Adapt3R: Adaptive 3D Scene Representation for Domain Transfer in Imitation Learning

Albert Wilcox[1,2], **Mohamed Ghanem**[1], **Masoud Moghani**[3], **Pierre Barroso**[2], **Benjamin Joffe**[1,2], **Animesh Garg**[1]

[1]Georgia Institute of Technology [2]Georgia Tech Research Institute [3]University of Toronto

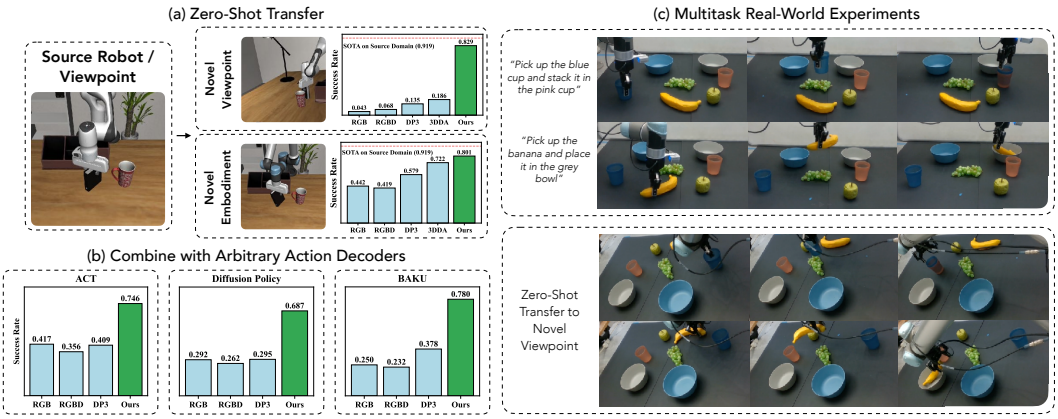


Figure 1: (a) Adapt3R facilitates zero-shot transfer to novel embodiments and viewpoints. (b) Adapt3R can be trained end-to-end as the encoder for a wide variety of imitation learning algorithms. (c) In a real-world multitask imitation learning benchmark, Adapt3R enables zero-shot transfer to an unseen camera pose.

Abstract: Imitation Learning can train robots to perform complex and diverse manipulation tasks, but learned policies are brittle with observations outside of the training distribution. 3D scene representations that incorporate observations from calibrated RGBD cameras have been proposed as a way to mitigate this, but in our evaluations with unseen embodiments and camera viewpoints they show only modest improvement. To address those challenges, we propose Adapt3R, a general-purpose 3D observation encoder which synthesizes data from calibrated RGBD cameras into a vector that can be used as conditioning for arbitrary IL algorithms. The key idea is to use a pretrained 2D backbone to extract semantic information, using 3D only as a medium to localize this information with respect to the end-effector. We show across 93 simulated and 6 real tasks that when trained end-to-end with a variety of IL algorithms, Adapt3R maintains these algorithms' learning capacity while enabling zero-shot transfer to novel embodiments and camera poses. For more results, visit https://pairlab.github.io/Adapt3R.

Keywords: Imitation Learning, 3D Perception, Cross-Embodiment Learning

1 Introduction

An important goal of the robot learning community is to train generalist agents that can solve a wide variety of tasks, as others have done for computer vision [1, 2, 3, 4, 5, 6, 7] and natural language processing (NLP) [8, 9, 10, 11]. While recent work in multitask imitation learning has achieved impressive results when evaluated on in-distribution domains [12, 13, 14, 15, 16, 17, 18], these policies are notoriously brittle when faced with changes in factors such as camera pose, and are not able to reliably transfer to new robot embodiments. Even large scale data collection efforts in robotics [19, 20, 21] yield datasets orders of magnitude smaller than those used for vision and NLP that are so

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

diverse and sparse that it's difficult to learn unified representations across them. Ultimately, collecting enough data to cover the full deployment distribution is currently infeasible.

To train generalist robots, we must design architectures that lend themselves to generalizing beyond training distributions. There is a wide variety of work that sets out to do this, but we specifically focus on those methods that do so using 3D representations. Several lines of work have considered the application of 3D scene representations to end-to-end IL, but these generally are limited to key pose prediction [14, 15, 22, 23, 24], make architectural choices that aren't suitable for out-of-domain transfer [25, 26, 27], or require per-task tuning that is not scalable to multitask settings [28, 29, 25].

The goal of this paper is to present an architecture explicitly designed for transfer to novel embodiments and camera viewpoints. We introduce Adapt3R, an Adaptive 3D Scene Representation. Adapt3R is an observation backbone that synthesizes outputs from calibrated RGBD cameras into a conditioning vector to be used interchangeably with a variety of IL algorithms when trained end-to-end. The key insight behind Adapt3R is to *shift all of the semantic reasoning to a 2D backbone* and use the 3D information *specifically to localize that semantic information with respect to the end-effector*. We do this by aligning feature volumes from a pretrained 2D foundation model with 3D point cloud points and, after carefully post-processing the point cloud, using a learned attention pooling operation to reduce it to a single conditioning vector. Using semantic features from a 2D foundation model allows us to bypass the data-intensive process of learning them directly from point clouds, and the post-processing helps the attention pooling step to reason about the points.

In summary, we contribute the following:

- The Adapt3R architecture, which extracts conditioning vectors from RGBD inputs to be used interchangeably with a variety of IL algorithms.
- Experiments showing that Adapt3R can combine with a variety of IL algorithms to achieve strong multitask performance and perform high-precision insertions across 93 simulated and 6 real tasks, and maintain this performance with unseen embodiments and camera viewpoints, including a 43.8% improvement over the next best baseline in our real experiments.

2 Related Work

2.1 Imitation Learning for Robotic Manipulation

Imitation learning (IL) [30] began with policies mapping observations to actions [30, 31, 32, 33] and has since advanced to energy-based models [34], diffusion models [35, 27, 25, 24], and transformer architectures [36, 18, 17, 37], but these methods still suffer outside of their training distributions. Even methods that train on large-scale datasets [12, 13, 38, 39, 40, 41, 42, 43, 21, 44] are brittle to changes in environment such as camera pose or novel embodiments. Recent methods have found success transferring to novel embodiments and camera viewpoints through inpainting [45, 46], but these require expensive inference-time processing or per-embodiment finetuning. Rather than attempting to achieve generalization purely through scaling, this paper introduces an architecture which uses 3D representations as a bridge between settings with different embodiments and camera poses.

2.2 3D Representations in Robot Manipulation

While the robotics community mostly uses 2D inputs for IL pipelines, a growing body of recent work has achieved success with 3D inputs. PerAct [14] and RVT [15, 22] use voxels and simulated camera views respectively to predict end effector keyposes, but this framework assumes tasks that can be neatly decomposed into a sequence of key poses, precluding tasks requiring higher-frequency control or nonlinear trajectories such as folding. Another line of work embeds outputs of foundation models in implicit scene representations [47, 48, 49], but these require expensive inference-time training, also precluding high-frequency control.

Most similar to Adapt3R is a body of work which uses point clouds, a relatively cheap-to-construct scene representation, as input to a policy. Gervet et al. [23] lifts outputs from a frozen pretrained backbone into a point cloud before using this information to predict key poses, and Xian et al.

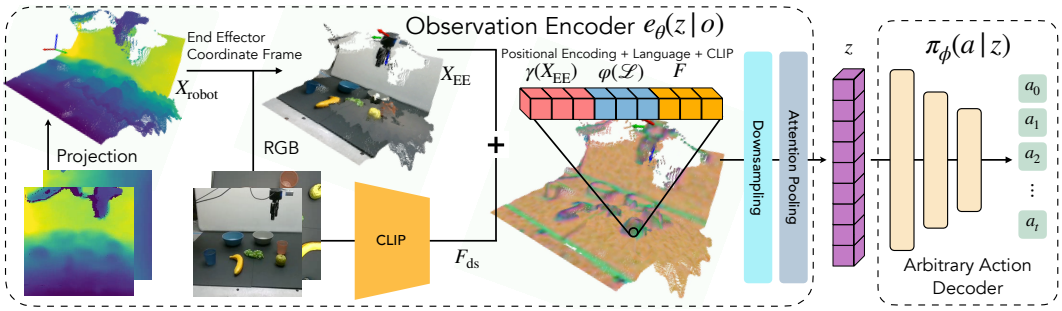


Figure 2: Adapt3R extracts scene representations from RGBD inputs for use with a variety of imitation learning algorithms. It lifts pre-trained foundation model features into a point cloud, carefully processes that point cloud, and uses attention pooling to compress it into a single vector z to be used as conditioning for end-to-end learning.

[24] builds on this by learning a diffusion policy planner to connect the predicted key poses. 3D Diffuser Actor [27] uses a similar scene representation, self attends between points in the scene and cross attends between scene points, action position samples, language tokens and proprioception. In particular, self-attending between points in the scene requires the agent to reason about its 3D geometry, which is difficult to do in a low-data regime and prone to overfitting. Adapt3R, on the other hand, is designed to use the 3D information only to localize the semantic information with respect to the end effector, enabling better transfer to novel camera viewpoints.

The 3D Diffusion Policy (DP3) line of work [25, 26] utilizes colorless point clouds as input, followed by a PointNet [50, 51] which pools the cloud into a single vector to be used as conditioning for a diffusion policy. While this design allows generalization to novel object appearances in a single-task setup, its omission of the rich semantic information available to RGB sensors causes it to suffer in multitask or high-precision settings, and its reliance on 3D geometry makes it suffer when evaluated on out-of-distribution camera poses and robot embodiments. Unlike DP3, Adapt3R uses semantic information from a pre-trained 2D vision backbone to enable multitask IL and alleviate the over-reliance on learning 3D geometric information from scratch. Another work uses sparse convolutions to encode point clouds before feeding them to a diffusion policy [52], but this struggles with small camera changes since it extracts semantic information from raw point cloud data.

A final line of work similar to Adapt3R uses DINO [4] features lifted into a 3D representation to achieve impressive instance generalization results [28, 29], but these works rely on hand-selecting reference features for each task, limiting its scalability to multitask setups or long-horizon tasks with several relevant objects. Adapt3R alleviates this through the attention operation described in Section 3.2, which automatically learns these features during end-to-end policy learning. A detailed comparison to related works is available in Appendix A.

