# Unsupervised Learning of Visual 3D Keypoints for Control

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


## Abstract

Learning sensorimotor control policies from high-dimensional images crucially relies on the quality of the underlying visual representations. Prior works show that structured latent space such as visual keypoints often outperforms unstructured representations for robotic control. However, most of these representations, whether structured or unstructured are learned in a 2D space even though the control tasks are usually performed in a 3D environment. In this work, we propose a framework to learn such a 3D geometric structure directly from images in an end-to-end unsupervised manner. The input images are embedded into latent 3D keypoints via a differentiable encoder which is trained to optimize both a multi-view consistency loss and downstream task objective. These discovered 3D keypoints tend to meaningfully capture robot joints as well as object movements in a consistent manner across both time and 3D space. The proposed approach outperforms prior state-of-art methods across a variety of reinforcement learning benchmarks. Code and videos at https://buoyancy99.github.io/unsup-3d-keypoints/.


## 1. Introduction

Learning to act from raw, high-dimensional observations like images is arguably the only viable way to scale sensorimotor control to complex real-world setups. However, the main challenge lies in figuring out the representation of these observations upon which the agent's policy is to be learned. The conventional wisdom today is to learn these representations either via task supervision in an end-to-end manner (Mnih et al., 2015a; Lillicrap et al., 2016) or via auxiliary loss functions (Jaderberg et al., 2016; Pathak et al., 2017; Laskin et al., 2020b). Although such representation learning paradigms are successful in vision (Krizhevsky et al., 2012), speech (Oord et al., 2016) or language (Devlin et al., 2018), they suffer from two key issues in case of robot learning. First, the representations tend to become specific to both the task and the visual environment, thus, any change in the visual input affects the generalization of the learned representations as well as policy. Second, these latent features do not capture the fine-grained location and orientation of objects or robot joints which is indispensable for robotic control in complex scenarios. This is in contrast to sensorimotor representations in humans which have explicit notions of objects, their relationships, 3D reasoning, and geometry (Spelke & Kinzler, 2007).

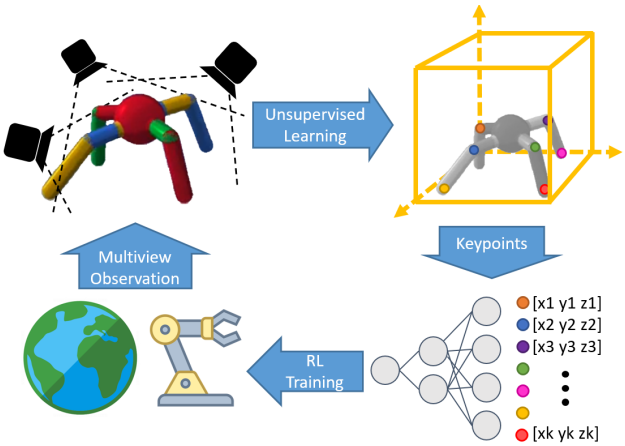



*Figure 1.* We propose an end-to-end framework for unsupervised learning of 3D keypoints from multi-view images. These keypoints are discovered to jointly optimize both multi-view reconstruction and downstream task performance. Our learned keypoints are consistent across 3D space as well as time.

Motivated by the need to capture fine-grained abstractions from visual data, a popular approach is to use keypoint-based representations. Keypoints are represented in Euclidean coordinates and provide a natural way to represent the kinematic structure of both the agent and the environment. Furthermore, keypoint- or particle-based representation provides an ideal way to represent deformable objects like cloth for precise robotic manipulation (Clegg et al., 2017; Sundaresan et al., 2020) but these keypoints are often hand-selected by human labelers which is expensive and hinders scalability. Some prior methods try to alleviate this problem (Minderer et al., 2019; Kulkarni et al., 2019) by

[1]UC Berkeley [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. 

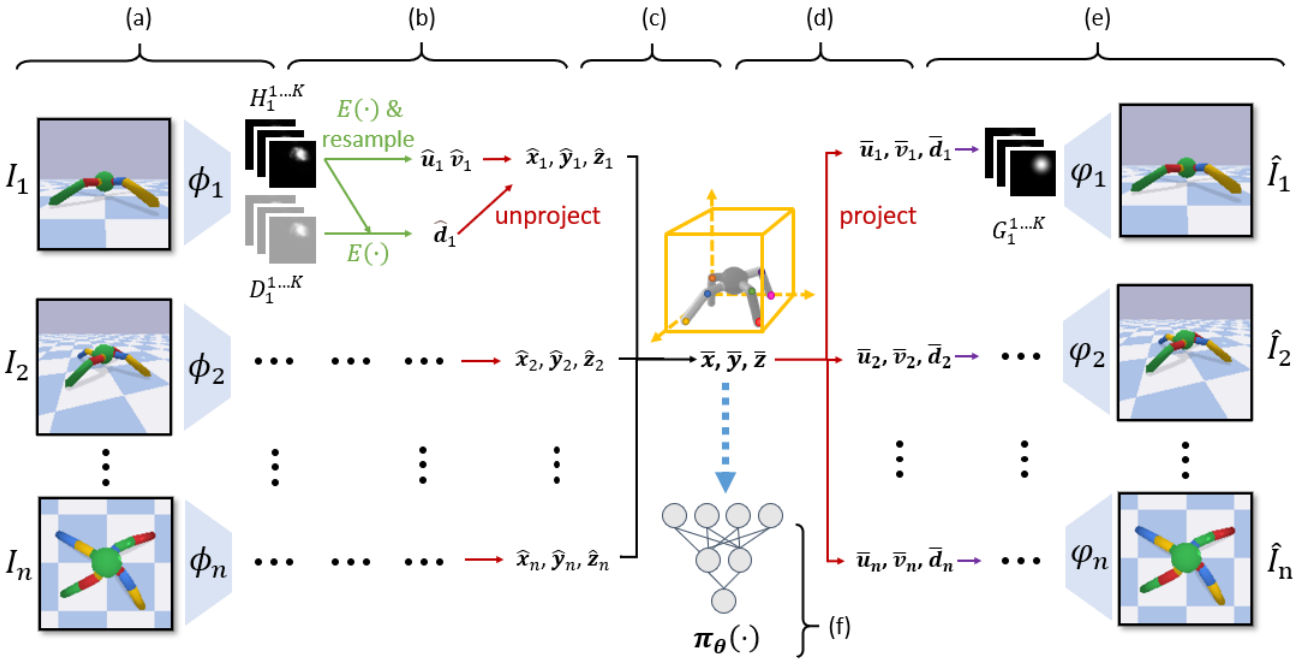



*Figure 2.* Overview of our *Keypoint3D* algorithm. (a) For each camera view, a fully convolutional neural network encodes the input image into K heat maps and depth maps. (b) We then treat these heat maps as probabilities to compute expectation of spatial $uv$ coordinates in camera plane. These expected values and the saptial variances are used to resample final $uv$ keypoint coordinates which adds noise that prevents the decoder from cheating to hide the input information in the relative locations $uv$ keypoints. We also take expectation of depth coordinate, $d$, using the same probability distribution. These $[u, v, d]$ coordinates are then unprojected into the world coordinate. (c) We take attention-weighted average of keypoint estimations from different camera views to get a single prediction in the world coordinate. (d) For decoding, we project predicted keypoints in world coordinate to $[u, v, d]$ in each camera plane. (e) Each keypoint coordinate is mapped to a gaussian map, where a 2D gussian is created with mean at $[u, v]$ and std inversely proportional to $d$. For each camera, gaussian maps are stacked together and passed into decoder to reconstruct observed pixels from the camera. (f) Together with reconstruction, we also jointly train a task MLP policy on top of predicted world coordinates via reinforcement learning.

learning keypoints in an unsupervised and task-agnostic manner, however, the keypoints are defined in the image coordinate space in 2D. This limits the agent's ability to perform 3D reasoning and occlusion handling which is vital in partially observable or even fully observable environments as the tasks are performed in a 3D world.

