Supplementary Material: Unsupervised Learning of Visual 3D Keypoints for Control

Boyuan Chen [1] **Pieter Abbeel** [1] **Deepak Pathak** [2]

1. Additional Implementation Details

Keypoint Resampling We find that the decoder can potentially cheat by hiding the input information in the relative locations keypoint to each other. To handle this issue, we do not use the exact locations produced as output by the keypoint encoder but re-sample them from the empirical distribution. We find that during training, injecting noise by re-sampling around $[\mathbb{E}[u_n^k], \mathbb{E}[v_n^k], \mathbb{E}[d_n^k]]^\top$ (corresponding to k-th keypoint in the n-th camera frame) produces sharper keypoint heatmaps that avoids potential multimodality issues. Such noise also encourages keypoints to correspond to actual visual features rather be simply used as bits to pass information to decoder. We can calculate σ_n^k, the spatial standard deviation of $[u_n^k, v_n^k]^\top$, from the heatmap.

$$\sigma_n^k = \sqrt{\sum_{u=1}^{S}\sum_{v=1}^{S} || \begin{bmatrix} u/S \\ v/S \end{bmatrix} - \begin{bmatrix} \mathbb{E}[u_n^k] \\ \mathbb{E}[v_n^k] \end{bmatrix} ||_2^2 \cdot H_n^k(u,v)}$$

At training time, we resample $[\hat{u}_n^k, \hat{v}_n^k, \hat{d}_n^k]^\top = [\mathbb{E}[u_n^k] + a\sigma_n^k, \mathbb{E}[v_n^k] + b\sigma_n^k, \mathbb{E}[d_n^k]]^\top$ where $a, b \sim \mathcal{N}(0, \nu)$ for some positive noise hyper parameter ν. This resampled version of predicted keypoint coordinate encourages heatmap to have small spatial variance to minimize noise while also making decoder more robust to noisy keypoint predictions. At inference time, we set $\nu = 0$ to disable noise and compute $[\hat{u}_n^k, \hat{v}_n^k, \hat{d}_n^k]^\top$=$[\mathbb{E}[u_n^k], \mathbb{E}[v_n^k], \mathbb{E}[d_n^k]]^\top$.

First Frame Features When reconstructing observed images from distillated keypoints, the decoder needs to recover the relatively static background pixels. This can be achieved by decoder itself. However, doing so wastes the compute and parameters of the decoder. We thus use the same technique in the 2D Keypoint baseline (Minderer et al., 2019) in experiment section, to alleviate this problem by concatenating first frame features to the Gaussian maps before decoding. Such features are extracted by passing a canonical frame, such as the first frame of environment or the photo of an empty scene, to a small convolutional encoder. This small encoder effectively captures the static pixels for reconstruction so the decoder can focus on keypoints better.

Hyperparameter	Metaworld	Ant
PPO batch size (metaworld)	6400	16000
Rollout buffer size	100000	100000
# Epochs per update	8	10
gamma	0.99	0.99
gae lambda	0.95	0.95
clip range (ϵ)	0.2	0.2
entropy coefficient	0.0	0.0
value function coefficient	0.5	0.5
gradient clip	0.5	0.5
target KL	0.12	0.12
policy learning rate	$3e-4$	$3e-4$
observation buffer size	100000	100000
unsupervised learning rate	$3e-4$	$3e-4$
# keypoints	32	16
# cameras (N)	3	3
# unsupvised learning steps (p)	400	400
noise ν when no augmentation	$5e-4$	$5e-4$
noise ν when augmented	0.0	0.0
Autoencoding weight (λ_{ae})	5.0	5.0
Multiview weight (λ_{multi})	0.05	0.05
Seperation weight (λ_{sep})	0.0025	0.0025

Table 1. The values of all hyperparameters for our algorithm. The code is available at https://buoyancy99.github.io/unsup-3d-keypoints/.

