, does not use any pretrained model and utilizes a 2D encoder with fourteen 2D convolutions activated by ReLU function. Our implementation generally follows DrQ (Kostrikov et al., 2020). The actor consists of fully connected layers activated by Tanh function. The critic applys fully connected layers activated by ReLU function and predicts double action value funtions Q with a shared encoder and two different heads. We apply the same data augmentation as in (Jangir et al., 2022) for better sim-to-real transferring, including *random shift* and *color jitter*.

ImageNet. We replace the 2D encoder in vanilla SAC with ResNet18 pretrained with supervised learning on ImageNet, to gain a stronger 2D representation.

MoCo. We replace the 2D encoder in vanilla SAC with ResNet18 which is trained by MoCo (v2) (Chen et al., 2020) on ImageNet under the setting where the batch size is 256, the number of epochs is 100, and the initial learning rate is 0.03.

Remove ImageNet normalization for usage. Our baseline methods MoCo and ImageNet are both pretrained with ImageNet and all input images are preprocessed with the normalization of ImageNet, *i.e*, with the mean $(0.485, 0.456, 0406)$ and the standard deviation $(0.229, 0.224, 0.225)$. A natural way to apply such pretrained networks in RL is using the same normalization to maintain the representation ability of the pretrained networks. However, by empirical experiments we find that normalizing the images directly into $[0, 1]^d$ achieves a much better performance, as shown in Figure 9a. Thus we adopt a stronger version as our baseline.

C Design of Cameras

Design of the static view. The static view is generally used for all baselines and our algorithm, for both the training phase and the inference phase. Thus the inner requirement is that the static view should contain the majority of useful information for the robotic task, to gain a strong baseline. In practise we carefully select a unified static view for all tasks.

Design of the dynamic view. The dynamic camera that shots I_{tgt} is essential, which largely decides whether the image reconstruction could work. Based on the priori that our task is object-centric and the intuition that interaction between the robot and the object is our focus, we move the dynamic view in a object-centric manner, *i.e.*, moving along a circle around the center of the scene, starting from the static view as a initial position. The center of the scene is designed to cover the necessary objects and the main scene information. For example, in the *peg in box* task, where the robot is required to move the peg into the box on the table, the box is essential to understand and solve this task, and thus the box could be seen as the center of the scene.

Formally, let $[x_s, y_s, z_s]^\top$ denote the position of the static view, $[x_d, y_d, z_d]^\top$ denote the position of the dynamic view, $[x_c, y_c, z_c]^\top$ denote the center of the scene, r denote the radius of the circle, ϕ_s denote the rotation angle of the static view, and ϕ_d denote the rotation angle of the dynamic view from the static view. We also introduce ϕ, which denotes the range of the rotation of the dynamic view. Then the translation of the cameras is given in Equation 5 and 6. The rotation angle is automatically computed by making the camera point to the center of the scene with the z-axis in the plane perpendicular to the ground.

$$[x_s, y_s, z_s]^\top = [x_c, y_c, z_c]^\top + r \cdot [\sin\phi_s, \cos\phi_s, 0]^\top, \text{ where } \phi_s \text{ is predefined.} \tag{5}$$

$$[x_d, y_d, z_d]^\top = [x_c, y_c, z_c]^\top + r \cdot [\sin(\phi_s + \phi_d), \cos(\phi_s + \phi_d), 0]^\top, \text{ where } \phi_d \in [0, \phi]\,. \tag{6}$$

D Novel View Synthesis in Real

Videos consisting of synthesised views are displayed in our project website https://3d4rl.github.io/. In this section we describe details of how we generate the synthesised views for the real world.

Let I_{real} denote the image shot in the real world from the same static view as in simulation. The deep voxel representation is thus generated as $g_\theta(f_\theta(I_{\text{real}}))$. Since we only have one static camera in the real world (we could have other cameras in real, but it is not necessary for our method), we use the simulation images $I_{\text{src}}, I_{\text{tgt}}$ (the same notation as before) to predict the relative transformation by PoseNet and get transformation R, t. Then the reconstructed image is $\hat{I}_{\text{recon}} = h_\theta(T_{R,t}(g_\theta(f_\theta(I_{\text{real}}))))$, which should be in the same view as I_{tgt}, but different in the scene content.