An alternative approach to capture 3D is to leverage structure from motion (Wei et al., 2020) to aggregate multi-view information. Recent works (Manuelli et al., 2019; 2020) learn keypoints by solving for 3D correspondence across views via multi-view stereo but the keypoint discovery phase is performed separately from control and not learned in an end-to-end manner. However, different tasks may demand different parts of the scene to be keypoints. For example, if a robot is to pick up a cup, its handle would be a keypoint but if something is to be poured into it, 3D keypoints should capture its interior circumference. How can we have the best of all worlds: fine-grained keypoints, 3D reasoning, and unsupervised, end-to-end, joint training with control?

In this work, we propose an end-to-end framework for unsupervised discovery of 3D keypoints from images that are learned directly via the task performance, shown in Figure 1. For such an approach to be general as well as successful, it should satisfy three properties: (a) **Consistency across 3D space**: the learned keypoints should capture the same 3D location in the world coordinate from different views of the same scene. (b) **Consistency across time**: the same keypoint should track the corresponding entity consistently across different timesteps of trajectory rollout. (c) **Joint learning with control**: 3D keypoints should be learned such that they are directly useful for the end-task performance. We conceptually integrate these properties within a common mathematical framework.

Given an input image from a camera view, we first predict keypoint locations and depth in the image space. These keypoints from different views of the same scene are then aggregated using camera information to obtain a global world coordinate of each keypoint via a differentiable unprojection operation. The consistency across views is learned via multi-view consistency loss that ensures the keypoint from different view maps to common world coordinate. This global coordinate is then projected back into the image plane

of each camera view to reconstruct the original input image of the same view. This setup creates a differentiable 3D keypoint bottleneck upon which the agent's control policy is trained via reinforcement learning. The keypoint extraction and agent's policy are optimized jointly in an end-to-end manner as shown in Figure 2. We refer to our unsupervised algorithm as *Keypoint3D* in short.

We evaluate Keypoint3D across a variety of reinforcement learning benchmark environments, and we perform the following analyses. We first investigate how well our Keypoint3D representations perform compared to other representations for RL. Second, we test the scalability to higher dimensional control problems. Third, we show that our Keypoint3D based policy is capable of manipulating deformable objects as evident from results on a task where a robot must put a scarf around a human mannequin. Finally, we show that our Keypoint3D representations generalize across tasks as well. Our method outperforms prior state-of-the-art across almost all the environments and our ablation study demonstrates its robustness across several design choices.

## 2. Unsupervised Learning of 3D Keypoints

We use a multi-view encoder-decoder architecture to learn 3D keypoints without supervision. Given $N$ cameras from different views of a same scene, we associate an encoder and a decoder with each camera. We provide three distinct sources of *unsupervised* training signal for learning of the Keypoint3D representation: (1) We force the predicted keypoints across encoders to be geometrically consistent in 3D by encouraging the different view-specific keypoint coordinates to unproject to the same point in 3D space. (2) We impose an image reconstruction loss, which penalizes inaccurate reconstructions from the decoder. (3) We use the reward incurred by the RL policy that takes as input the learned keypoints, and backpropagate the reward signal through the weights of the encoder.

### 2.1. Preliminaries

Let $I_n \in \mathbb{R}^{H\times W\times C}$ be the image observation from camera $n \in 1...N$ with extrinsic matrix $V_n$ and intrinsic matrix $P_n$. Let K be the number of keypoints we intend to detect. Keypoints are indexed with $k = 1...K$. For a 3D point $[x, y, z]^\top$ in world coordinate, we can use extrinsic matrix $V_n$ and perspective intrinsic matrix $P_n$ to project it to camera coordinate $[u, v, d]^\top$ for camera $n$, where $u, v \in [0, 1]$ is a normalized coordinate on camera plane and $d > 0$ is the depth value of that point from camera plane. Let the operator $\Omega_n : [x, y, z]^\top \to [u, v, d]^\top$ denote this projection, and let its inverse $\Omega_n^{-1} : [u, v, d]^\top \to [x, y, z]^\top$ denote the unprojection operator that maps a camera coordinate to a world coordinate. Both $\Omega_n$ and $\Omega_n^{-1}$ are differentiable and can be expressed analytically.

### 2.2. Differentiable 3D Keypoint Bottleneck

We want to leverage the spatial inductive bias of our fully convolutional auto-encoder yet ensure the latent learned latent representation is in the form of xyz coordinates rather than feature maps. To achieve this, we need a differentiable keypoint bottleneck that maps from dense maps to sparse coordinates. While a extraction for world coordinates is not possible from encoded feature space directly, (Jakab et al., 2018) offers a way to parameterize points in uv coordinates on camera plane using spatial probability. We can add depth prediction to the uv coordinate prediction and extract world coordinate using projection geometry.

**Keypoint Detector** For each camera $n$, we pass $I_n$ into a fully convolutional encoder $\phi_n$ to get $K$ confidence maps $C_n^k \in \mathbb{R}^{S\times S}, k = 1...K$ and depth maps $D_n^k \in \mathbb{R}^{S\times S}, k = 1...K$. For each confidence map $C_n^k$, we can take a spatial softmax to compute a probability heatmap $H_n^k$: $H_n^k(i, j) = \frac{\exp(C_n^k(i,j))}{\sum_{p=1}^{S}\sum_{q=1}^{S}\exp(C_n^k(p,q))}$. Each entry in heatmap $H_n^k \in \mathbb{R}^{S\times S}$ represents the probability of a 3D keypoint $k$ to be at that position on the 2D image plane if viewed from camera $n$. Each depth map $D_n^k$ is a dense prediction of distance from camera plane for 3D keypoint $k$ being at each position.

For each pair of heatmap $H_n^k$ and depth map $D_n^k$, we can extract the expected 3D position of the $k$-th keypoint in the $n_{th}$ camera's coordinate. We can calculate the expected $[u, v, d]$ camera coordinate over the probability map using the following equations. Note that crucially, we are taking an expectation over the map of *coordinates*, with weights given by the predicted heatmap values.

$$\mathbb{E}[u_n^k] = \frac{1}{S}\sum_{u,v} u \cdot H_n^k(u, v); \quad \mathbb{E}[v_n^k] = \frac{1}{S}\sum_{u,v} v \cdot H_n^k(u, v)$$

$$\mathbb{E}[d_n^k] = \sum_{u=1}^{S}\sum_{v=1}^{S} D_n^k(u, v) \cdot H_n^k(u, v)$$

Let $[\hat{u}_n^k, \hat{v}_n^k, \hat{d}_n^k]^\top = [\mathbb{E}[u_n^k], \mathbb{E}[v_n^k], \mathbb{E}[d_n^k]]^\top$. We found one way for the system to not capture meaningful 3D keypoints is cheating to hide the information about input image in the relative locations keypoint to each other. To avoid this issue, we do not use the exact keypoint locations predicted by the encoder but resample them from by assuming a gaussian distribution around the mean keypoint with standard deviation computed via spatially across the heatmap. This adds some stochasticity in the exact location of keypoint coordinates preventing the collapse. More details of this resampling process are in the appendix.