3 Adapt3R Architecture

Unlike prior methods, which in general make architectural choices leading the policy to infer semantic information from the point cloud, the key idea behind Adapt3R is to entirely *offload the extraction of semantic information to the 2D backbone*, and instead use *3D information to localize the semantic information* with respect to the end-effector. Adapt3R first lifts RGBD images into points paired with semantic and spatial information as described in Section 3.1. Next, it computes an attention map over points in the cloud in order to reduce the cloud into a single vector as described in Section 3.2. Finally, Adapt3R uses that vector as conditioning information for arbitrary downstream action decoders as described in Section 3.3, and the full pipeline is trained end-to-end, as shown in Figure 2.

Problem Setup: We assume access to a dataset $\mathcal{D} = \{\mathcal{L}_i, (o_{i,1}, a_{i,1}), \ldots, (o_{i,T_i}, a_{i,T_i})\}_{i=1}^N$ consisting of N expert demonstrations with corresponding natural language instructions $\mathcal{L}_i$. Each observation $o_{i,t} = (\mathcal{I}, \mathcal{R}, \mathcal{P})$ is a tuple consisting of a set of RGBD images $\mathcal{I} = \left\{I_1, \ldots, I_{n_{\text{cam}}} \in \mathbb{R}^{H \times W \times 4}\right\}$ streaming from n_{cam} cameras. We assume access to a set of corresponding camera calibration matrices mapping 2D image coordinates and depth values to 3D coordinates in the robot's base frame, and proprioceptive information including the end effector pose. Adapt3R's goal is to learn an encoder

$e_\theta(z|o)$ that subsequently maximizes the success rate of a policy $\pi_\phi(A|z)$ conditioned on e_θ's outputs when the two are trained together end-to-end.

3.1 3D Scene Representation

We construct a 3D scene representation with the necessary information and structure for our end goal of compressing it into a single compact conditioning vector. Similarly to [27, 23, 24, 29, 28], we use camera calibration data to lift semantic features from several viewpoints into a single fused point cloud. This allows Adapt3R to reason about the scene without extracting semantic information from the point cloud, which is prone to overfitting on smaller datasets.

Specifically, for each image I_i, we extract an intermediate feature volume using a pretrained CLIP ResNet [53] and project it to feature $F_i \in \mathbb{R}^{h\times w\times d}$, where $h < H$ and $w < W$. We deproject each image into a point cloud, transform it into the robot base frame, and resize it to $X_i \in \mathbb{R}^{h\times w\times 3}$ using nearest neighbors interpolation. Finally, we flatten and concatenate these to construct features $F \in \mathbb{R}^{m\times d}$ and point locations $X_{\text{robot}} \in \mathbb{R}^{m\times 3}$ with $m = n_{\text{cam}}hw$.

End Effector Coordinate Frames: Wrist camera observations are often used to help robots reason about the relative positions of the end effectors and target objects [54]. Applying the same intuition to point clouds, we use proprioceptive information to transform them into the end effector's coordinate frame, giving us the new point cloud X_{EE}. Liu et al. [55] notes that this change can lead to improved data efficiency, but we observe that it is especially helpful for cross-embodiment learning, where it is important to reason about the relative positions of the end effector and the scene without overfitting to the training distribution embodiment.

Positional Encoding: As described in [56, 57], neural networks are not well suited to fitting data with high-frequency variation. In robotics, where a small difference in relative position between the end effector and object can determine the success of a rollout, this issue is especially important. Thus, as in [57] we encode each 3D point using a Fourier basis of increasing frequencies, mapping raw positions into a higher-dimensional space that is more amenable to learning complex functions. We apply $\gamma(x) = (\sin(2^0\pi x), \cos(2^0\pi x), \ldots, \sin(2^{L-1}\pi x), \cos(2^{L-1}\pi x))$ with $L = 10$, which lifts the positions into $\hat{x} \in \mathbb{R}^{60}$. Let $\hat{X}_{\text{EE}} = \gamma(X_{\text{EE}})$ denote the lifted point cloud.

3.2 Point Cloud Reduction

After constructing the 3D scene representation as described in Section 3.1, we reduce it into a vector to be used as conditioning for the downstream policy. Our method is designed to do so without extracting semantic information from the relative locations of the points, instead using the information from the 2D backbone, and using 3D only to localize this information with respect to the robot. The key steps in this process are cropping, downsampling, and the final learned reduction.

Cropping: Rather than carefully selecting tight bounds [25, 58] or forgoing cropping altogether [27, 26], we take an intermediate stance, noting that especially with novel camera poses some cropping can help to ignore OOD background points. Specifically, we crop scenes such that the entire table is visible but objects outside the table are not. Next, since the positive z axis points outward from the end-effector in X_{EE}, we filter points with negative z values, cropping out most of the arm and aiding cross-embodiment transfer. More details and a sensitivity study are in Appendix D.

Downsampling: When changing camera poses the distribution of points on task-relevant and task-irrelevant (eg. the table) objects changes drastically, and we address this through a strategic choice in downsampling algorithm. Unlike [25, 26], which performs furthest point sampling based on Cartesian coordinates, we adopt the approach of Ke et al. [27] and downsample according to ℓ_2 distance between the features F. We find this strategy to yield point clouds that are more concentrated around the task-relevant objects and more robust to changes in camera pose, providing visualizations of this in Appendix D. We downsample to $\hat{X}_{\text{EE,ds}} \in \mathbb{R}^{p\times 60}$ and $F_{\text{ds}} \in \mathbb{R}^{p\times d}$ where $p < m$.

Language Embeddings: Prior work has found FiLM [59] modulation based on language inputs in the RGB backbone to help policies reason about their tasks in multitask settings [60, 18]. Likewise, we concatenate projected CLIP language embeddings $\varphi(\mathcal{L}) \in \mathbb{R}^d$ to the representations of the points

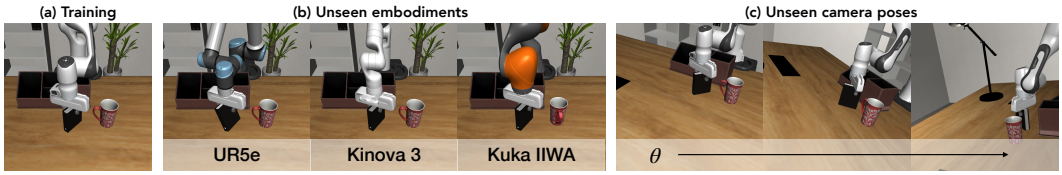


Figure 3: We train on the Franka Panda and viewpoint shown in (a). Then, we evaluate zero-shot with the UR5e, Kinova3 and IIWA (b) embodiments, and unseen camera poses (c).

in our point cloud. We define our final point cloud $P \in \mathbb{R}^{p \times (2d+60)}$ to be the concatenation of positional encodings $\hat{X}_{\text{EE,ds}}$, feature vectors F_{ds} and language embeddings $\varphi(\mathcal{L})$.

Encoding Extraction: After post processing our point cloud, we must reduce it into a conditioning vector without risk of overfitting to specific attributes of the training scenes. We learn an attention map over the points and use the resulting attention pooling operation to output a final vector z which we use as conditioning for the final policy. Concretely we learn key and value MLPs $K_\theta : \mathbb{R}^{2d+60} \to \mathbb{R}^{d_k}$ and $V_\theta : \mathbb{R}^{2d+60} \to \mathbb{R}^{d_e}$, and learn a fixed query $q \in \mathbb{R}^{d_k}$. The final embedding z is defined to be $z = \text{softmax}\left(\frac{qK_\theta(P)}{\sqrt{d_k}}\right) V_\theta(P)$. We find that in settings with very large changes in camera pose, this operation doesn't attned to out-of-distribution points, which helps with generalization to the new viewpoint. Visualizations of these attention maps are provided in the supplementary material.

3.3 Policy Learning

After extracting the encoding vector z, we use it to train a policy. Adapt3R is designed as a modular encoder compatible with a range of downstream action decoders, and we combine it with Action Chunking Transformer (ACT) [36], Diffusion Policy (DP) [35], and BAKU [18]. For each method we use Adapt3R to extract z from the observations and use it as conditioning to the policy. This process is described for each policy choice in greater detail in Appendix B.

4 Simulated Experiments

In this section, we set out to answer the following questions: (1) Does Adapt3R have the capacity to achieve strong performance in multitask and high-precision settings? (2) How does Adapt3R perform in zero-shot cross-embodiment settings compared with existing encoders? (3) How does Adapt3R perform in zero-shot camera viewpoint change experiments compared with existing encoders? (4) How do the various design decisions described in Section 3 affect Adapt3R's performance?

Benchmarks – **LIBERO-90** [60] is a multitask learning benchmark consisting of 90 rigid-body and articulated manipulation tasks which we use to study Adapt3R's multitask reasoning capabilities. **MimicGen** [61] includes 3 difficult insertion tasks, and we use it to study Adapt3R's suitability for settings requiring precise control. Further details about both benchmarks are available in the Appendix C.

4.1 Baselines

The goal of these experiments is to understand how a variety of observation backbones affect the properties of the final learned policy. To this end, for each of the action decoders mentioned in Section 3.3, our baselines run the same algorithm while replacing the observation encoder e_θ with the following backbones: **(1) RGB**: The CNN backbone of a ResNet-18 [62] with pre-trained ImageNet weights [1] followed by a spatial softmax [63], fine-tuned. **(2) RGBD**: Similar to RGB, but with an added depth channel. **(3) DP3**: The encoder from 3D Diffusion Policy [25], which lifts the depth information into a colorless point cloud and processes that cloud into a conditioning vector using PointNet [50]. Finally, we compare to **3D Diffuser Actor (3DDA)** [27], which lifts a semantic feature map into the point cloud and denoises action sequences (similarly with DP) by self-attending on the set of points and cross-attending between points and noisy action sequences. DP3 and 3DDA have access to the same world frame cropping as Adapt3R.