To generate videos, we do interpolation on the generated transformations and apply these new transformations to get interpolated views,. The output of PoseNet is $[\alpha, \beta, \gamma, x, y, z]^\top$, and let $\lambda \in [0, 1]$ denote the interpolation factor. Then the interpolated transformations are

$$(1 - \lambda)[0, 0, 0, 0, 0, 0]^\top + \lambda[\alpha, \beta, \gamma, x, y, z]^\top = [\lambda\alpha, \lambda\beta, \lambda\gamma, \lambda x, \lambda y, \lambda z]^\top\,, \tag{7}$$

from which we could trivially gain R, t.

E Additional Results

E.1 Dynamic Camera Moving Range ϕ

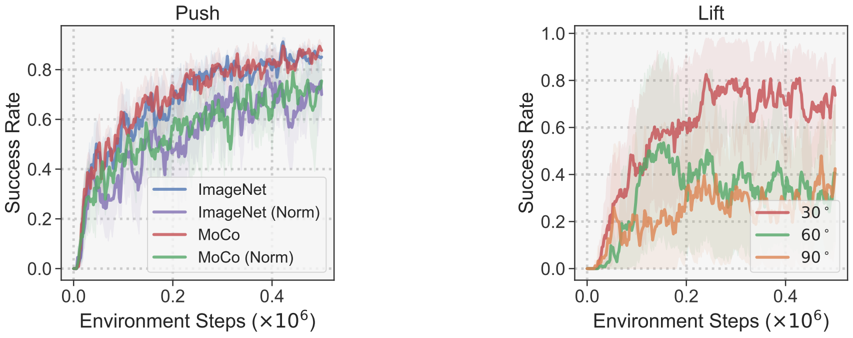
The moving range of the dynamic camera affects the performance of our 3D algorithm. Specifically, a larger ϕ may impede the learning process and hurt the performance. We use *lift* task as an example to illustrate the effect of ϕ, as shown in Figure 9b. We find that with a relatively small range, *i.e.*, 30°, 3D could be more stable.

E.2 ImageNet Normalization in Baselines

When the visual representation is frozen, the ImageNet normalization is generally used across all pretrain methods (Xiao et al., 2022; Nair et al., 2022; Parisi et al., 2022). However, we find that when we could train the policy and the visual representation end-to-end, it would be better to not apply the ImageNet normalization, as shown in Figure 9a. We thus adopt the stronger baseline.

E.3 Pose Estimation

Our 3D method trains a PoseNet that could estimate the relative pose between two frames and we evaluate our pose estimation results quantitatively in this section. For a whole trajectory generated



(a)

(b)

Figure 9: *(a)* Compare baselines with and without ImageNet Normalization. *(b)* Success rate of our method with different camera moving range ϕ on *lift*.

Table 4: **Quality of pose estimation for peg in box task.** Each table shows the root mean square error (RMSE) and the maximal error (MaxE) of different dynamic camera angle ϕ_d given certain finetuning scale λ_{ft}. We only train with $\phi_d = 30°$. We could observe that the finetuning leads to consistent smaller errors.

Pretrain			$\lambda_{\text{ft}} = 0.01$			$\lambda_{\text{ft}} = 0.10$			$\lambda_{\text{ft}} = 1.00$		
ϕ_d	RMSE↓	MaxE↓	ϕ_d	RMSE↓	MaxE↓	ϕ_d	RMSE↓	MaxE↓	ϕ_d	RMSE↓	MaxE↓
15	0.041	0.093	15	0.041	0.089	15	0.040	0.091	15	0.046	0.126
30	0.066	0.142	30	0.059	0.122	30	0.033	0.097	30	0.041	0.124
45	0.120	0.246	45	0.064	0.150	45	0.046	0.102	45	0.044	0.120
60	0.174	0.334	60	0.130	0.393	60	0.132	0.382	60	0.133	0.365
avg	0.100	0.204	avg	0.073	0.189	avg	0.063	**0.168**	avg	**0.066**	0.184

by the interaction between our agent and the environment, we estimate the relative pose between the dynamic camera and the static camera for each timestep. Since the estimated transformation is in the coordinate space of deep voxels, Umeyama alignment (Umeyama, 1991) is applied to align the predicted trajectory with the ground truth trajectory provided by our simulation environment. We set diverse dynamic camera angles to test both in-domain $(15°, 30°)$ and out-of-domain $(45°, 60°)$ pose estimation under various finetuning scales. Results in Table 4 show that our method reduces the pose estimation error compared to the network that is only pretrained with CO3D dataset. Our method could also generalize to 45 degrees with a small error equal to 0.064, nearly half of the one with only pretraining. In addition, we find that larger finetuing scales generally reduce the error, and even finetuning with a very small scale could result in a gap compared to the pretrain model.