**Attention** After predicting keypoints in each of our $n$ camera coordinate frames, we need a way to combine the $n$

coordinates into one. A naive approach is to simply average the predictions from each view. However, certain keypoints might be occluded from certain view points, and might thus get predicted poorly. To address this, we repurpose our predicted confidence maps in order to compute weights for a weighted average. This allows us to ignore the less confident keypoint estimations which may hurt the task performance.

Recall that the $(i, j)$ entry of the confidence map $C_n^k$ denotes the predicted log likelihood of keypoint $k$ appearing at pixel $(i, j)$ in the image from camera $n$. We then assign a "confidence score" for the $n$-th encoder's prediction for keypoint $k$. This score is proportional to the mean of the confidence map $C_n^k$, and the scores of the $k$ keypoints are normalized to sum to 1 via a softmax:

$$A_n^k = \frac{\exp(\frac{1}{S^2}\sum_{p=1}^{S}\sum_{q=1}^{S} C_n^k(p,q))}{\sum_{i=1}^{K}\exp(\frac{1}{S^2}\sum_{p=1}^{S}\sum_{q=1}^{S} C_n^i(p,q))}$$

**Extracting world coordinates** Given predicted keypoints $[\hat{u}_n^k, \hat{v}_n^k, \hat{d}_n^k]^\top, n = 1...N, k = 1...K$ in camera coordinats, we can then unproject them to world coordinates using $[\hat{x}_n^k, \hat{y}_n^k, \hat{z}_n^k]^\top = \Omega_n^{-1}([\hat{u}_n^k, \hat{v}_n^k, \hat{d}_n^k]^\top)$. This is the predicted world coordinate of the $k^{th}$ keypoint conditioned on the image from camera $n$. For each keypoint, we have $N$ independent predictions from $N$ cameras. We then take an average over these predictions for each keypoint weighted with normalized confidence, that is

$$[\bar{x}^k, \bar{y}^k, \bar{z}^k]^\top = \sum_{n=1}^{N} \frac{A_n^k}{\sum_{m=1}^{N} A_m^k} \cdot [\hat{x}_n^k, \hat{y}_n^k, \hat{z}_n^k]^\top$$

**Keypoint Decoder** To also leverage spatial inductive bias for the decoder, we need to give keypoint coordinates spatial structure again before passing them into the decoder. This can be achieved by reprojecting 3D keypoints to each image view again and constructing a 2D gaussian for each keypoint on the camera plane. We reproject all $K$ keypoints in world coordinate to the $N$ camera planes to regain spatial structure before decoding. Using the projection operator, for each camera and keypoint we can get $[\bar{u}, \bar{v}, \bar{d}]^\top = \Omega_n([\bar{x}, \bar{y}, \bar{z}]^\top)$. To regain spatial structure, for each camera and keypoint we create Gaussian maps $G_n^k \in \mathbb{R}^{S\times S}$. Each Gaussian map is a 2D gaussian with mean $[\bar{u}, \bar{v}]^\top$ and covariance matrix $\mathbf{I}_2/\bar{d}$, where $\mathbf{I}_2$ is the $2 \times 2$ identity matrix. This matrix makes closer point have a larger spatial span in the gaussian map corresponding to that camera. Let us define averaged attention across all views as $\bar{A}^k = \frac{1}{N}\sum_{n=1}^{N} A_n^k$. The decoder $\psi_n$ for each camera takes the stacked Gaussian maps $G_n$ to predict $I_n$ where $G_n = K \cdot \text{stack}([G_n^1\bar{A}_n^1, G_n^2\bar{A}_n^2...G_n^K\bar{A}_n^K])$. The decoder $\psi_n$ then decodes the stacked Gaussian maps to reconstruct the observed image.

**Algorithm 1** Keypoint3D: RL with 3D Keypoint Bottleneck

$B_{obs} \leftarrow \emptyset$
**for** $update = 1, 2...$ **do**
  $B_{rollout} \leftarrow \emptyset$
  **for** $actor = 1, 2..., N$ **do**
    Run policy $\pi_{\theta_{old}}$ for $T$ steps to get $(s, a, r)_{1...T}$
    $B_{obs} \leftarrow s_{1...T} \cup B_{obs}$
    $B_{rollout} \leftarrow (s, a, r)_{1...T} \cup B_{rollout}$
  **end for**
  **for** $i = 1, 2...p$ **do**
    Optimize $L_{unsup}$ with $s \sim B_{obs}$ wrt $\theta_{ae}$
  **end for**
  Compute advantage estimates $\hat{A}_{1:N,1:T}$ for $B_{rollout}$
  **for** $epoch = 1, 2...q$ **do**
    Optimize $L_{policy} + L_{unsup}$ throughout $B_{rollout}$
  **end for**
  $\theta_{old} \leftarrow \theta$
**end for**

### 2.3. Losses for Training 3D Keypoint Encoder

Our Keypoint3D approach jointly optimizes the multi-view reconstruction via unsupervised learning and the task policy via reinforcement learning. We backpropagate the sum of unsupervised learning loss and policy loss to train the Keypoint3D pipeline. The unsupervised losses have three components as defined below.

**Multi-view Auto Encoding Loss** We encourage the keypoints to track important entities and structures. An Auto-encoding loss has been shown to be useful for this, as the architectural bottleneck forces the latent representation to capture the most salient aspects of the scene: $L_{ae} = \sum_{n=1}^{N} ||\psi_n(G_n) - I_n||_2$.

**Multi-view Consistency Loss** We enforce that for each keypoint $k$, the camera-frame coordinates of the keypoint $k$ from each of the $n$ encoders all have identical 3D world coordinates. We use a multi-view consistency loss to penalize disagreement between predictions from different views:

$$L_{multi} = \sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N} ||[\hat{x}_i^k, \hat{y}_i^k, \hat{z}_i^k]^\top - sg([\hat{x}_j^k, \hat{y}_j^k, \hat{z}_j^k]^\top)||_2$$

where $sg$ is the stop gradient operator.

**Seperation Loss** Finally, the keypoints should not collapse together and should keep a distance from each other to effectively track different objects in the scene:

$$L_{sep} = \frac{1}{K^2}\sum_{i=1}^{K}\sum_{j=1}^{K} \frac{1}{1 + M * ||[\bar{x}^i, \bar{y}^i, \bar{z}^i]^\top - [\bar{x}^j, \bar{y}^j, \bar{z}^j]^\top||_2^2}$$

where M is a positive value chosen to be 1000.