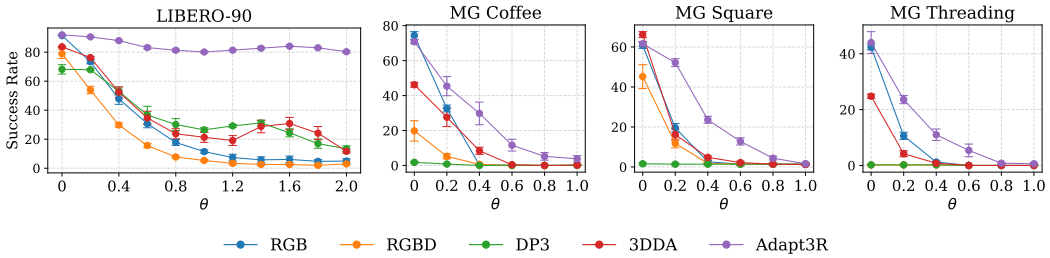


Figure 4: Unseen Camera Pose. We rotate the scene camera by θ radians about the vertical axis through the end-effector starting position. LIBERO-90 results use BAKU and MimicGen (MG) results use DP.

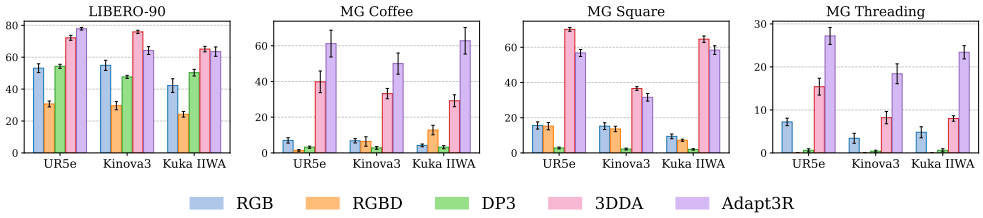


Figure 5: Cross Embodiment. We evaluate zero-shot with three unseen embodiments. LIBERO-90 results aggregate across all action decoders and MimicGen (MG) results use DP. Adapt3R and 3DDA consistently outperform comparisons, indicating that semantic-aligned point clouds are conducive to embodiment transfer.

All baselines have access to the same set of camera inputs, camera calibration data, proprioceptive information and language instructions as Adapt3R. We train 5 seeds for each algorithm variation and perform 10 evaluation rollouts per environment per seed for LIBERO and 100 rollouts for MimicGen. We present aggregate results here and full numerical results in Appendix D.

4.2 In-Distribution Behavior Cloning

In this experiment, we set out to understand whether Adapt3R has the modeling capacity to achieve strong success rates in multitask settings and high-precision insertion tasks.

Table 1: In-distribution Success Rate. LIBERO-90 results are aggregated across ACT, BAKU and DP, and MG results are from DP.

	RGB	RGBD	DP3	3DDA	Adapt3R
LIBERO-90	**90.9**	78.8	70.7	83.7	**90.0**
MG-Coffee	**74.4**	19.8	1.8	46.2	*70.8*
MG-Square	60.8	45.2	1.6	**66.2**	*61.6*
MG-Threading	*42.4*	0.2	0.2	24.8	**44.0**

We present results in Table 1, where LIBERO-90 results are aggregated across all action decoders (full results in Table 6)and MimicGen results use DP. We see that Adapt3R does in fact have the modeling capacity to achieve strong results on in-distribution settings, achieving similar performance to baselines in all experiments, and achieving the strongest overall success rate for the MimicGen Threading task.

One takeaway is that Adapt3R has far stronger performance than DP3, which does not include semantic features, indicating that *sparse point clouds without semantic features are not enough to succeed in multitask or high-precision settings*. Another surprising takeaway is that RGBD performs worse than RGB despite having access to more information. This is likely because the wrist camera helps with the spatial reasoning that depth would otherwise help with and the depth is simply extra information that the policy overfits to.

4.3 Unseen Camera Poses

In this section, we investigate whether Adapt3R leads to improved performance when rolling out with unseen camera poses. We train using the original viewpoints from the dataset, which include a scene camera and a wrist camera. At inference time, we rotate the scene camera about the vertical axis through the end-effector starting position as shown in Figure 3. Results are presented in Figure 4, where LIBERO-90 results use BAKU and MimicGen results use DP. We see that Adapt3R consistently gives a substantial boost in performance over comparison observation encoders, especially in settings with very large changes in camera viewpoint, indicating that our scene representation does a good job facilitating this type of generalization. Notably, Adapt3R maintains $> 80\%$ success rate on LIBERO-90, and has the only nonzero MimicGen success rates after ≥ 0.6 radians of rotation.

Table 2: Ablations. In-distribution results, cross embodiment results and camera change results for ablated versions of Adapt3R. We find that the design decisions lead to substantial improvements in generalizability.

Variant	Orig.	UR5e	Kinova3	IIWA	$\theta = 0.4$	$\theta = 1.0$	$\theta = 2.0$	Average
No EECF	83.0	69.3	61.6	61.2	81.6	79.3	78.4	73.5
No EE Crop	91.3	77.8	67.4	70.0	83.4	82.2	82.1	79.2
No Image Features	60.9	48.7	37.2	43.5	2.9	1.9	4.4	28.5
RGB Point Cloud	60.5	48.2	35.7	42.5	3.7	2.1	4.8	28.2
No Lang. Features	91.3	78.7	62.7	64.9	**84.7**	82.2	82.4	78.1
No Positional Encoding	84.3	69.9	61.3	66.9	58.0	70.5	70.9	68.8
Position-Based FPS	91.6	76.8	59.6	64.2	71.7	65.1	70.9	71.4
No Attention	**92.0**	**82.7**	**70.0**	69.5	71.3	64.1	64.9	73.5
Ours	**91.9**	80.1	65.3	**71.8**	**84.7**	**82.9**	**82.9**	**80.0**

DP3 and 3DDA do not do well in this setting. Both of them use calibrated depth cameras to create a point cloud which should intuitively be viewpoint-agnostic. We speculate that this is because the changes in camera position change the distribution of points in the scene and these methods, which condition more on the 3D geometry of the scene, are more brittle to this change.

4.4 Cross Embodiment

In this section, we investigate whether Adapt3R leads to improved performance in a zero-shot cross-embodiment setting, training on the Panda robot and evaluating on the embodiments in Figure 3. Results are shown in Figure 5, where LIBERO-90 is aggregated across all action decoders and MimicGen uses DP. We see that Adapt3R consistently outperforms RGB, RGBD and DP3, especially in the MimicGen experiments where reasoning about scene camera inputs is more important, indicating its superiority in this setting among standalone observation encoders. Adapt3R achieves similar performance to 3DDA in LIBERO and MG Square and superior performance for Coffee and Threading, indicating that point clouds with semantic features are consistently an effective choice for cross-embodiment transfer.

4.5 Ablation Studies

In this section, we ablate several design decisions that contribute to Adapt3R, hoping to understand how these decisions affect its multitask modeling capacity and ability to generalize to novel settings. In addition, we experiment with backbone choice, world frame cropping choice and pooling architecture in the supplement. Table 2 presents ablated versions of Adapt3R when trained together with BAKU on LIBERO-90.

No EECF and **No EE Crop** removes the transformation to the end effector's coordinate frame and cropping in that frame respectively. Results show that both of these lead to improvements across the board as they help the robot reason about the end effector's position relative to objects in the scene.

No Image Features and **RGB Point Cloud** remove image features F and replace them with nothing or the original RGB values respectively. Overall we see that removing this information leads to a substantial drop in performance, especially when generalizing to new camera poses.

No Lang Features does not concatenate the language embeddings $\varphi(\mathcal{L})$. We see that adding them leads to a modest improvement across the board.

No Positional Encoding does not lift the $x \in \mathbb{R}^3$ to $\gamma(x) \in \mathbb{R}^{60}$. We find this change to be important, especially when generalizing to new viewswhere it leads to an average 11.2% improvement.

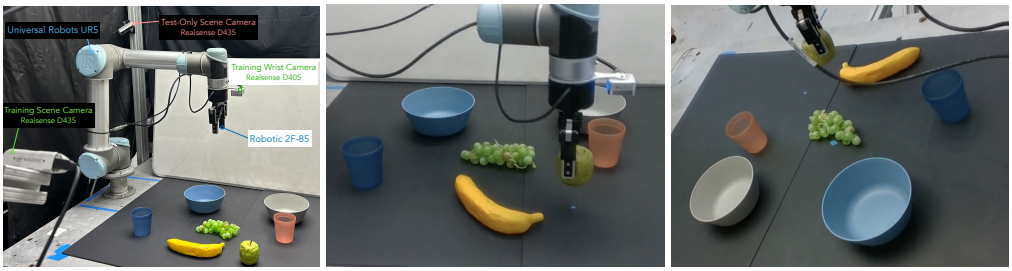
Position-Based FPS downsamples point clouds according to Cartesian coordinates rather than feature vectors. This design choice leads to a respectable benefit across the board, improving success rates by an average of 6.6%. It is especially helpful when transferring to novel camera views.

No Attention uses max pooling instead of attention pooling. Attention pooling is particularly helpful when transferring to new viewpoints, where it gives less attention to out-of-distribution points.

5 Real Experiments

In this section we investigate Adapt3R's applicability to a real-world multitask IL benchmark with 6 tasks, shown in Figure 6, and test whether it shows the same performance with unseen camera poses.

Experiment Setup – For this experiment we compare the Adapt3R backbone to RGB, DP3 and 3D Diffuser Actor (3DDA). We train with data from the scene camera shown in Figure 6b and, for



(a) Real experiment setup **(b)** Training Viewpoint **(c)** Zero-Shot Evaluation Viewpoint

Figure 6: Real-Robot Setup. (a) Illustration of Hardware. (b) The viewpoint used to train all policies. (c) The viewpoint used for our zero-shot evaluation experiments.

our zero-shot experiments, evaluate with the scene camera shown in Figure 6c. For our results we record *progress towards task completion* as opposed to raw success rates. Extended details are in Appendix C.3.

5.1 Results

We present results from the real benchmark in Figure 7. Adapt3R's in-distribution performance is similar to 3DDA and shows an improvement over the RGB and DP3 baselines. DP3's performance was particularly poor, likely because it omits RGB semantic information and uses point locations alone to infer information about the scene which is unreliable with noisy depth estimates from the real sensor. This indicates the importance of semantic features for real-world IL from 3D inputs.