E.4 Compare with other 3D Pretrain Methods

In our main sections we demonstrate the advantage of our method over 2D representations, and we are now showing that our 3D self-supervised representation is also better than other straightforward pretrain methods that contain 3D information. Specifically, we consider the ResNet50 model pretrained by the depth estimation task using convolutional spatial propagation network (CSPN) (Cheng et al., 2018b). We still freeze the visual representation across methods. The results are

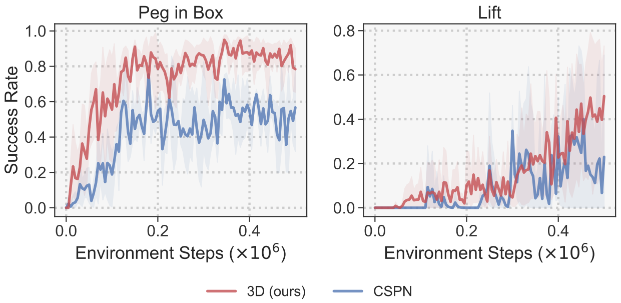


Figure 10: **Success rate of different frozen visual representations.** We compare our 3D visual representation with CSPN (Cheng et al., 2018b) on *peg in box* and *lift*.

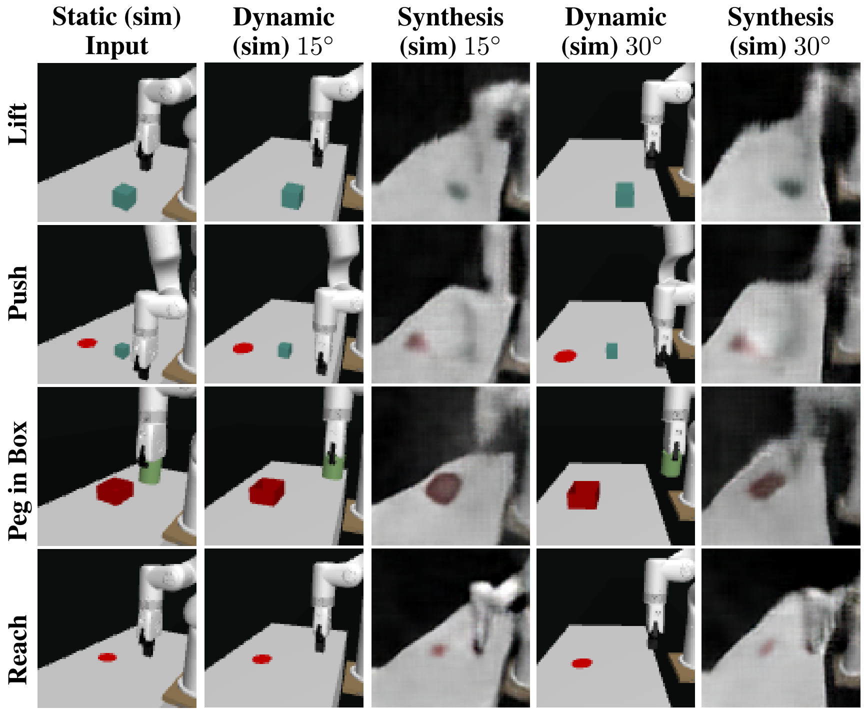


Figure 11: **Novel view synthesis in simulation.** We display the reconstruction results for $\phi_d = 15°, 30°$ in four tasks.

shown in Figure 10 under the same setting as Figure 6. We find that our 3D representation is consistently better on *peg in box* and *lift*, while the CSPN model could also gain reasonable accuracy.

E.5 Computational Overhead

Although our 3D based algorithm is elegantly designed for better sample efficiency, the computational overhead of utilizing the auxiliary task for joint optimization is non-negligible. We measure the computation time for one 3D update (0.038s) and one RL update (0.063s) averaged over 10 iterations on a NVIDIA GeForce RTX 3090. The large overhead is mainly because our method reconstructs the image from the 3D scene latent, which is higher dimensional ($\mathcal{O}(n^3)$) than common 2D methods ($\mathcal{O}(n^2)$). How to make the utilization of 3D information more computational efficient is interesting to explore in our future work.

