Visual Reinforcement Learning with Self-Supervised 3D Representations

Anonymous Author(s)
Affiliation
Address
email

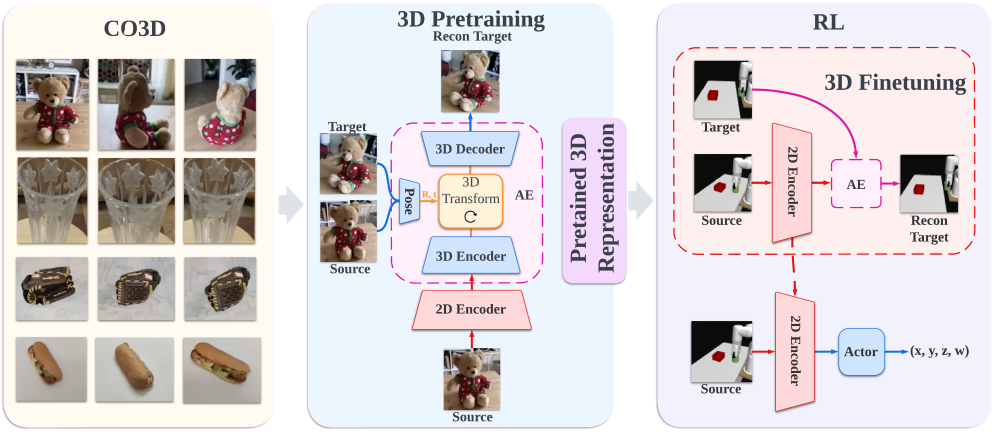

Figure 1: **Overview of our approach.** *(left)* We pretrain a 3D deep voxel-based auto-encoder on the *Common Objects in 3D* (CO3D) dataset, a large object-centric dataset. *(right)* We train an RL policy in simulation using the learned representation as initialization, and jointly finetune the representation with 3D and RL objectives on in-domain data collected by the RL agent.

Abstract: We present a unified framework for self-supervised learning of 3D representations for visual reinforecment learning. Our 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, better sim-to-real transfer, and robustness compared to 2D representation learning methods. 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, 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. 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 arguably need to perceive it to tackle the enormous complexity of real world environments.

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 1 provides an overview of our method. In the first phase, we learn a generalizable 3D representation using a repurposed *video autoencoder* [1] that performs 3D deep voxel-based novel view synthesis without assuming access to ground-truth cameras, pretrained on Common Objects

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

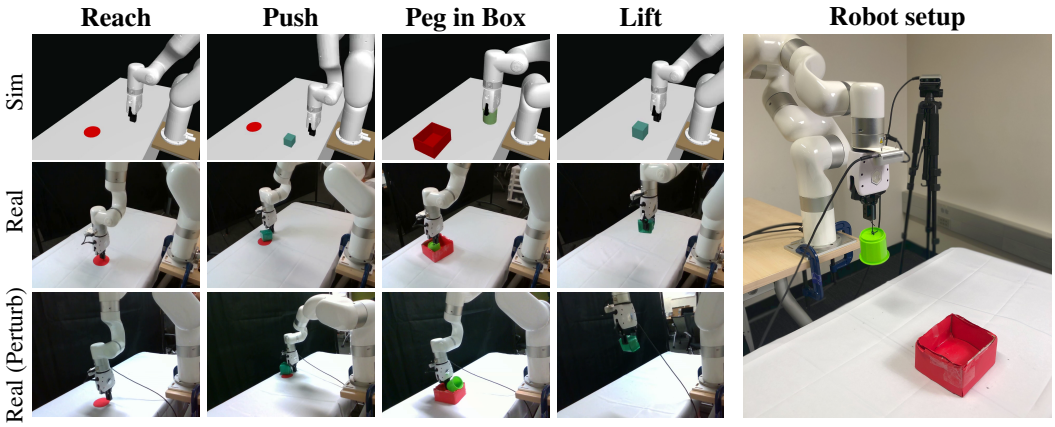

Figure 2: **Overview of sim-to-real tasks.** We consider four tasks for our sim-to-real experiments: *(1) reach*, where the agent needs to position the gripper at the red goal, *(2) push*, where the agent needs to push a green cube to the red goal, *(3) peg in box*, where the agent needs to place a green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

in 3D (CO3D) [2] – a large-scale object-centric 3D dataset. In the second phase, we finetune the learned representation together with policy learning on in-domain data collected by online interaction. 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 [3] tasks and four robotic manipulation tasks with camera feedback both in simulation and the real world as shown in Figure 2. 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 2). 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.