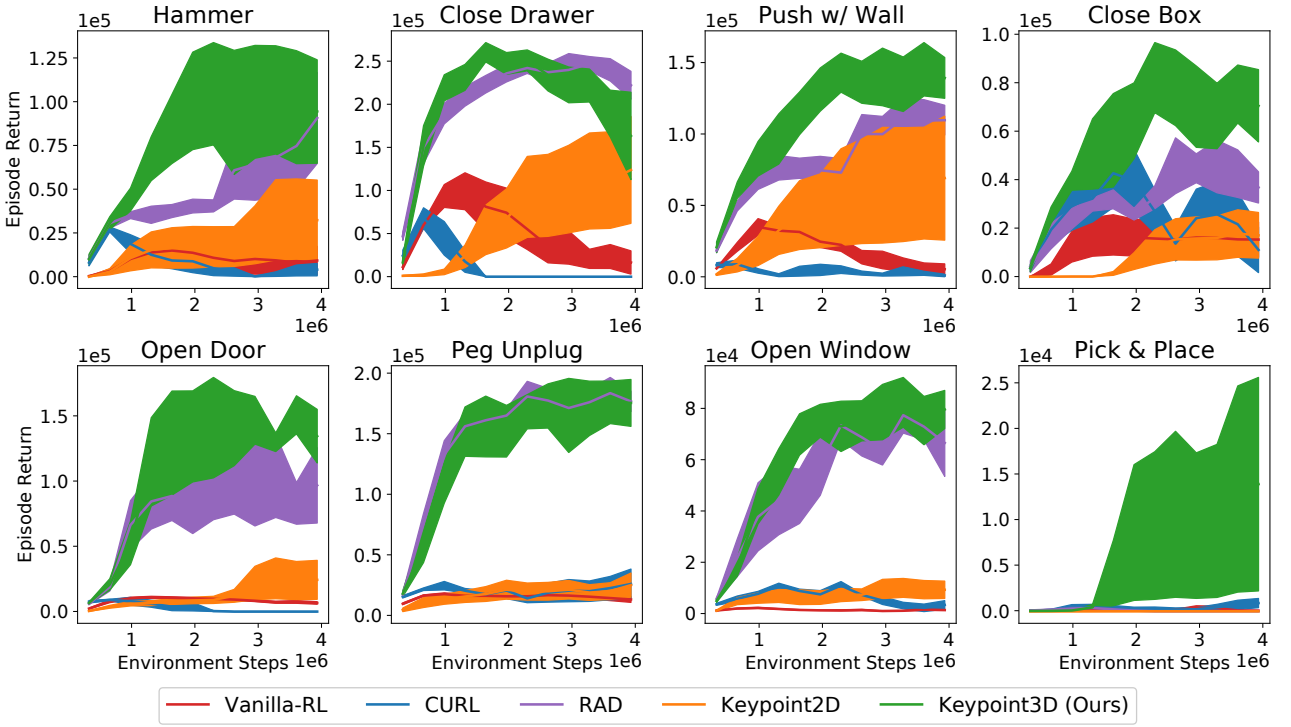



*Figure 3.* Plots show the performance of *Keypoint3D* on 8 metaworld environments with varying difficulty compared to different representation learning methods. Mean and standard error are shown across 4 random seeds for each environment. Our 3D keypoint method outperforms the strongest baseline, RAD, on 5 out of the 8 environments while being similiar in the rest.

Finally, the total unsupervised learning loss is:

$$L_{unsup} = \lambda_{ae} \cdot L_{ae} + \lambda_{multi} \cdot L_{multi} + \lambda_{sep} \cdot L_{sep}$$

### 2.4. Joint Learning of 3D Keypoints with Control

We integrate our 3D keypoint learning with PPO (Schulman et al., 2017) so a control policy can be trained jointly on top of learned keypoints. We keep an observation buffer for all observed images. Before each policy gradient update, we train unsupervised learning from observations sampled from this buffer for several steps, updating encoder and decoder parameters $\theta_{ae}$ with $L_{unsup}$. During policy gradient update, we optimize all parameters with the sum of policy loss $L_{policy}$ and unsupervised learning loss $L_{unsup}$. The policy is learned on top of keypoints represented in world coordinates which can have arbitrary scaling and can hurt the training process. To handle this, we *renormalize* the values of predicted 3D keypoints before feeding them to actor and critic network. Pseudo code for the method is provided in Algorithm 1 with our addition in magenta color.

**Temporal Consistency of 3D Keypoints** In many pixel-observation environments that requires temporal reasoning, a common technique is stacking adjacent frames as observation (Mnih et al., 2015b). In our method, we use the technique but also apply the same keypoint detector on each frame independently with shared encoder weights. The result is thus stacked keypoints across frames. We compute a difference vector indicating keypoint movements across adjacent frames and concatenate it with latest keypoints before feeding into fc layers of policy network. Note we further add data augmentation which forces the keypoint to be consistent across different appearances, thereby granting consistency over time. We also tried adding a temporal prediction loss but it didn't help because of sufficient temporal tracking signal from the data augmentations.

### 2.5. Data Augmentation as a Self-Supervisory Signal

Because the keypoint uv coordinates have explicit geometric interpretations, we can leverage data augmentation (Krizhevsky et al., 2012) to provide additional self-supervision. The core intuition is that, when the input image is translated, the predicted keypoints, projected on camera plane, should also get translated by the same amount. Because the random shift operator, $f$, effectively transforms heatmap coordinates from the original unshifted image, we apply a reverse transformation, $f^{-1}$, on the $\mathbb{E}[u_n^k]$, $\mathbb{E}[v_n^k]$ before unprojection to world coordinate. Before constructing gaussian maps, $f$ is applied to reprojected keypoints again to map them back to the shifted coordinate. The random translation with shift offset plays a critical role in getting high quality keypoints.

## 3. Experimental Setup

We evaluate our method on a variety of 3D control tasks with three-camera pixel observation. A reinforcement learning policy is trained jointly with unsupervised 3D keypoint learning as described in Section 2.4. We choose a set of 3D manipulation environments (Yu et al., 2019), a high-dof 3D locomotion environment (Coumans & Bai, 2016–2019), a customized soft-body environment and a meta-learning benchmark (Yu et al., 2019) to evaluate our method from different perspectives. These environments are originally developed for state based RL and are hard tasks for pixel based RL. We setup three third-person-view cameras in the scene such that most objects are visible from all three views. More details in the appendix.

We compare *Keypoint3D* with a variety of visual reinforcement learning algorithms. All the baselines, including Keypoint3D, are implemented on top of PPO (Schulman et al., 2017) with the same CNN architecture and algorithm hyperparameters. We stack images from all three view as observation for the baselines: (1) RAD (Laskin et al., 2020a) is the state-of-art method for pixel based reinforcement learning. It achieves high sample efficiency through data augmentation. (2) CURL (Laskin et al., 2020b) trains reinforcement learning on top of learned representation through contrastive learning. (3) Vanilla PPO is the original PPO algorithm (Schulman et al., 2017) with a CNN architecture to take in image observations. (4) Keypoint 2D is PPO-based implementation of (Minderer et al., 2019). We choose the number of 2D keypoint to be $3/2$ of that of 3D keypoints so both methods have same number of coordinate bits.

## 4. Results and Analysis

We investigate following questions: a) How well does our 3D keypoint representation compare to other representations for policy learning? b) How well does our 3D keypoint method scale to high dimensional control from high dimensional observations? c) How well does our method adapt to deformable object manipulation where keypoints are more desirable but harder to track? d) How well does our 3D keypoint representation generalize to unseen setups?

### 4.1. Accuracy and Efficieny of 3D Keypoints

We select 8 tasks of varying difficulty from meta-world, each featuring 50 random configurations. As shown in Figure 3, our method out performs the state-of-art method RAD by a margin in 5 out of the 8 environments while having similar performance in rest of the environments.