E.6 Novel View Synthesis Results in Sim and Real

We provide qualitative results of our novel view synthesis both in simulation and in the real world. In Figure 7 we show the synthesis using real world images and our model is only trained in simulation. Figure 11 gives more results in simulation. We also compare the synthesis generated by the pretrained model and the finetuned model in Figure 12, where we find that the pretrained model could grasp the main objects in the scene while the domain gap, *e.g.*, color, could be clearly observed.

F Visualization of our Environments

We give more visualization of our environments, including four xArm environments: *Lift*, *Push*, *Peg in Box*, and *Reach*, and four Meta-World environments: *Basketball*, *Box Close*, and *Coffee Push*, and *Hammer*, shown in Figure 13. For one single xArm task, we are giving different initialization setting, showing the randomization in our environments for generalization. For one single Meta-World task, we show different views along the trajectory of the dynamic camera. We also visualize the perturbed

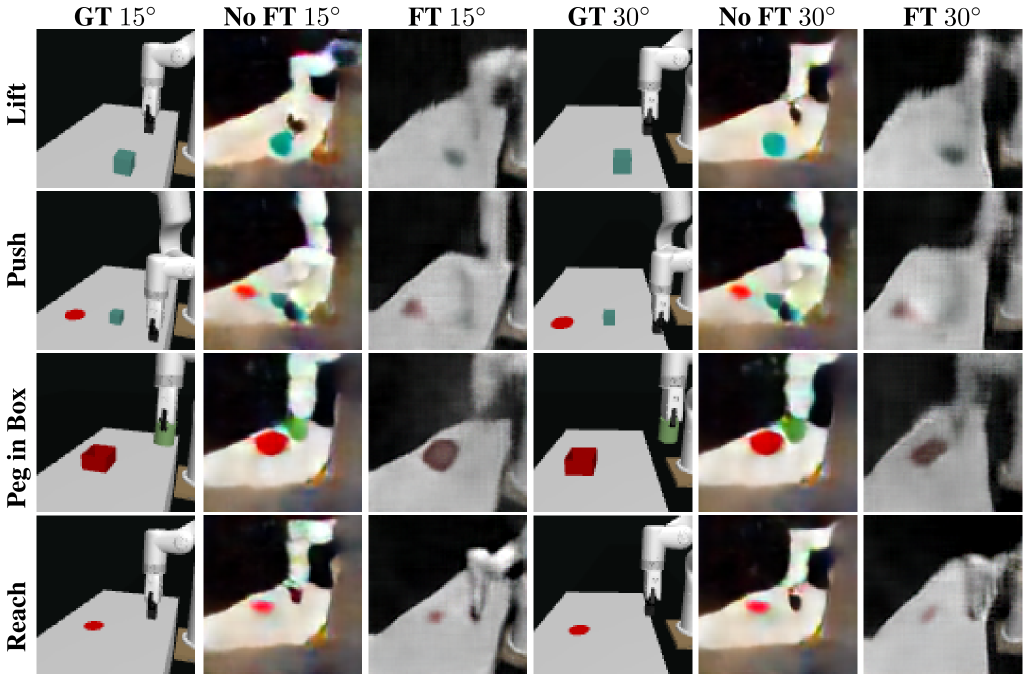


Figure 12: **The effect of 3D finetuning for novel view synthesis in simulation.** *GT* represents for ground truth and *FT* represents for finetuning. We display the reconstruction results for $\phi_d = 15°, 30°$ in four tasks.

simulation environments as shown in Figure 14 and the examples of successful trajectories as shown in Figure 15.

G Hyper-parameters

We provide all relevant hyper-parameters used in our experiments in Table 5, including both parameters that are discussed and not discussed in our paper.