2 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. Figure 1 provides an overview of our approach.

2.1 Object-Centric 3D Pretraining

Our framework is implemented as a deep voxel-based 3D auto-encoder [1] 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. The training scheme is similar to [1] except we adopt a ResNet18 as the 2D encoder and only remain the image reconstruction loss.

2.2 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 π_θ that takes feature maps from the pretrained 2D encoder as input and outputs a continuous action. During this phase, we optimize the 2D encoder 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

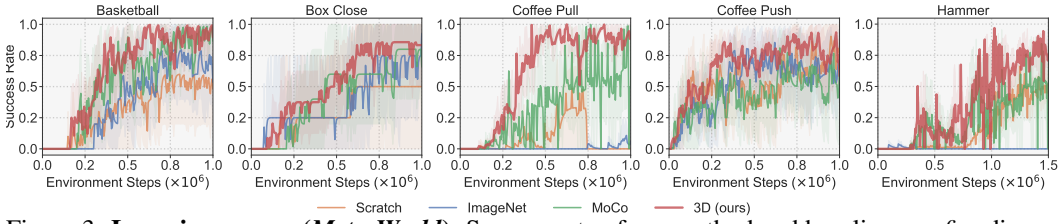

Figure 3: **Learning curves (*Meta-World*)**. Success rate of our method and baselines on *five* diverse image-based Meta-World tasks. Mean of 5 seeds, shaded areas are 95% CIs. Our method achieves non-trivial success rates faster than other methods. See supplementary material for samples.

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.

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 $\phi_d \in [0, \phi]$. 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 $\text{lr}_{\text{3D}} = \lambda_{\text{ft}} \times \text{lr}_{\text{RL}}$.

3 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. 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. *We are committed to releasing our code.*

Robot setup. Our real robot setup is shown in Figure 2 *(right)*. We use an xArm robot equipped with a gripper in our real-world experiments, and observations are captured by a static third-person RGB camera with dimension 84×84.

Baselines. We implement our method and all baselines using Soft Actor-Critic (SAC; [4]) as the backbone RL algorithm and use the same hyperparameters whenever applicable. 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 [5]; and *(iii)* a ResNet18 pretrained on ImageNet using the self-supervised **MoCo** objective. All methods use ± 4 random shift [6] 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. 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. 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) 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 2 *(left)* for the two setups.

3.1 Sample-Efficiency

We train for 1m environment steps across Meta-World tasks. Results for Meta-World tasks are shown in Figure 3. We find that 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*, while it fails to solve harder tasks like *coffee pull*. MoCo generally leads to better downstream performance than pretraining with ImageNet classification, which is consistent with observations made in prior work [7], while the performance gap is relatively small for most tasks. Our proposed

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

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

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* and *hammer* that require spatial understanding.

3.2 Sim-to-Real Transfer

We evaluate policies trained in xArm simulation environments. 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.

3.3 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 real world includes the camera position, the camera orientation, the lighting, and the background, as visualized in Figure 2. 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 and better in generalization.

4 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. 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.

References

[1] Z. Lai, S. Liu, A. A. Efros, and X. Wang. Video autoencoder: self-supervised disentanglement of static 3d structure and motion. *ArXiv*, abs/2110.02951, 2021.

[2] 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 *International Conference on Computer Vision*, 2021.

[3] 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 *Conference on Robot Learning (CoRL)*, 2019. URL https://arxiv.org/abs/1910.10897.

[4] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. 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] I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *International Conference on Learning Representations*, 2020.

[7] S. Parisi, A. Rajeswaran, S. Purushwalkam, and A. K. Gupta. The unsurprising effectiveness of pre-trained vision models for control. *ArXiv*, abs/2203.03580, 2022.