CURL and vanilla PPO performed poorly on all 8 visual metaworld environments. Vanilla PPO and RAD usually perform well in easier tasks where a camera attached to end effector will suffice. Such setups assume a strong inductive bias that a big change in task space, such as reward, is associated with large pixel space change, such as an object that occupies half of the camera image. In a third person view setup such inductive bias is no longer true and a change in task space can correspond to an insignificant change in pixel space, if at a distance from camera. RAD mitigates this problem by using data augmentation to let encoder focus on a larger set of global features yet still doesn't explicitly distill structure of the scene. In contrast, our method uses keypoints to distill the fine-grained movement of all moving components in the scene. 2D Keypoint performs better than CURL and Vanilla PPO but didn't outperform our 3D Keypoint method. This is likely because, in metaworld environments, objects can get very close to camera and show a large surface, which can be a problem for the 2D Keypoint representation (Minderer et al., 2019) as it uses a fixed spatial variance in Gaussian map. Our 3D Keypoint method significantly mitigates the problem by assigning large spatial variance in gaussian map for closer objects, telling the decoder the presence of a potential large surface.

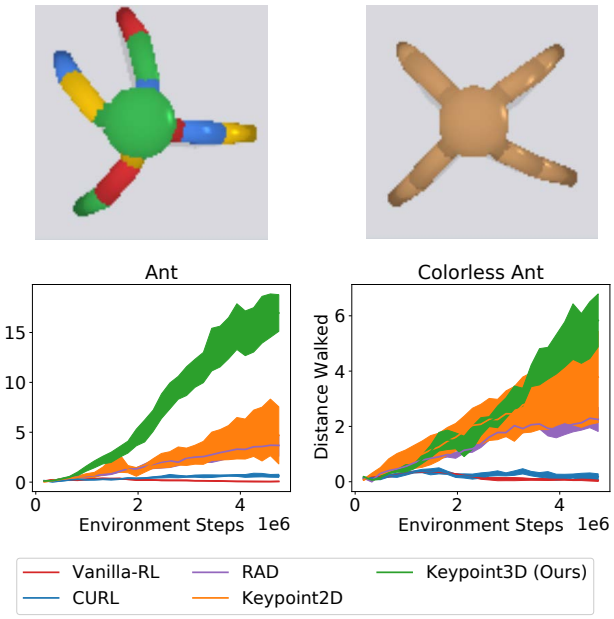



*Figure 4.* Benchmark reinforcement learning on Pybullet Ant (Left), a highly dynamical environment where fine-grained understanding of movable joints and parts are essential. Colorless Ant(Right) is a modified version without homogeneous limb colors to evaluate our method in low-texture setting.

### 4.2. Scaling to High-Dimensional Control

To evaluate our method on high dimensional control, we choose a highly dynamical 3D-locomotion environment, Pybullet (Coumans & Bai, 2016–2019) Ant, where fine-grained understanding of movable joints and parts are essential. As locomotion environments require temporal reasoning, we use a frame stack of 2. The original ant environment

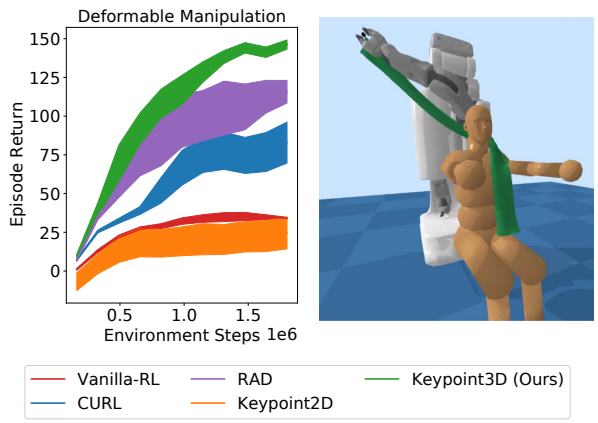



*Figure 5.* Left: Learning curve on scarf manipulation environment. Our Keypoint3D approach outperforms all the prior works consistently. Right: Scarf manipulation environment. The objective is to wind the scarf around human neck for one and a half full circle.

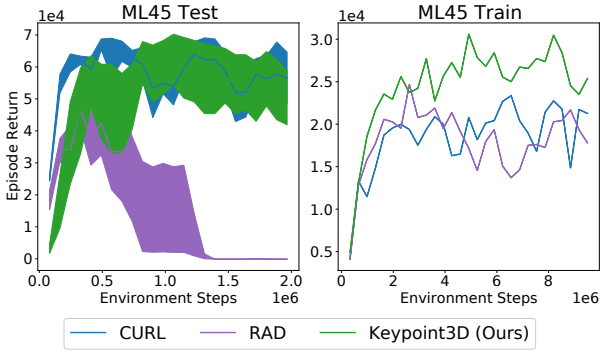



*Figure 6.* Transfer Learning result on ML45 multi-tasking benchmark. Our 3D Keypoint method out performs others by a margin in a training environment consist of 45 tasks. When finetuned on a 5-task test environment, both representation learning methods, CURL and 3D Keypoint, outperforms RAD.

in Pybullet assigns different colors to adjacent limbs. To further show that our method is robust to low-textured objects, we also benchmark on a colorless variant of ant.

As shown in Figure 4, our method significantly outperforms all baselines in both ant and its colorless variant. Our method, in particular, avoids local minimum as it better captures the structure of the robot as visualized in Figure 7, where the Keypoint3D captures the 3D movement of the ant robot in world coordinates. The ant environment, from pixels, requires the temporal knowledge of velocity of body and limb to apply control effectively. We believe the temporal consistency as described in Section 2.4 exactly achieved this, estimating the movement vector and feeding the mission-critical velocity information into our policy just like in state-space. Superiority of keypoint-based method is also reflected in the high performance of 2D keypoint baseline on the environment. On the other hand, Vanilla-RL, RAD and CURL lack explicit modeling of movement. CURL's contrastive learning will overly focus on pixels and fails to capture the fine-grained joint positions or movements that results in little pixel change.

### 4.3. Keypoints for Deformable Object Manipulation

We further evaluate our method on a customized 3D scarf manipulation environment based on (Erickson et al., 2020). In this environment, the objective is putting a scarf around human neck. This task requires both 3D reasoning of occlusion and understanding of the highly dynamical 3D soft object. Figure 5 shows the result of Keypoint3D on scarf manipulation is the best among all methods.

### 4.4. 3D Keypoints for Transfer Learning across Tasks

To test how well does our 3D keypoint representation generalize to unseen objects, we conduct a transfer learning experiment on the ml45 multi-task learning benchmark of metaworld (Yu et al., 2019). We followed the train/test split of the benchmark, first pre-train our method and baselines on 45 training environments featuring distinct objects for 10M steps. We then conduct transfer learning on 5 unseen test environments with the pre-trained weights for 2M steps. For each method, we freeze the pre-trained encoder and finetune the trained mlp part of the policy. We compare our method against the transfer result of RAD as well as CURL as shown in Figure 6. Our method outperforms all the baselines in the training environment for the same reason in the single-task setting of metaworld. Both our 3D Keypoint method and CURL shows great transfer result compared to RAD, which fails because it does not have a representation learning component.

### 4.5. Visualization of Discovered 3D Keypoints

To better understand how our 3D keypoint learning method capture meaningful 3D structure of the tasks, we visualize the learned keypoints by projecting them onto the camera plane of each view. Keypoints of the same color across different views correspond to the same keypoint in 3D space. We can also filter out a portion of the keypoints with a threshold of learned attention. Figure 7 illustrates that the learned keypoints with most attention consistently follow the movement of essential moving components in the scene. In the colorless ant environment, keypoints track the 12 limbs and joints of the ant robot throughout time despite being colorless. The same phenomenon can be observed in the metaworld environments. More visualizations are shown in the appendix.