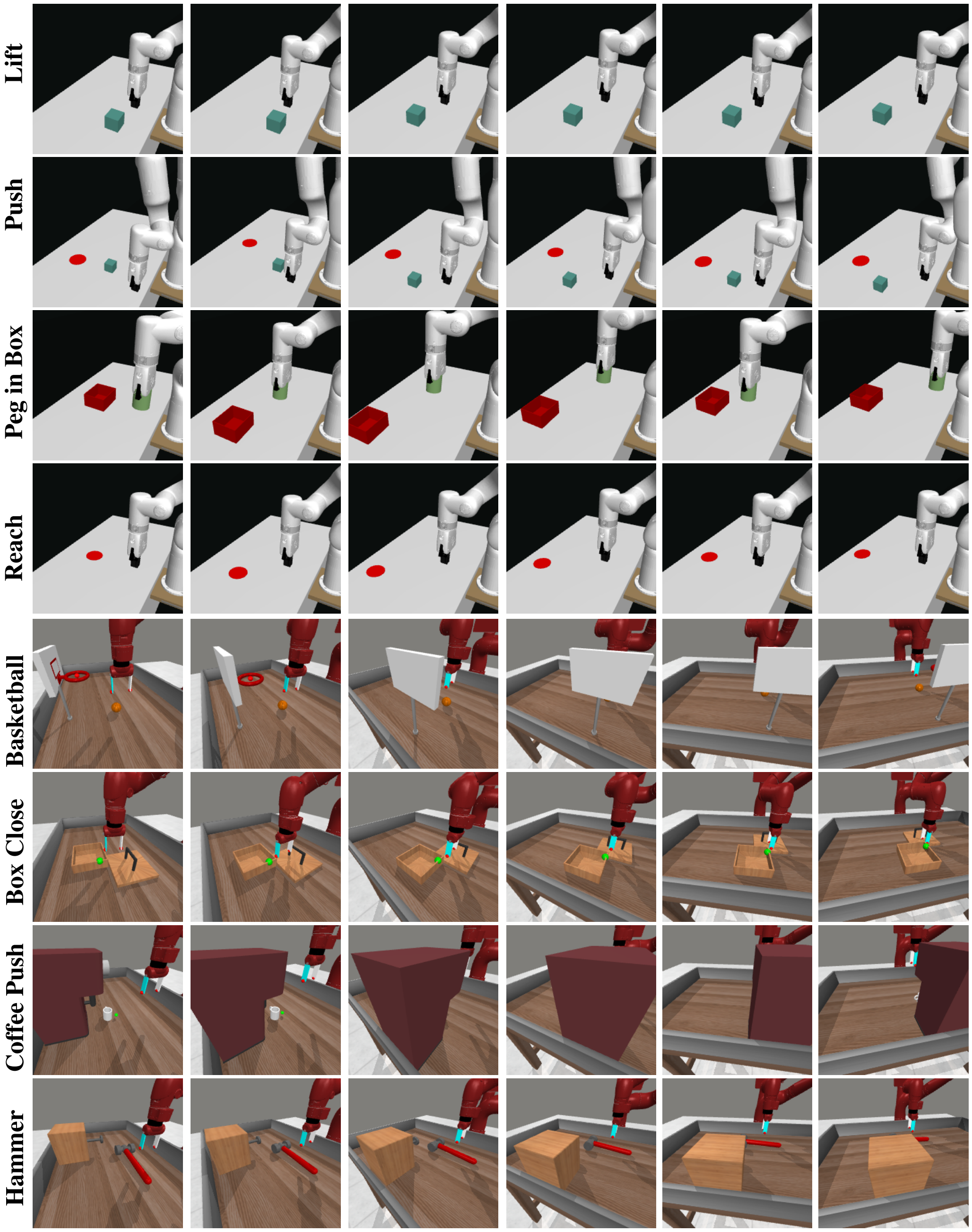


Figure 13: **Visualizations.** We visualize four xArm environments and selected four Meta-World environments. Our xArm environments are shown across different initialization, where initial position of end-effector and objects are randomized. Meta-World environments are shown along the trajectory of the dynamic camera.