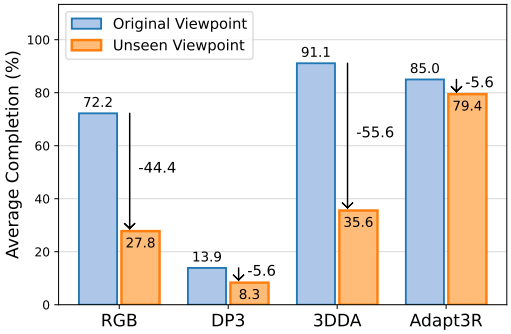


Figure 7: Real Benchmark Results. We see that Adapt3R achieves strong in-distribution performance and retains performance under the change in viewpoint, while baselines do not.

Next, we evaluate policies after replacing the scene camera input with the unseen camera viewpoint shown in Figure 6c. We see that while RGB and 3DDA drop in performance by 44.4% and 55.6% respectively, Adapt3R's performance drops by less than 6%, indicating strong performance with the unseen camera pose. Notably, it achieves a higher average task completion out-of-distribution than RGB and DP3 do in-distribution. Although 3DDA achieves strong in-distribution performance it suffers under changes in camera pose, indicating that its architecture overfits to the training distribution. On the other hand, Adapt3R, whose architecture is explicitly designed to encourage the learner not to excessively condition on the relative locations of the points, does not have this issue and reliably completes the task with the new camera pose.

6 Conclusion

In this paper, we present Adaptive 3D Scene Representation (Adapt3R), a general-purpose observation encoder using 3D scene representations. The key idea behind Adapt3R is to offload the inference of semantic information to a pretrained 2D vision foundation model, and use 3D information specifically to localize those semantic features with respect to the end-effector pose. Experimental results demonstrate that Adapt3R facilitates strong performance in multitask BC setups and high-precision insertion tasks, and leads to substantial improvements for zero-shot transfer to novel viewpoints and embodiments. Experiments in a real-world multitask IL benchmark show that Adapt3R is deployable on a physical robot and lends itself equally well to domain transfer. We hope this work can be a stepping stone towards the application of 3D representations for general autonomy.

7 Limitations

The most important limitation of this work is its reliance on both depth information and camera calibration matrices to facilitate the creation of fused multi-view point clouds, and the transformation to the end-effector's coordinate frame. Collecting this information can be cumbersome, and it is sometimes missing or inaccurate in large-scale datasets [21, 19, 64, 20]. Furthermore, depth cameras fail on objects that are particularly reflective or translucent. We are hopeful that in the future these issues can be resolved by methods using foundation models to extract calibration, like Khazatsky et al. [21] did using DUSt3R [65], and with more accurate learned depth estimators [66].

Another limitation is that the transformation to the end effector's coordinate frame raises questions about the applicability of Adapt3R to bimanual settings. A simple way to handle this would be to simply condition on two Adapt3R vectors, one for each hand or convert to a robot torso frame. In the future we hope to explore 3D representations with less restrictive choices of canonical coordinate frame.

A third limitation of the work is its rather narrow treatment of "cross embodiment". Specifically, we only consider cross embodiment learning between robots with shared action spaces (6D pose prediction), which allows zero-shot transfer to unseen embodiments. In this setting cross embodiment learning is merely a perception problem, but it is unrealistic to assume a shared action space across a large diverse dataset. In the future we'd like to train more complex architectures, perhaps using a separate head for each action space as in [67]. With this being said, we hope this paper's core claim–that Adapt3R helps to close the perception gap between embodiments–is useful for more general cross-embodiment settings.

Acknowledgments

We would like to thank Akshay Krishnan and Jeremy Collins for helpful discussions and for help with writing the paper.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[3] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[4] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[5] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[6] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[7] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[8] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[9] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[10] T. B. Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[11] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[12] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[13] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[14] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.

[15] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710. PMLR, 2023.

[16] H. Ha, P. Florence, and S. Song. Scaling up and distilling down: Language-guided robot skill acquisition. In *Proceedings of the 2023 Conference on Robot Learning*, 2023.

[17] A. Mete, H. Xue, A. Wilcox, Y. Chen, and A. Garg. Quest: Self-supervised skill abstractions for learning continuous control, 2024. URL https://arxiv.org/abs/2407.15840.

[18] S. Haldar, Z. Peng, and L. Pinto. Baku: An efficient transformer for multi-task policy learning, 2024. URL https://arxiv.org/abs/2406.07539.

[19] Q. Vuong, S. Levine, H. R. Walke, K. Pertsch, A. Singh, R. Doshi, C. Xu, J. Luo, L. Tan, D. Shah, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

[20] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*. PMLR, 2023.

[21] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[22] A. Goyal, V. Blukis, J. Xu, Y. Guo, Y.-W. Chao, and D. Fox. Rvt-2: Learning precise manipulation from few demonstrations. *arXiv preprint arXiv:2406.08545*, 2024.

[23] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki. Act3d: 3d feature field transformers for multi-task robotic manipulation. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=-HFJuX1uqs.

[24] Z. Xian, N. Gkanatsios, T. Gervet, T.-W. Ke, and K. Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *7th Annual Conference on Robot Learning*, 2023.

[25] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3d diffusion policy. *arXiv preprint arXiv:2403.03954*, 2024.

[26] Y. Ze, Z. Chen, W. Wang, T. Chen, X. He, Y. Yuan, X. B. Peng, and J. Wu. Generalizable humanoid manipulation with improved 3d diffusion policies, 2024. URL https://arxiv.org/abs/2410.10803.

[27] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *Arxiv*, 2024.

[28] Y. Wang, M. Zhang, Z. Li, T. Kelestemur, K. Driggs-Campbell, J. Wu, L. Fei-Fei, and Y. Li. D^3fields: Dynamic 3d descriptor fields for zero-shot generalizable rearrangement. In *8th Annual Conference on Robot Learning*, 2024.

[29] Y. Wang, G. Yin, B. Huang, T. Kelestemur, J. Wang, and Y. Li. Gendp: 3d semantic fields for category-level generalizable diffusion policy. In *8th Annual Conference on Robot Learning*, 2024.

[30] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[31] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 5628–5635. IEEE, 2018.

[32] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations, 2021. URL https://arxiv.org/abs/2003.06085.

[33] P. Florence, L. Manuelli, and R. Tedrake. Self-supervised correspondence in visuomotor policy learning, 2019. URL https://arxiv.org/abs/1909.06933.

[34] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[35] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[36] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023.

[37] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. M. Shafiullah, and L. Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[38] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[39] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

[40] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[41] J. Wen, Y. Zhu, J. Li, M. Zhu, K. Wu, Z. Xu, R. Cheng, C. Shen, Y. Peng, F. Feng, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2409.12514*, 2024.

[42] J. Wen, M. Zhu, Y. Zhu, Z. Tang, J. Li, Z. Zhou, C. Li, X. Liu, Y. Peng, C. Shen, and F. Feng. Diffusionvla: Scaling robot foundation models via unified diffusion and autoregression. *arXiv preprint arXiv:None*, 2024.

[43] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

[44] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[45] L. Y. Chen, K. Hari, K. Dharmarajan, C. Xu, Q. Vuong, and K. Goldberg. Mirage: Cross-embodiment zero-shot policy transfer with cross-painting, 2024.

[46] L. Y. Chen, C. Xu, K. Dharmarajan, M. Z. Irshad, R. Cheng, K. Keutzer, M. Tomizuka, Q. Vuong, and K. Goldberg. Rovi-aug: Robot and viewpoint augmentation for cross-embodiment robot learning. In *Conference on Robot Learning (CoRL)*, Munich, Germany, 2024.

[47] A. Rashid, S. Sharma, C. M. Kim, J. Kerr, L. Y. Chen, A. Kanazawa, and K. Goldberg. Language embedded radiance fields for zero-shot task-oriented grasping. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=k-Fg8JDQmc.

[48] J. Yu, K. Hari, K. Srinivas, K. El-Refai, A. Rashid, C. M. Kim, J. Kerr, R. Cheng, M. Z. Irshad, A. Balakrishna, T. Kollar, and K. Goldberg. Language-embedded gaussian splats (legs): Incrementally building room-scale representations with a mobile robot, 2024. URL https://arxiv.org/abs/2409.18108.

[49] J. Yu, K. Hari, K. El-Refai, A. Dalil, J. Kerr, C.-M. Kim, R. Cheng, M. Z. Irshad, and K. Goldberg. Persistent object gaussian splat (pogs) for tracking human and robot manipulation of irregularly shaped objects. *ICRA*, 2025.

[50] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[51] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017. URL https://arxiv.org/abs/1706.02413.

[52] C. Wang, H. Fang, H.-S. Fang, and C. Lu. Rise: 3d perception makes real-world robot imitation simple and effective. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2870–2877, 2024. doi:10.1109/IROS58592.2024.10801678.

[53] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[54] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

[55] M. Liu, X. Li, Z. Ling, Y. Li, and H. Su. Frame mining: a free lunch for learning robotic manipulation from 3d point clouds. In *6th Annual Conference on Robot Learning*, 2022.

[56] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of neural networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5301–5310. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/rahaman19a.html.

[57] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[58] Y. Wang, Z. Li, M. Zhang, K. Driggs-Campbell, J. Wu, L. Fei-Fei, and Y. Li. D^3 fields: Dynamic 3d descriptor fields for zero-shot generalizable robotic manipulation. *arXiv preprint arXiv:2309.16118*, 2023.

[59] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[60] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[61] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *7th Annual Conference on Robot Learning*, 2023.

[62] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[63] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies, 2016.

[64] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets, 2021.

[65] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20697–20709, June 2024.

[66] B. Wen, M. Trepte, J. Aribido, J. Kautz, O. Gallo, and S. Birchfield. Foundationstereo: Zero-shot stereo matching. *arXiv*, 2025.

[67] NVIDIA, J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. J. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, J. Jang, Z. Jiang, J. Kautz, K. Kundalia, L. Lao, Z. Li, Z. Lin, K. Lin, G. Liu, E. Llontop, L. Magne, A. Mandlekar, A. Narayan, S. Nasiriany, S. Reed, Y. L. Tan, G. Wang, Z. Wang, J. Wang, Q. Wang, J. Xiang, Y. Xie, Y. Xu, Z. Xu, S. Ye, Z. Yu, A. Zhang, H. Zhang, Y. Zhao, R. Zheng, and Y. Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

[68] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

[69] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017. URL https://arxiv.org/abs/1608.03983.

[70] W. Hu, L. Xiao, and J. Pennington. Provable benefit of orthogonal initialization in optimizing deep linear networks, 2020. URL https://arxiv.org/abs/2001.05992.