## 5. Analysis and Ablation

In order to better understand the effect of different components on the sample efficiency of our method, we conduct ablative studies on two metaworld environments.

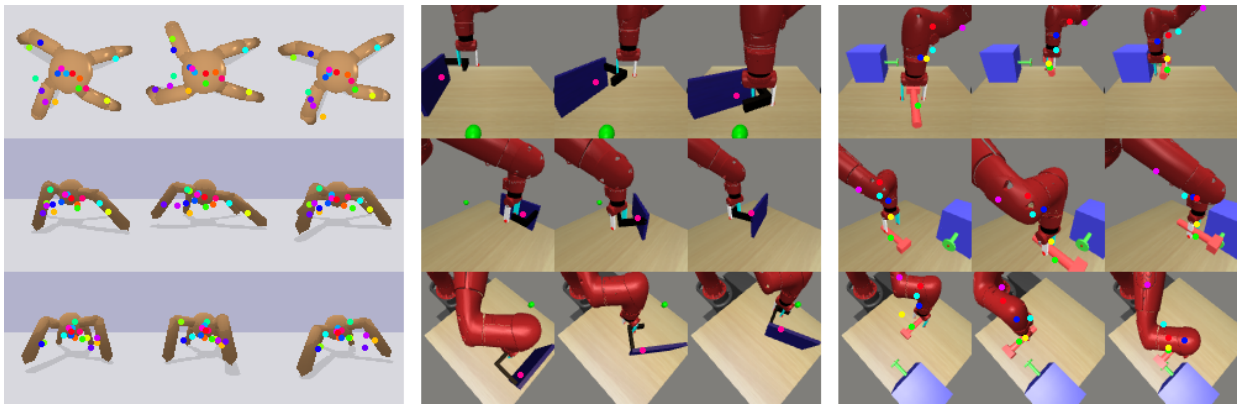


*Figure 7.* Visualization of the learned keypoints on colorless ant (left), metaworld close door (middle), metaworld hammer (right) environments. For the two metaworld environments, we filter keypoints with a threshold on learned keypoint attention. In colorless ant, the keypoints consistently track the limbs. In door the pink point with highest confidence tracks the movement of the door consistently. In hammer, the green point tracks hammer, while other points tracks the end-effector and sections of the arm respectively.

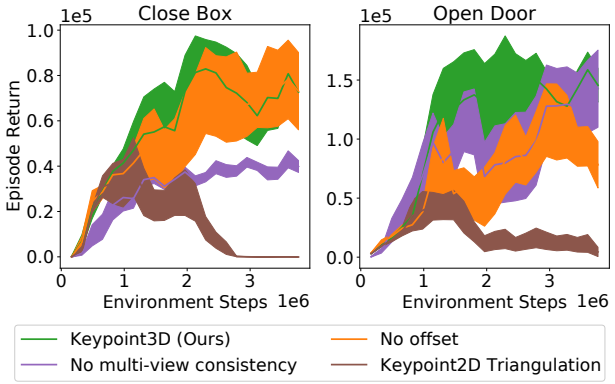



*Figure 8.* Ablations of our Keypoint3D approach with respect to different design choices. Variations include: remove camera offset in the cropping data-augmentation, remove multi-view consistency loss, map keypoints in 2D baseline to 3D space using triangulation.

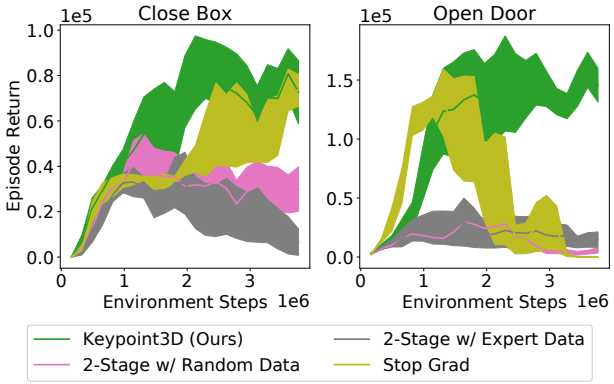



*Figure 9.* Analyzing the role of joint training of unsupervised keypoint learning with control policy. We compare to three different variants of training policy disjointly where keypoint training data is collected via random policy (2-stage w/ Random) and via expert policy (2-stage w/ Expert). Third approach is to keep the same pipeline as ours but stop backpropagation from policy branch. The results show that joint training is crucial.

**Effects of crop offset** Removing the random crop offset, described in the Section 2.5, significantly lowered our performance on the open door environment as shown in Figure 8. This is expected as the random shift of the input image should result in a corresponding change in the keypoint coordinate. When the crop offset is removed, randomly shifted keypoints will be incorrectly unprojected as we don't change the camera matrix.

**Effects of multi-view consistency** The ablation results for removing the multi-view consistency loss are shown in Figure 8. No multi-view consistency severely hurts the performance illustrating that 3D representation is indeed a better representation. We also experiment with a variant of Keypoint2D where we map all 3D keypoints to 3D coordinates using multi-view triangulation. The performace is even worse, indicating multi-view consistency is critical even if we use triangulation instead of depth predictor.

**Effects of joint training** To examine the benefit of training keypoint detection with policy jointly, we carry out three ablations as follows: (a) First train unsupervised 3D keypoint learning on random exploration data. Then train the policy with encoder weights frozen. (b) First train unsupervised 3D keypoint learning on expert data. Then train the policy with encoder weights frozen. c) We stop gradient of policy from backpropagating into encoder. (a)(b) are both 2-stage variants with offline data while (c) examines the effect of joint gradient with on-policy data. Our method outperforms all these three ablations as shown in Figure 9, showing joint training is important to achieve high sample efficiency as well as performance.

**Robustness under changing background** All of the previous experiments have static background due to the multi-

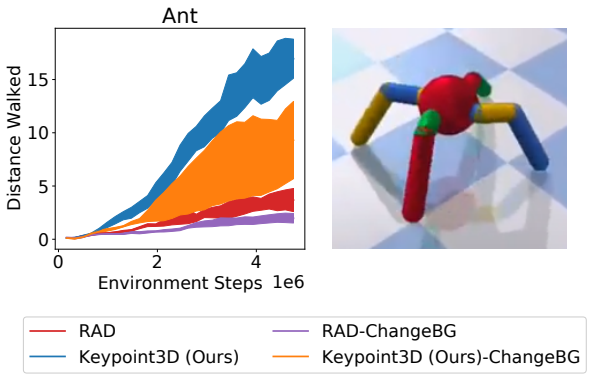



*Figure 10.* Ablation with respect to background changes in high-dimensional control. We show that Keypoint3D is robust to changes in the background unlike other methods because keypoints focus on the agent.

view setup. To evaluate the effectiveness of Keypoint3D under changing background, we run an ant environment variant with checkerboard floor. As shown in Figure 10, changing background of the environment makes the task harder from pixels for all methods. But our method still outperforms the strongest baseline, RAD.