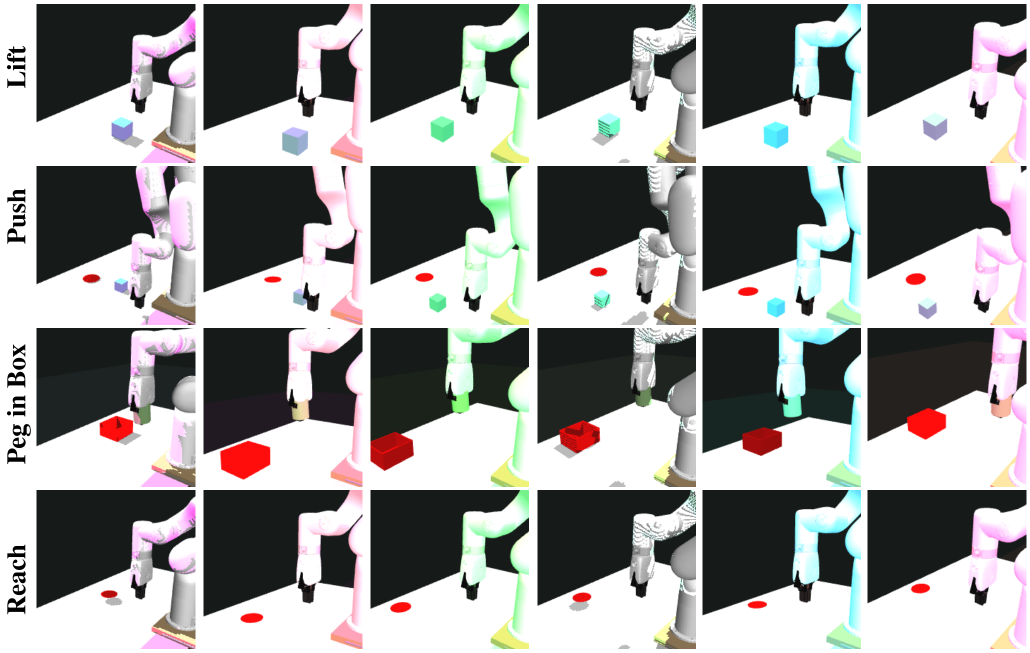


Figure 14: **Visualizations of perturbed simulated environments.** We visualize four xArm environments. Our xArm environments are shown across different initialization, together with texture randomization, lighting randomization, and camera perturbation.

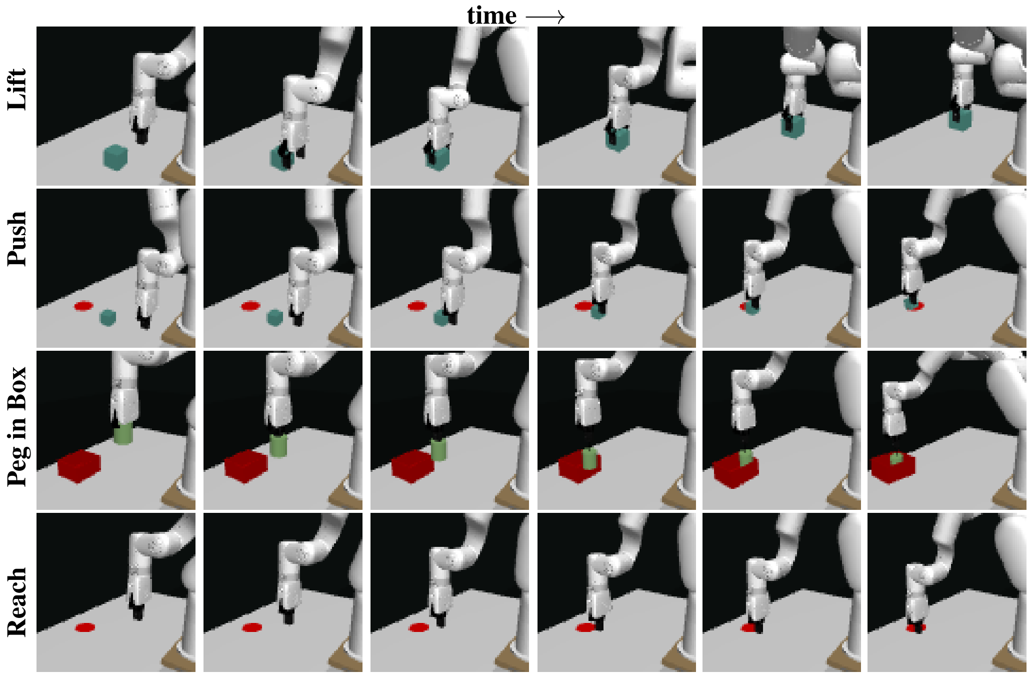


Figure 15: **Visualizations of trajectories in simulated environments.** We sample one successful trajectory for each xArm environment.

Table 5: **Hyper-parameters.**

Variable	Description	Value
b	the bounded interval of sampling training frames	9
lr_{RL}	learning rate of the RL agent	10^{-3}
λ_{up}	frequency of 3D update	0.5
λ_{ft}	finetuning scale	0.01
ϕ	dynamic camera angle range	$30°$
–	observation shape	$84 \times 84 \times 3$
–	episode length of xArm tasks	50
–	episode length of MetaWorld tasks	200
–	replay buffer capacity	$500k$
–	batch size of replay buffer sampling	128
–	training steps (xArm)	$500k$
–	training steps (Meta-World)	1m
–	discount factor	0.99
–	initial random steps	1000
–	initial temperature	0.1
–	frequency of RL update	1
–	random shift padding	4
–	brightness of color jitter	0.4
–	saturation of color jitter	0.4
–	contrast of color jitter	0.4
–	hue of color jitter	0.5