[71] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[72] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arxiv:2006.11239*, 2020.

[73] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

[74] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, Y. Zhu, and K. Lin. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

A Comparison to Similar Methods

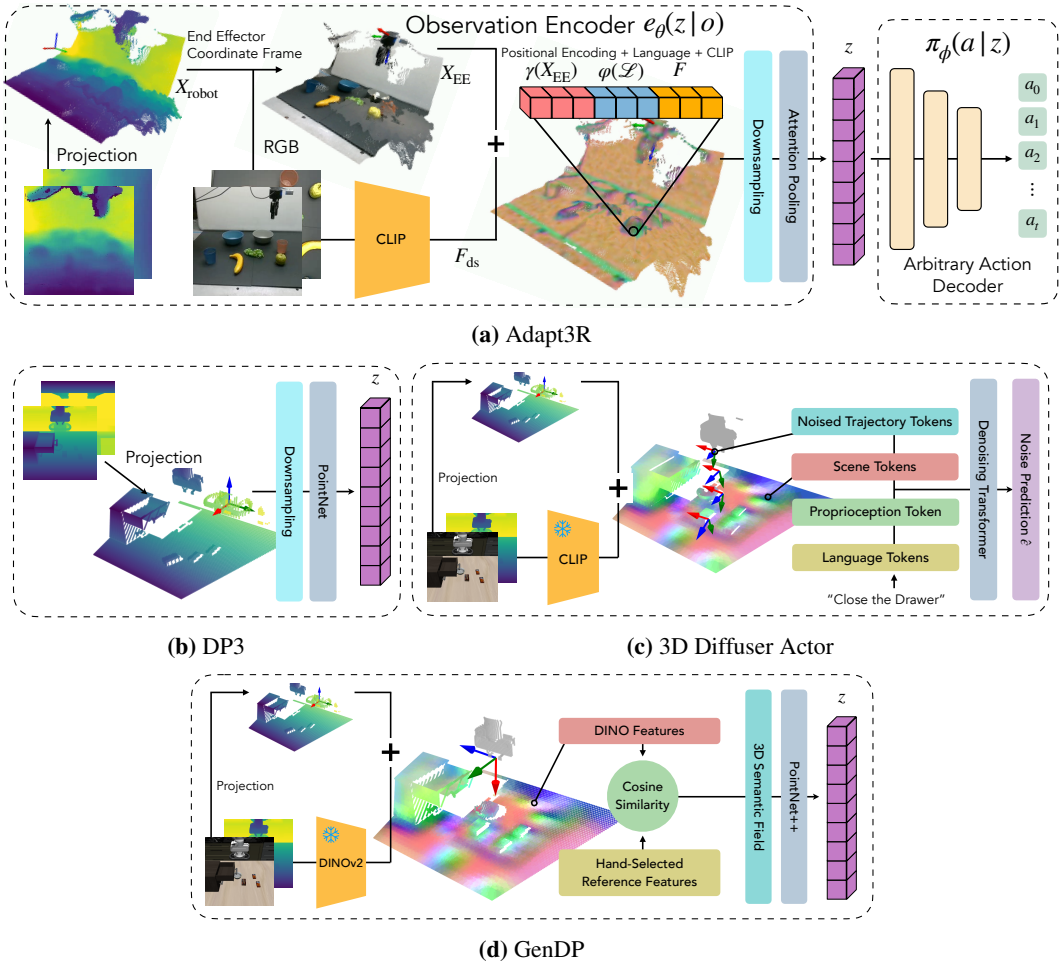


Figure 8: In this figure we compare Adapt3R to several recent methods which also use point clouds for imitation learning. (a) We provide a diagram of Adapt3R for reference. (b) DP3 [25] omits any semantic information, instead conditioning on colorless point clouds. (c) 3D Diffuser-Actor [27] cross attends between noisy action trajectories and scene tokens. (d) GenDP [29] hand selects important reference features in the training data, and constructs semantic fields through cosine similarity between reference features and scene features.

In Figure 8 we present a comparison to several recent imitation learning algorithms which also use point clouds as a means to achieve some degree of generalization.

3D Diffusion Policy (DP3) [25, 26] operates based on the core idea that a colorless point cloud contains sufficient information to complete a variety of tasks, and shows impressive results generalizing to new object instances, camera poses and scenes. It lifts the depth maps into colorless point clouds followed by cropping, some amount of downsampling and a neural network to process the points before a max pooling operation.

As we show in our experiments in Sections 4, these methods fail in multitask and high-precision settings, where reasoning about semantic information regarding the items in the scene is much more important. In multitask settings the semantic information helps the agent to identify its target object, and in high-precision settings the sparse point cloud is insufficient since it does not contain the granularity to discern whether the objects are properly aligned to facilitate the insertion. Adapt3R differs from these methods not only in its use of semantic information from the CLIP backbone, but also in its point cloud processing steps such as the positional encoding γ and the use of the feature space rather than Cartesian coordinates as the metric space for farthest point sampling.

Furthermore, as we show in Section 5, these methods are particularly ill-fit to settings with noisy point cloud inputs, such as those arising from stereo depth matching. While Ze et al. [25, 26] use a LiDAR camera for their experiments, we argue that the assumption of access to high-quality point clouds is limiting, especially since some large-scale data collection efforts only provide stereo depth (eg DROID [21]). By incorporating semantic information in the point cloud, Adapt3R relies less on accurate point clouds to infer information about the scene and is able to make better sense of these noisy point clouds.

3D Diffuser Actor (3DDA) [27] is another similar method designed to use 3D scene representations to learn a diffusion policy. Specifically, it trains a transformer to, conditioned on the observation and a noisy action a_t, predict noise ϵ which can be removed from a_t to lead it closer to the original action a_0. Note that the actions a_t are absolute pose actions.

Like Adapt3R, it lifts semantic features from a frozen CLIP model into a point cloud. It also lifts the noisy action poses into the cloud, and embeds the proprioceptive information and language instruction into more tokens. Finally, it self attends between points in the scene and cross attends between scene points, action position samples, language tokens and proprioception.

A core part of this algorithm is that it extracts information from the scene through a self-attention operation on the points in the scene and the noisy trajectory candidate. This has several disadvantages. For one this operation is slow, with time complexity growing quadratically with the number of points in the scene. Furthermore, since this step requires an up-to-date diffusion trajectory candidate, it must be performed again for each diffusion iteration, drastically slowing inference. Next, we believe it is a likely cause of 3DDA's failure to generalize well to unseen camera poses. Because this operation conditions on the 3D geometry of the scene when it self-attends between the points in the scene, it is susceptible to overfitting to that geometry, leading to issues when generalizing to new camera viewpoints.

GenDP [29] is a recent method which extends D^3Fields [28] to general manipulation policies. The key idea is to extract DINOv2 [4] features from the images, which are lifted into 3D. It uses the descriptor field algorithm from Wang et al. [28] to combine features from several viewpoints. Then, it computes cosine similarity between these features and some hand-selected task-relevant reference features (such as shoelaces on a shoe) to compute a semantic feature map. It concatenates these features with Cartesian coordinates to form final representations which it processes with a PointNet++ [51] into a final conditioning vector for a diffusion policy.

Beyond some of the point cloud processing details discussed in Sections 3.1 and 3.2, the key difference between GenDP and Adapt3R is GenDP's dependence on hand-selected reference features in their construction of the semantic feature map. While this may be acceptable in their setting, where they set out to achieve object instance generalization for specific tasks, this is not scalable to a multitask setting, where it would be necessary to define such reference features for each tasks. Adapt3R directly uses the outputs of its RGB backbone without need for task-specific reference features.

B Extended Method Details

Here we present detailed information about the methods behind the experiments, including detailed information about the inputs to the policies, hyperparameter choices, and action decoder implementation details.

For all algorithms, $\ell_i \in \mathbb{R}^{d_e}$ is the projected output from a pre-trained frozen language encoder (CLIP), $u_{i,t} \in \mathbb{R}^{d_e}$ is the output of a learned proprioception encoding network U_θ and $z_{i,t} \sim e_\theta(z_{i,t}|o_{i,t})$ is the output of the Adapt3R encoder. We assume the actions $a_{i,t}$ are captured from an expert demonstrator that consistently completes the task and define $A_{i,t} = \{a_{i,t}, \ldots, a_{i,t+H-1}\}$ to be the length H chunk of actions starting from timestep i.

All policies have access to two cameras, as described in Appendix C. All policies have access to proprioceptive information including end effector position and gripper state. All the baseline methods concatenate it and pass it through an MLP encoder, while Adapt3R uses it to transform the point cloud into the end effector's coordinate frame.

We use the following hyperparameters for all experiments:

Parameter	Value
Optimizer	Adam [68]
Learning Rate	1×10^{-4}
Weight Decay	1×10^{-4}
Scheduler	Cosine Annealing [69]
Batch Size	64
Grad Clip	100
# Train Epochs	100
Embed Dimension d_e	256
# Downsample Points p	512
Weight Initialization	Orthogonal [70]

Table 3: Shared hyperparameters across all experiments

We found it necessary to change some hyperparameters to accommodate the high precision requirements of the MimicGen environments, and the parameters we tune are presented in Table 4. In particular, we found finetuning the CLIP backbone to be helpful in reasoning about small position changes needed for the MimicGen experiments. Also we found temporal aggregation, which smoothes predicted trajectories by averaging over several past predictions for the next timestep, to be detrimental in this setting where highly precise motions are required.

Parameter	LIBERO-90	MG-Coffee	MG-Square	MG-Threading	Real
Temporal Aggregation	True	False	False	False	True
Resample Frequency	N/A	6	2	2	N/A
Finetune CLIP	False	False	True	True	False

Table 4: Changing hyperparameters across experiments

B.1 Action Decoder Detail

Here we summarize the action decoders used in the paper.