Additional states In partially observable environments, we also concatenate some unobservable robot states with learned keypoints before feeding into the fully connected policy layers. These states are those easily accessible on real robot but can hardly be assessed from our third person view camera setup. For metaworld, we use an indicator of whether the gripper is opening or closing because the gripper is hardly visible when arm is far from the third-person-view cameras. For pybullet-ant, such state is the x coordinate of the robot center, which cannot be estimated visually since the camera is always moving with robot and the ground is

[1]UC Berkelely [2]Carnegie Mellon University. Correspondence to: Deepak Pathak <dpathak@cs.cmu.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

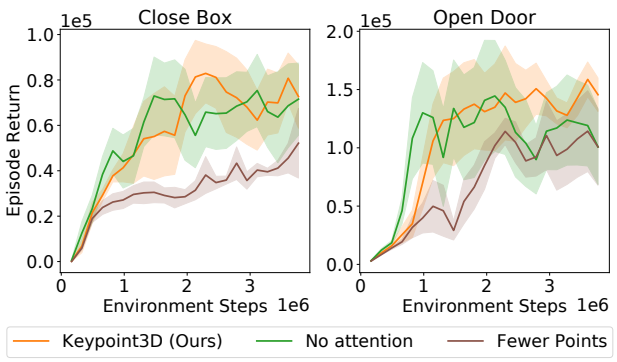


Figure 1. Additional ablations of our Keypoint3D approach with respect to different design choices. Variations include: removing attention, using fewer keypoints.

white. In scarf environment, such state is pr2 robot's arm joint angles since they can hardly be estimated under low resolution even by humans. They are also critical states for joint space control with obstacle avoidance.

Implementation details We provide all hyper parameters for PPO training as well as that for unsupervised learning in Table 1. More details can be found in the code we provided.

2. Additional Ablations

In addition to the ablations provided in the main paper, we present few more ablative studies in this section for better understanding of our method.

Effects of attention Figure 1 shows removing the attention mechanism has some minor impact on the reinforcement learning performance. To understand this, we notice the role of attention is to ignore irrelevant keypoints. To achieve this, we can either use attention or let the decoder learn to figure out by itself. Thus the decoder can still achieve the similar effect when attention is removed. However, we still decide to keep the attention module since we found attention mechanism to be very important for visualization of keypoints, in which we use a threshold on attention to filter out unimportant keypoints.

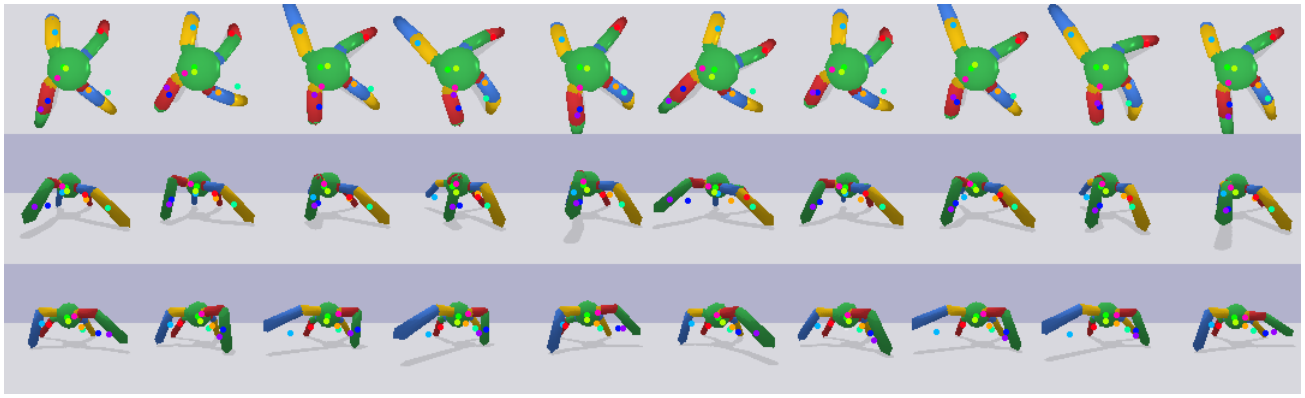
Effects of Effects of number of keypoints We already used much fewer keypoints compared to prior unsupervised learning methods (Minderer et al., 2019). However, to further investigate the effect of this problem on keypiont-based methods by drastically decrease the number of keypoint used. We use $\frac{3}{16}$ of the number of points used in our original method and notice drop in performance as indicated in Figure 1.