## 6. Related Work

**Representation Learning for RL** Common approach in RL is to learn image representations via end-to-end CNN policy (Mnih et al., 2013; 2015b; 2016). Recent works enrich this representation via data augmentation (Kostrikov et al., 2020; Laskin et al., 2020a). Alternative to data augmentation is to learn representations via auxiliary losses like inverse model (Pathak et al., 2017), pixel change (Jaderberg et al., 2016), VAEs (Kingma & Welling, 2013; Burda et al., 2019), or contrastive loss (Laskin et al., 2020b).

**Unsupervised Keypoint Learning** Keypoints has been widely used as the representation of structure in image animation (Siarohin et al., 2020; 2019), video prediction (Kim et al., 2019; Minderer et al., 2019) or control (Kulkarni et al., 2019; Minderer et al., 2019). These works builds on a fully differentiable keypoint bottlebeck (Jakab et al., 2018) that can map a spatial probability map to images coordinates. They achieve their goal of animating a human or robot in source image by giving a target pose, represented by keypoints, before decoding. However, most of these works learn 2D keypoints in image space. In 3D setting, an object appearing to be a point at distance can be a large chunk of pixels when moving closer to camera. These methods would thus fail when the scene contains 3D movements.

**3D Keypoints Learning** Suwajanakorn et al. (2018) proposes a method to learn category specific 3D keypoints without supervision given a large dataset of rendered image and camera matrix tuples from many different views of the shapenet (Chang et al., 2015) dataset. The method learns category specific 3D keypoints without direct supervision through multi-view consistent pose prediction. It requires this large data set of rendered images from many angles centered around rigid objects. In reinforcement learning setting, however, a scene is dynamic rather than a single rigid object. Cameras are also usually fixed during training. (Zhao et al., 2020) learns to discover geometrically consistent 3D keypoints from pairs of images of object with rigid body transformation information. Tang et al. (2019) learns depth-aware keypoints through appearance and geometric matching from videos for autonomous driving.

**Keypoint Discovery for Control** Florence et al. (2018) propose to learn an object-centric dense correspondence model through contrastive learning. The process involves a dataset collected without supervision, by scanning target object from a variety of angles using a robotic arm and doing 3D reconstruction using the data. (Manuelli et al., 2020) samples points from this dense correspondence model as descriptor of an object and applies model predictive control on such keypoint based descriptor. (Manuelli et al., 2019) proposes to use semantic 3D keypoints as the object representation for category-level pick and place tasks instead of transformation from a fixed object template. This method, however, requires manually labeled keypoint dataset. Gao & Tedrake (2019) improves its result by combining these keypoints with dense geometry of object. In contrast, we learn 3D keypoints directly from policy learning loss in an unsupervised and end-to-end manner.

## 7. Conclusion

In this work, we presented a framework to learn useful 3D keypoints without supervision for continuous control. Our key insight is to leverage multi-view consistency with a world coordinate transform in the bottleneck layer for learning reliable keypoints. We jointly train unsupervised 3D keypoint learning in conjunction with reinforcement learning to achieve significant sample efficiency improvement in a variety of 3D control environments. The 3D keypoints learned by our algorithm are consistent across space and time. The keypoints offer several benefits as they make the control policy less dependent on the visual input, and hence, provide a great means for transfer learning even if the visual distribution changes a lot, for instance, transfer from simulation to real. We hope our method serves as a bridge between pixel domain and 3D control tasks.

## Acknowledgments

We thank Zackory Erickson, Alexander Clegg, and Charlie Kemp for fruitful discussions. This work was supported by NSF IIS-2024594 and NSF IIS-2024675.

## References


Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Large-scale study of curiosity-driven learning. *ICLR*, 2019. 9

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. Shapenet: An information-rich 3d model repository, 2015. 9

Clegg, A., Yu, W., Tan, J., Kemp, C. C., Turk, G., and Liu, C. K. Learning human behaviors for robot-assisted dressing. *arXiv preprint arXiv:1709.07033*, 2017. 1

Coumans, E. and Bai, Y. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019. 6

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

Erickson, Z., Gangaram, V., Kapusta, A., Liu, C. K., and Kemp, C. C. Assistive gym: A physics simulation framework for assistive robotics. *ICRA*, 2020. 7

Florence, P. R., Manuelli, L., and Tedrake, R. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*, 2018. 9

Gao, W. and Tedrake, R. kpam-sc: Generalizable manipulation planning using keypoint affordance and shape completion. *arXiv preprint arXiv:1909.06980*, 2019. 9

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016. 1, 9

Jakab, T., Gupta, A., Bilen, H., and Vedaldi, A. Unsupervised learning of object landmarks through conditional image generation. *arXiv preprint arXiv:1806.07823*, 2018. 3, 9

Kim, Y., Nam, S., Cho, I., and Kim, S. J. Unsupervised keypoint learning for guiding class-conditional video prediction. *arXiv preprint arXiv:1910.02027*, 2019. 9

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 9

Kostrikov, I., Yarats, D., and Fergus, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020. 9

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25: 1097–1105, 2012. 1, 5

Kulkarni, T. D., Gupta, A., Ionescu, C., Borgeaud, S., Reynolds, M., Zisserman, A., and Mnih, V. Unsupervised learning of object keypoints for perception and control. *Advances in neural information processing systems*, 32: 10724–10734, 2019. 1, 9

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020a. 6, 9

Laskin, M., Srinivas, A., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. *Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119*, 2020b. arXiv:2004.04136. 1, 6, 9

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

Manuelli, L., Gao, W., Florence, P., and Tedrake, R. kpam: Keypoint affordances for category-level robotic manipulation. *arXiv preprint arXiv:1903.06684*, 2019. 2, 9

Manuelli, L., Li, Y., Florence, P., and Tedrake, R. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. *arXiv preprint arXiv:2009.05085*, 2020. 2, 9

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, 6, 9, 11, 12

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 9

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 2015a. 1

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 2015b. 5, 9

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016. 9

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016. 1

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017. 1, 9

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. 5, 6

Siarohin, A., Lathuilière, S., Tulyakov, S., Ricci, E., and Sebe, N. Animating arbitrary objects via deep motion transfer, 2019. 9

Siarohin, A., Lathuilière, S., Tulyakov, S., Ricci, E., and Sebe, N. First order motion model for image animation. *arXiv preprint arXiv:2003.00196*, 2020. 9

Spelke, E. S. and Kinzler, K. D. Core knowledge. *Developmental science*, 10(1):89–96, 2007. 1

Sundaresan, P., Grannen, J., Thananjeyan, B., Balakrishna, A., Laskey, M., Stone, K., Gonzalez, J. E., and Goldberg, K. Learning rope manipulation policies using dense object descriptors trained on synthetic depth data. In *ICRA*, 2020. 1

Suwajanakorn, S., Snavely, N., Tompson, J., and Norouzi, M. Discovery of latent 3d keypoints via end-to-end geometric reasoning. *CoRR*, abs/1807.03146, 2018. 9

Tang, J., Ambrus, R., Guizilini, V., Pillai, S., Kim, H., Jensfelt, P., and Gaidon, A. Self-supervised 3d keypoint learning for ego-motion estimation. *arXiv preprint arXiv:1912.03426*, 2019. 9

Wei, X., Zhang, Y., Li, Z., Fu, Y., and Xue, X. Deepsfm: Structure from motion via deep bundle adjustment. In *European conference on computer vision*, pp. 230–247. Springer, 2020. 2

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2019. 6, 7

Zhao, W., Zhang, S., Guan, Z., Zhao, W., Peng, J., and Fan, J. Learning deep network for detecting 3d object keypoints and 6d poses. In *CVPR*, 2020. 9

# A. 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 $\sigma_n^k$, 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 $\nu$. 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.