Action chunking transformer (ACT) [36] learns to predict chunks of actions by casting the learning as a conditional variational inference problem, and sampling from the likelihood distribution which is parameterized by a transformer model [71]. Specifically, it learns an encoder $q_\phi(\eta|A_{i,t}, z_{i,t}, u_{i,t})$, where η is a latent variable, and decoder $\pi_\phi\left(\hat{A}_{i,t}|z_{i,t}, u_{i,t}, \ell, \eta\right)$ and optimizes the following variational objective:

$$\mathscr{L}(\phi) = \text{MSE}\left(A_{i,t}, \pi_\phi(z_{i,t}, u_{i,t}, \ell, q_\phi(A_{i,t}, z_{i,t}, u_{i,t}))\right) + \beta D_{\text{KL}}(q_\phi(\eta|z_{i,t}, u_{i,t})||\mathcal{N}(0,1)). \quad (1)$$

In Zhao et al. [36]'s implementation, the RGB input is a sequence of patch embeddings. For our implementation we replace this with the output of either the Adapt3R perception backbone or the backbones described in Section 4.1, which we found to simplify the pipeline and still achieve very strong performance. Additionally, for Zhao et al. [36]'s implementation the VAE encoder only conditions on proprioceptive information, since it would be too inefficient to also condition on all of the perception outputs. Since our version has far fewer tokens to condition on, the encoder conditions on them without efficiency issues.

For the KL weight we found $\beta = 10$ to be good. For baselines we found that a chunk size of $H = 15$ worked well and for Adapt3R we got best results with $H = 10$.

Diffusion policy (DP) [35] samples chunks of actions through a learned reverse Langevin dynamics process from [72]. Specifically, we sample $k \sim \text{Uniform}(\{1, 2, \ldots, K\}), \epsilon \sim \mathcal{N}(0, I)$, and minimize

$$\mathscr{L}(\phi) = \text{MSE}(\epsilon, \epsilon_\phi(A_{i,t} + \sigma_t \epsilon, t, z_{i,t}, u_{i,t}, \ell_i)) \tag{2}$$

where σ_k is a noise scale according to the noise schedule and t is the timestep in the noising MDP.

There are several design choices to make in designing a diffusion policy. We used the CNN UNet version of the model with delta pose actions. For conditioning, we concatenate perception, proprioception and language embeddings, which we project down to d_e and use as global conditioning.

We use the DDIM [73] scheduler with 100 training steps and 10 inference steps and a `squaredcos_cap_v2` beta schedule. For Adapt3R we found a chunk size of $H = 8$ to work well, while a chunk size of $H = 16$ worked well for baselines.

BAKU [18] passes a sequence containing task conditioning ℓ, observation encodings $z_{i,t}$, proprioception encodings $u_{i,t}$ and a learned readout token to a transformer decoder network T_ϕ [71]. It uses the last output from the transformer as input to an MLP action head π_ϕ, which outputs a Gaussian distribution over chunks of actions. The final pipeline is trained with the following NLL objective:

$$\mathscr{L}(\phi) = -\log \pi_\phi(A_{i,t} | T_\phi(\ell, z_{i,t}, u_{i,t})[-1]). \tag{3}$$

With the exception of chunk size, we use the default hyperparameters from Haldar et al. [18]. For chunk size, we use a value of $H = 10$ and $H = 15$ for the baselines and Adapt3R respectively.

B.1.1 Action Decoder Choice

You'll notice that we use different action decoders for different experiments. For example, we use Baku for the ablation experiment and DP for all the MimicGen experiments. However, within each experiment, **we always use the same action decoder for Adapt3R and baselines**, meaning there is **always a 1:1 comparison**. Different decoders have different properties. BAKU trained quickly, so we used it for the expensive ablation study (BAKU alone $\approx$ 2000 GPU hours). We only used BAKU for the LIBERO camera viewpoint sweep since this was also expensive ($\approx$ 900 GPU hours), but aggregated across all decoders for the in-distribution and cross embodiment (XE) experiments because these were cheaper. In Table 9 we present viewpoint change experiments for all decoders with similar results. We chose DP for MG experiments because it better handled precise tasks.

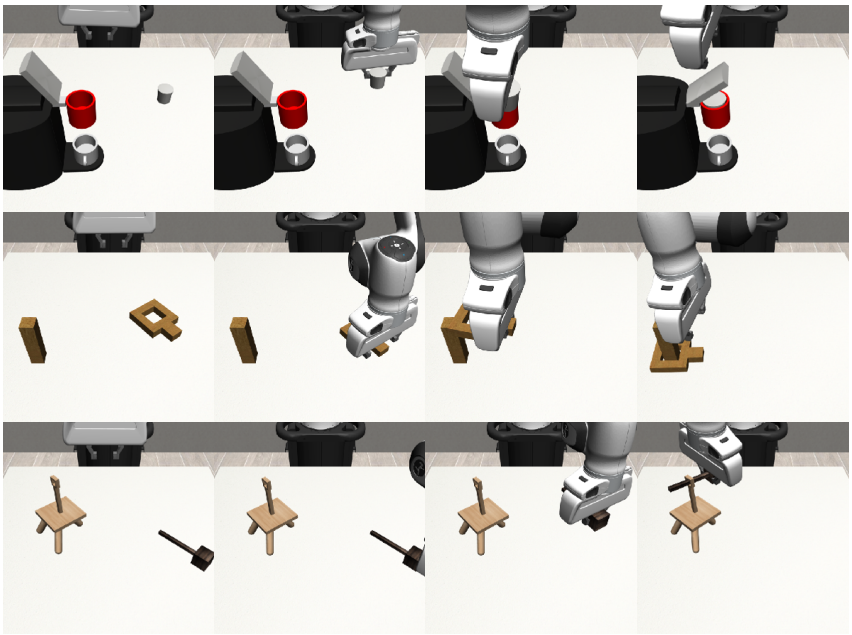


Figure 9: We experiment with the MimicGen Coffee (top), Square (middle) and Threading (bottom) environments, which all require high-precision insertions. For each task we use the D1 reset distribution which requires the policy to learn to complete the task from a wide range of starting configurations.

C Experiment Details

In this section we provide further details and visualizations describing the experiments described in Section 4

- **LIBERO** [60] is a multitask learning benchmark consisting of several task suites designed to study lifelong learning. We evaluate on LIBERO-90, which includes 90 rigid-body and articulated manipulation tasks with corresponding natural language instructions. All algorithms have access to a scene camera, wrist camera, and proprioceptive information, and there are 50 demonstrations per task, for a total of 4500 demonstrations including about 670k state-action pairs.
- **MimicGen** [61] is a data generation framework that creates large-scale imitation learning datasets from a small number of human demonstrations by segmenting them into object-centric subtasks and generating new trajectories by transforming and executing these segments in novel scenes. It offers a suite of 18 manipulations tasks spanning high-precision, long-horizon, and contact-rich settings. We focus our evaluations on three tasks: **Coffee**, **Square**, and **Threading**.

C.1 LIBERO Environment Details

Most of our experiments use the LIBERO-90 benchmark [60]. The benchmark consists of 90 tasks and corresponding language instructions, and is designed to evaluate agents' lifelong learning capabilities. The tasks are spread across 20 scenes in 3 settings (Kitchen, Living Room, Study).

For all experiments, the action space is delta poses in the robot base frame, and the robots are controlled by the Robosuite OSC controller [74].

For the **change of robot experiment**, we replaced the Franka Panda with the robots shown in Figure 3. Since the robots are controlled by delta pose actions, they share an action space and we could run the outputs from the policy with no further processing. Since we use the gripper joint states as proprioception input, we used the Franka Panda end effector for all robots. In addition to the robots in Figure 3 we tried the Sawyer robot, but found it to have low success rates ($< 15\%$) for all algorithms.

The LIBERO benchmark comes with 50 initialization states for each task, which specify the poses of the robots and all objects in the scene for deterministic evaluation. For the change of robot experiment, we process these such that all objects in the scene have the original pose, and the end effector of the

Task No.	Language Instruction	# Trajectories	# Samples
1	"Pick up the apple and place it in the grey bowl"	87	19209
2	"Pick up the apple and place it in the blue cup"	80	15925
3	"Pick up the grapes and place them in the blue bowl"	59	10339
4	"Pick up the banana and place it in the grey bowl"	74	14635
5	"Pick up the grey bowl and stack it in the blue bowl"	60	10775
6	"Pick up the blue cup and stack it in the pink cup"	77	16583
	Total	437	87466

Table 5: Statistics about our real dataset.

new robot has the same pose as the original end effector. We found that without this change (ie using the end effector's default initialization pose) all policies had very low success rates, presenting a compelling challenge for future work.

For the **change of camera experiments** we simply moved the camera and updated the camera calibration matrices. For the results in the main paper we rotate the camera about the vertical axis going through the end-effector starting position, as described in Section 4.3. For the tables in the appendix, we consider `small`, `medium` and `large` changes in camera pose. Specifically, for the `small` change, we shift the camera by $(0.0, +0.3, -0.1)$ m, for the `medium` change, we shift the camera by $(-0.2, +0.7, -0.2)$ m and for the `large` change, we shift the camera by $(-1.2, +1.0, -0.2)$ m. For all camera position changes we subsequently rotate the camera such that it is facing the end effector's starting location.

C.2 MimicGen Environment Details

For the MimicGen experiments we use the provided D1 datasets, each of which includes 1000 trajectories procedurally generated from 10 human trajectories. We choose the Coffee, Square and Threading tasks because they are difficult enough to showcase Adapt3R's applicability to difficult high-precision insertion tasks.

We use the same change of robot and change of camera pose experiments from LIBERO. We generate initial states for each environment that we use to ensure fair evaluation.

C.3 Real Experiment Details

The real benchmark includes 6 pick-and-place tasks with between 59 and 87 demonstrations per task for an average of 73 demonstrations per task, collected using a Meta Quest 3 headset for teleoperation. A sample of tasks are shown in Figure 1 and the full suite is shown in the supplement. We use a Universal Robots UR5 and a Robotiq 2F-85 gripper. The setup includes two Realsense D435 scene cameras and a Realsense D405 wrist camera. A diagram of the setup is shown in Figure 6a.

For all observation backbones we train diffusion policies with absolute end-effector pose action space. All policies are trained in a multitask fashion using CLIP-encoded language instructions to differentiate tasks.

For our results we record *progress towards task completion* as opposed to raw success rates, where grasping the target object corresponds to $\frac{1}{3}$ progress, bringing it into some contact with the goal object corresponds to $\frac{2}{3}$ and completing the task is $\frac{3}{3}$. We train two random seeds per policy and perform 5 evaluation trials per seed for a total of 10 trials per task per algorithm.