3. Additional Visualizations

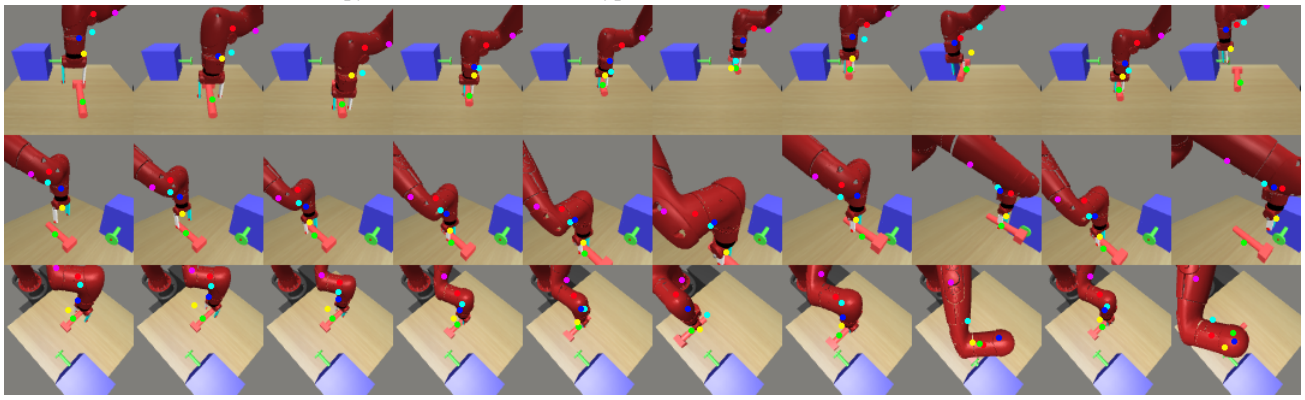
In Figure 2, we provide additional visualizations on several environments. In the pybullet ant environment as shown in Figure 2a, we can observe that the red point tracks the movement of upper red leg very consistently in all poses. In Figure 2b, a the green point tracks the handle of the hammer, whether it's on table top or in the gripper. Other points follow different joints of the robot arm respectively. When the hammer drops onto the table in the right most column, the point movement also reflects this change by moving itself away from those points corresponding to the arm. In Figure 2c, the purpose point moves with the cover of the box while the red point and yellow point follows different segments the end effector. Overall, our unsupervised learning method learns high-quality meaningful 3D keypoints for these 3D tasks.

References

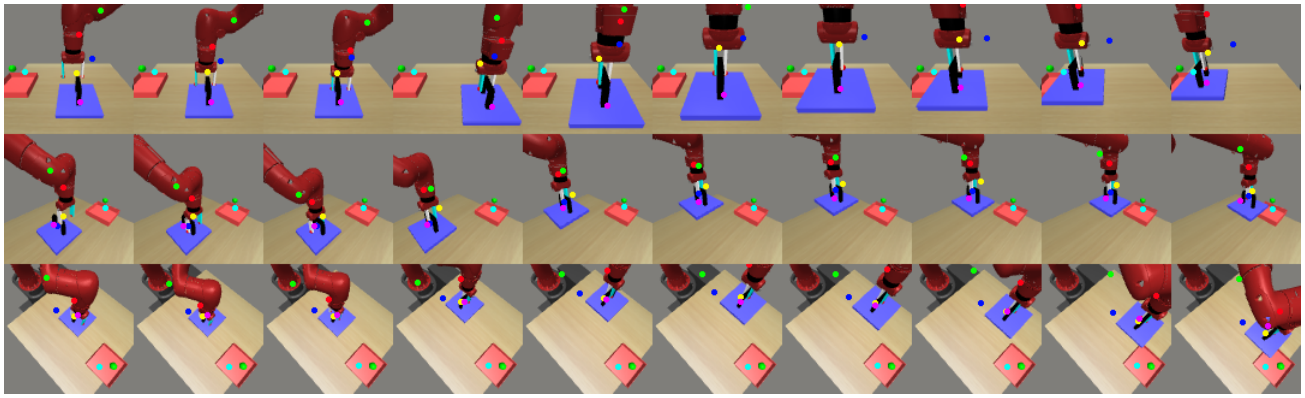
Minderer, M., Sun, C., Villegas, R., Cole, F., Murphy, K., and Lee, H. Unsupervised learning of object structure and dynamics from videos. *arXiv preprint arXiv:1906.07889*, 2019. 1, 2



(a) In pybullet ant, the learned keypoints track limbs of the locomotion robot



(b) In hammer manipulation, the learned keypoints track joints of arms as well as the hammer



(c) In box closing, the learned keypoints track the cover, the end effector and the box

Figure 2. Our unsupervised learning based Keypoint3D method learns high-quality 3D keypoints across the benchmarked manipulation tasks. We provide video visualizations of the 3d keypoints on the website https://buoyancy99.github.io/unsup-3d-keypoints/. Please refer to videos for better understanding of results.