**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 white. In scarf environment, such state is pr2 robot's arm

| **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 ($\epsilon$) | 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 $\nu$ when no augmentation | $5e-4$ | $5e-4$ |
| noise $\nu$ when augmented | 0.0 | 0.0 |
| Autoencoding weight ($\lambda_{ae}$) | 5.0 | 5.0 |
| Multiview weight ($\lambda_{multi}$) | 0.05 | 0.05 |
| Seperation weight ($\lambda_{sep}$) | 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/.

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.

## B. 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 11 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.

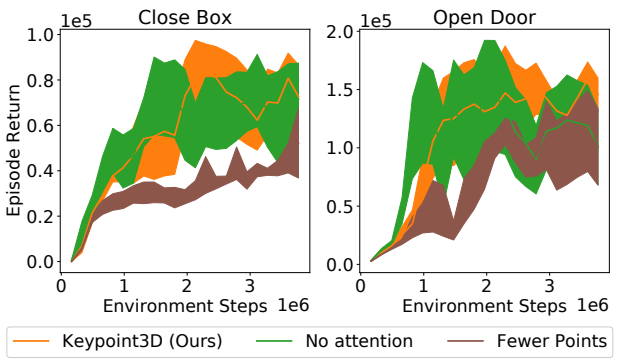
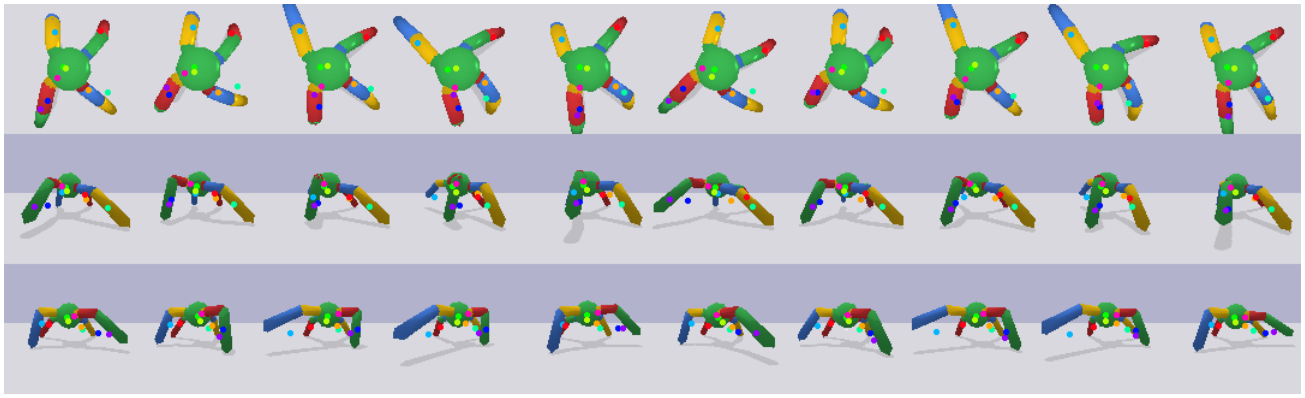



*Figure 11.* Additional ablations of our Keypoint3D approach with respect to different design choices. Variations include: removing attention, using fewer keypoints.

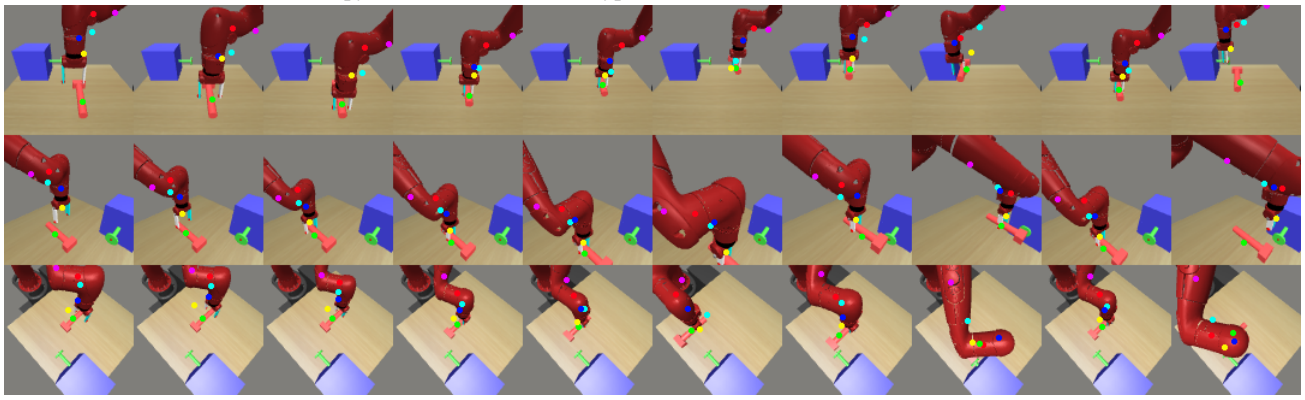
**Effects of Effects of number of keypoints** We already used much fewer keypoints compared to prior unsupervised learning methods (Minderer et al., 2019). However, to further investigate the effect of this problem on keypiont-based methods by drastically decrease the number of keypoint used. We use $\frac{3}{16}$ of the number of points used in our original method and notice drop in performance as indicated in Figure 11.

## C. Additional Visualizations

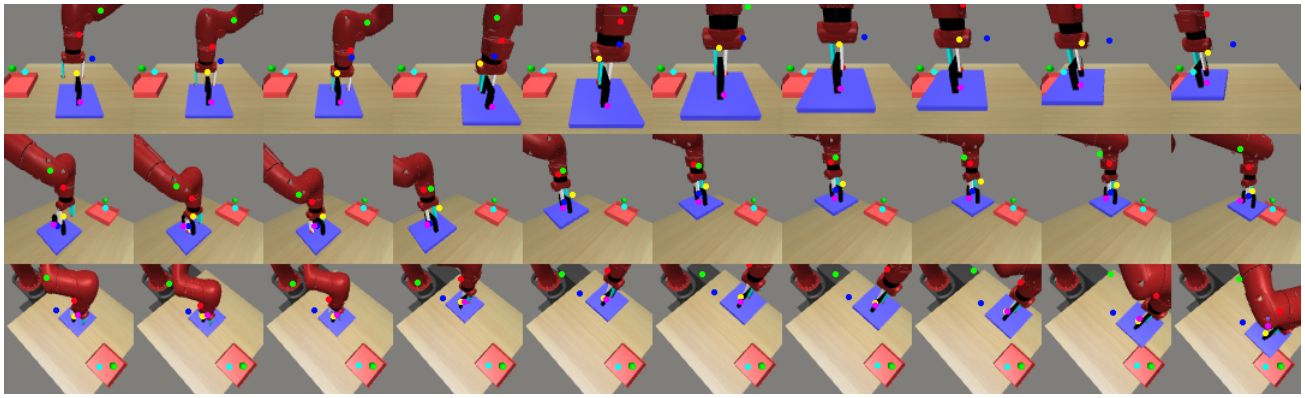
In Figure 12, we provide additional visualizations on several environments. In the pybullet ant environment as shown in Figure 12a, we can observe that the red point tracks the movement of upper red leg very consistently in all poses. In Figure 12b, a the green point tracks the handle of the hammer, 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 12c, 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.



(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 12.* 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.