Our dataset includes a total of 437 trajectories for 6 tasks. In Table 5 we present exact statistics about the datasets for the 6 tasks. In Figure 10 we show visualizations of all tasks present in our multitask imitation learning benchmark.

For all policies we use end effector pose as proprioceptive input. For all objects we select a home location and initialize them within a 10 cm radius of that home location. For objects which are not axisymmetric about the vertical axis (eg. the banana and grapes) we randomly rotate them. At evaluation time we deterministically initialize object poses in a grid near the home location.

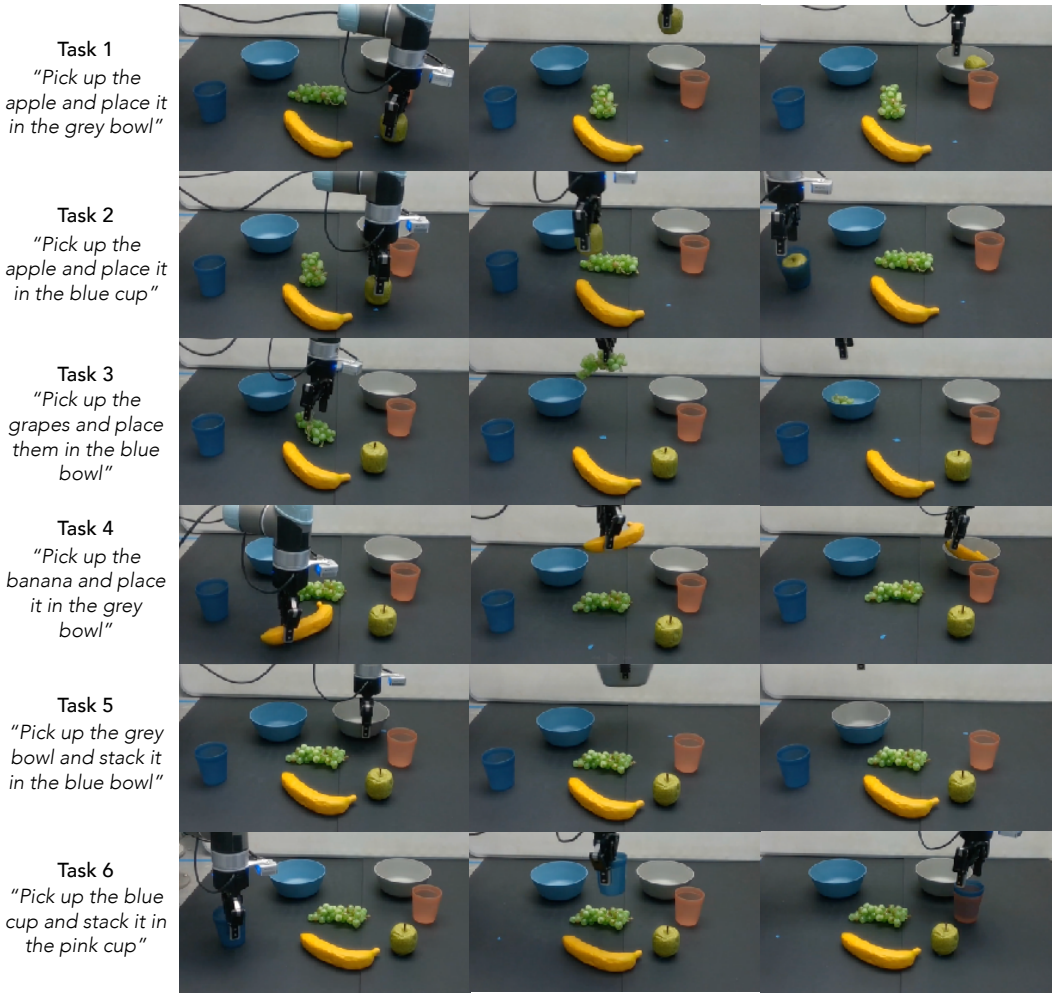


Figure 10: Our real-world multitask imitation learning benchmark includes 6 pick-and-place tasks shown above.

C.4 Action Space Coordinate Frame

Actions are in the robot base frame (eg the default for Robosuite OSC controller). We looked into transforming them into the end effector coordinate frame and did not see substantial benefits from this. This may be an inductive bias that makes sense to us but is not necessary in practice for the domains we consider. That being said, choice of action spaces is an interesting direction for future work.

D Further Results and Visualizations

In this section we present additional results that did not fit in the main paper. For all experiments we consider the LIBERO-90 benchmark.

D.1 Full LIBERO Results

Here we present tables with full numerical results that were summarized in the bar charts in the paper.

Table 6 shows the in-distribution results for all three action decoders on the LIBERO-90 dataset. We see that Adapt3R achieves strong multitask performance in this setting and combines effectively with each action decoder.

Table 6: Success rates for in-distribution evaluations on the LIBERO 90 benchmark. We see that Adapt3R has a similar modeling capacity to strong baselines such as RGB and 3DDA for LIBERO 90.

Encoder	ACT	Diffusion Policy	BAKU
RGB	**0.908**	**0.905**	0.914
RGBD	0.705	0.868	0.789
DP3	0.753	0.687	0.681
3D Diffuser Actor	-	0.837	-
Adapt3R	0.903	0.880	**0.919**

Table 7 presents full results for the cross embodiment experiment on LIBERO-90. We see that Adapt3R combines will with ACT and BAKU to achieve strong cross embodiment results across the borad for these algorithms. When combined with DP, it achieves strong performance when switching to the UR5e embodiment, but is outperformed by 3DDA, which also uses a 3D scene representation that is more robust to changes in embodiment.

Table 7: Success rates when rolling out policies in a zero-shot cross-embodiment setting. We see that Adapt3R consistently outperforms all other observation encoders and achieves similar performance to 3DDA.

Algorithm	UR5e	Kinova3	IIWA
ACT + RGB	0.578	0.643	0.525
ACT + RGBD	0.493	0.505	0.420
ACT + DP3	0.592	0.574	0.613
ACT + Adapt3R	**0.777**	**0.743**	**0.714**
DP + RGB	0.543	0.562	0.415
DP + RGBD	0.464	0.441	0.332
DP + DP3	0.563	0.404	0.423
3D Diffuser Actor	0.722	**0.759**	**0.651**
DP + Adapt3R	**0.757**	0.529	0.505
BAKU + RGB	0.442	0.458	0.352
BAKU + RGBD	0.419	0.356	0.310
BAKU + DP3	0.579	0.468	0.511
BAKU + Adapt3R	**0.801**	**0.653**	**0.718**

In Table 8 we present results for all algorithms under a zero-shot camera change experiment, with camera viewpoints described in Section C.1. Note that these angles are different than those presented in the main body of the paper. We see that, across all action decoders, Adapt3R enables strong performance with unseen camera poses.

D.2 Inference Speed

In Table 9 we present inference speeds for Adapt3R and the main comparisons. These numbers reflect the rate at which a single observation can be repeatedly passed through the pipeline to output chunks of actions on an

Table 9: Inference rates (Hz) for Adapt3R and baselines.

RGB	DP3	3DDA	Adapt3R
132.2	146.1	2.6	44.1

Table 8: Success rates when rolling out policies in a zero-shot camera change setting for LIBERO-90. We see that Adapt3R consistently outperforms all comparisons and achieves success rates an average of 56.1% higher than the next best comparison.

Algorithm	Small	Medium	Large
ACT + RGB	0.310	0.216	0.230
ACT + RGBD	0.323	0.220	0.175
ACT + DP3	0.376	0.218	0.081
ACT + Adapt3R	**0.787**	**0.706**	**0.749**
DP + RGB	0.133	0.058	0.042
DP + RGBD	0.178	0.075	0.080
DP + DP3	0.179	0.174	0.028
3D Diffuser Actor	0.226	0.208	0.186
DP + Adapt3R	**0.740**	**0.798**	**0.791**
BAKU + RGB	0.141	0.061	0.043
BAKU + RGBD	0.176	0.065	0.068
BAKU + DP3	0.311	0.261	0.135
BAKU + Adapt3R	**0.847**	**0.829**	**0.829**

RTX 3090 using BAKU as the action decoder (with the exception of 3DDA which diffuses). In other words, chunks per second. We observe that while RGB and DP3 run at high speed, they perform poorly, and in contrast 3DDA performs well but at an untenable 2.6 Hz. Adapt3R provides high performance at a 44.1 Hz, which is faster than the input rate of prevalent RGBD cameras (∼30 Hz).

D.3 Failure Case Study

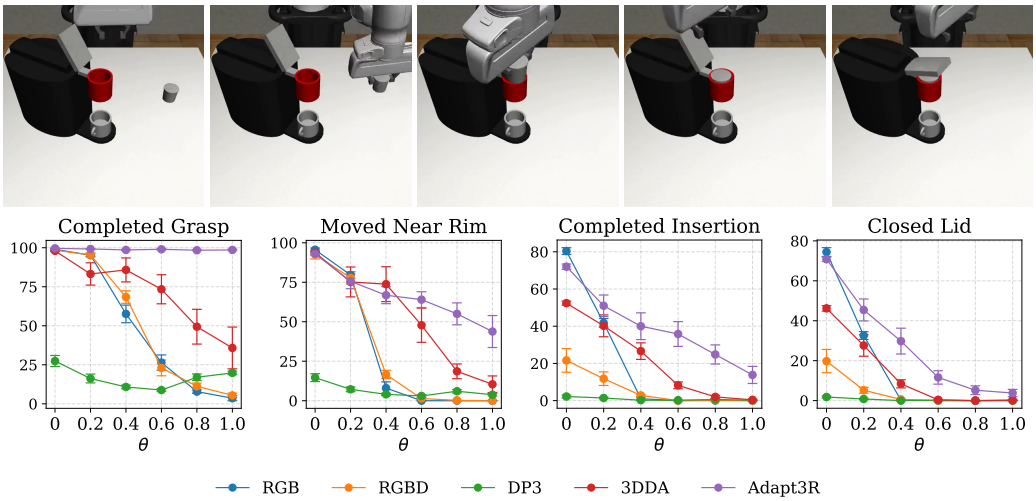


Figure 11: (Top) We visualize the stages of the coffee task whose respective success rates are presented in the bottom row. From left to right, `start`, `grasp`, `rim`, `insertion`, `task`. **(Bottom)** We plot the subtask success rate for each stage of the coffee task (eg what percentage of trials made it to a particular stage). We see that even with a very large change in camera pose Adapt3R consistently completes the grasp and is more successful at localizing the rim and completing the insertion than comparison algorithms.

Here we study failure cases from the MimicGen Coffee environment (MG Coffee) to help understand Adapt3R's strengths and limitations in more detail with results in Figure 11. We see that even with very large changes in camera pose Adapt3R consistently (over 99%) completes the grasp, indicating that it is very effective at localizing the target object even in out of distribution settings. Furthermore, it outperforms all other baselines for all the other subtasks. While it does relatively well for the `grasp` and `rim` subtasks, it struggles with the insertion subtask. This is because this subtask requires more visual servoing which is less robust to changes in camera pose.

D.4 Real

In Table 10 and Table 11 we present results for all tasks in our real-world multitask benchmark. We see that in the original domain, Adapt3R achieves the strongest overall performance in two of the six tasks. When transfering to the unseen camera pose, Adapt3R achieves the strongest performance by far and is the only algorithm to consistently complete the tasks.

Table 10: Average task completion for each task in our benchmark when using the original camera viewpoint.

Algo.	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Avg.
RGB	**100**	90.0	43.3	50.0	50.0	**100**	72.2
DP3	0.0	0.0	53.3	3.3	26.7	0.0	13.9
3DDA	**100**	**100**	90.0	**70.0**	**86.7**	**100**	**91.1**
Adapt3R	**100**	90.0	**93.3**	63.3	80.0	83.3	85.0

Table 11: Average task completion for each task in our benchmark when rolling out with the unseen camera viewpoint shown in Figure 6c.

Algo.	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Avg.
RGB	30.0	16.7	50.0	0.0	33.3	36.7	27.8
DP3	0.0	10.0	23.3	0.0	16.7	0.0	8.3
3DDA	80.0	26.7	50.0	20.0	33.3	3.3	35.6
Adapt3R	**100**	**83.3**	**90.0**	**53.3**	**73.3**	**76.7**	**79.4**

D.5 Attention Maps

One unique attribute from our architecture is the use of attention pooling, which allows us to visualize attention maps and understand which parts of the scene our policy attends to at which times, as we do in Figure 12.

For the Coffee task (top) we see that early on it attends primarily to the coffee pod, which it is trying to grasp. Later on it attends to points around the base of the coffee machine, which it uses to localize itself with respect to the machine for the precise insertion. For the Square task (middle) it begins by attending primarily to the object it must grasp before later attending to the peg. For the Threading task (bottom), it once again starts by attending to the object it must grasp, and then attends to the tripod to help with aligning for the insertion.

D.6 Architecture Study

In this section we study design choices in the final part of our pipeline, which reduces the point cloud constructed in Section 3.1 into a single vector for use as conditioning for the action decoder. Overall we conclude that this design choice is not as important as the design choices discussed earlier in the paper.

Our pipeline uses an attention pooling reduction as described in Section 3.2. We compare to the following architectures:

- The architecture from **DP3** [25], which runs each point through an MLP before performing max pooling across the point cloud. This method is similar to ours, but replacing the attention pooling with max pooling.
- The architecture from **GenDP** [29], which uses PointNet++ [51], a hierarchical point processing architecture.

We present results from this comparison in Table 12. Overall, we see that although Adapt3R narrowly outperforms the comparison architectures, the results do not resoundingly indicate any one architecture is best. This indicates to us that for future work it may be more fruitful to study changes to the structure of the point cloud itself rather than the final embedding extractor architecture.

Figure 12: We study attention maps from the MimicGen environments at various critical points in the rollouts. In order to create attention maps at full resolution we bilinearly interpolate feature maps to the size of the full point cloud, which causes them to be slightly fuzzy around the edges of the objects.

Table 12: How do choices in the embedding extractor architecture affect the final success rate? In this table we present success rates after varying the architecture as discussed in Section D.6. While Adapt3R has narrowly better overall performance, the results are inconclusive. All experiments are from combining Adapt3R with BAKU.

	DP3	iDP3	GenDP	Ours
Orig.	**0.920**	0.881	0.886	0.919
UR5e	**0.827**	0.758	0.735	0.801
Kinova3	**0.700**	0.611	0.619	0.653
IIWA	0.695	0.660	0.697	**0.718**
Small	0.713	0.747	**0.859**	0.847
Medium	0.641	0.781	0.792	**0.829**
Large	0.649	0.747	0.781	**0.829**

D.6.1 RGB Backbone Choice

In this section we study the choice of RGB backbone used to define f in Section 3.1. In the final version of Adapt3R we use a CLIP [53] backbone with frozen weights, similarly with [27]. However, it is unclear whether pretrained CLIP weights are indeed the best choice. To that end, in Table 13 we compare CLIP weights with pretrained ResNet weights [62], and ablate the finetuning. We also explore the use of DINOv2 [4] in Table 14, which is known to extract richer local features important for robotic manipulation tasks [29, 28].

First of all, we see no clear difference between performance with CLIP weights and ResNet18 weights. This is surprising, and indicates to us that the model is likely not reasoning deeply about the semantic information in F, but rather is using the feature vectors to look up important points.

Another interesting takeaway from Table 13 is that finetuning does not help with in-distribution settings but is harmful to out-of-distribution settings, especially when transferring to novel camera viewpoints. One explanation is that finetuning leads the backbone to overfit to the dataset and fail more often for slightly OOD states in in-distribution settings and fail frequently for OOD evaluation settings.

Table 13: How does the choice of image feature extractor backbone affect the performance of Adapt3R? 'FT' means the model was finetuned while 'Fr' means its weights are frozen. 'RN18' means a ResNet-18 pretrained with ImageNet weights [62, 1], and CLIP refers to CLIP [53]. All experiments are from the combination of Adapt3R and BAKU.

	FT RN18	Fr RN18	FT CLIP	Fr CLIP
Orig.	0.909	0.906	0.912	**0.919**
UR5e	0.774	0.758	0.774	**0.801**
Kinova3	0.597	0.615	0.611	**0.653**
IIWA	0.674	0.704	0.654	**0.718**
Small	0.840	**0.848**	0.819	0.847
Medium	0.819	**0.840**	0.806	0.829
Large	0.807	0.822	0.780	**0.829**

Finally, in Table 14, we explore the potential of DINOv2 [4] as a richer feature extractor. DINOv2 is known for capturing detailed local features, as a result of its self-supervised pretraining task, which emphasizes learning spatially localized representations. This ability could enhance the model's performance at identifying fine-grained details critical for robotic manipulation tasks. For our experiments, we use the `dinov2_small` model, which outputs a 384-dimensional feature vector for each patch. We modify the stride in the patch extractors and pad the input images to ensure the feature volume output from DINOv2 is compatible with the Adapt3R architecture.

We investigate the usefulness of representations from different layers of DINOv2 by extracting features from various depths:

- **Patch Embeddings**: The embeddings from the patch projection before being passed into the DINOv2 backbone.
- **Layers** n: The feature vectors output by the DINOv2 backbone after being sequentially processed through the first n transformer blocks.

In all cases, we discard the class token (`[CLS]`) to focus on the local features. We avoid experimenting with deeper layers or finetuning as we found DINOv2 feature extraction to be prohibitively time consuming and processing beyond the third layer becomes impractical for both training time and inference speed.

In the original setting, there is no noticeable difference in performance when using DINOv2 features compared to CLIP or ResNet. This consistency across RGB backbones reinforces our earlier claim that the model primarily uses the features to identify task-relevant points rather than leveraging their semantic richness.

In the **change of robot experiments**, DINOv2 features show a slight advantage, with a 2-3% improvement, particularity in the shallowest layer (Layer 1). This may suggest that the local features extracted by DINOv2's shallow layers are less sensitive to the agent's embodiment and more focused on the objects and interactions in the scene, enabling better generalization to new robot embodiments.

In the **change of camera experiments** DINOv2 under-performed compared to CLIP or ResNet, likely because the shallow layers in DINOv2 focus on local details and fail to capture the broader spatial context needed for viewpoint changes. CLIP, on the other hand, is better at capturing global information, making it more effective for scenarios involving significant camera shifts. Notably, performance improves when using features from deeper DINOv2 layers, which are known to encode information over larger spatial extents [4]. This highlights the importance of global representations for handling viewpoint changes, as they allow the model to reason about the overall structure of the scene rather than relying solely on local features.

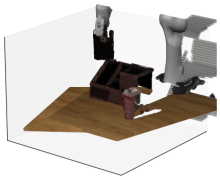
D.6.2 Cropping

An important detail in several recent works is cropping, or the lack thereof [27, 25, 26]. In this experiment, we set out to understand how different cropping schemes impact the success rate of the final policy. Specifically, we compare the cropping schemes in Figure 13, including:

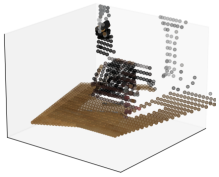
Table 14: Effect of using different layers in DINOv2 on the performance of Adapt3R. 'Patch' refers to the initial patch embeddings before the DINOv2 backbone, while 'L1', 'L2', and 'L3' denote features extracted after the first, second, and third transformer block, respectively. 'Ours' corresponds to the baseline model, Adapt3R using the CLIP backbone with frozen weights. All experiments are from the combination of Adapt3R and BAKU. Note that the versions with DINO features are from an older version of our pipeline which didn't use proprioception, which explains their marked deficiency for new camera poses.

	Patch	L1	L2	L3	Ours
Orig.	0.909	**0.937**	0.929	0.923	0.919
UR5e	0.809	**0.840**	0.818	0.820	0.801
Kinova3	0.714	0.766	**0.774**	0.713	0.653
IIWA	0.662	0.716	0.697	0.711	**0.718**
Small	0.442	0.549	0.543	0.591	**0.847**
Medium	0.208	0.426	0.513	0.585	**0.829**
Large	0.167	0.437	0.558	0.654	**0.829**

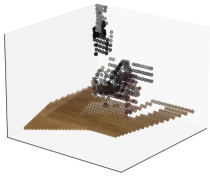
(a) RGB Image



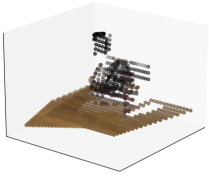
(b) Uncropped



(c) Loose cropping



(d) Tight Cropping



(e) EE cropping

Figure 13: In this figure we present visualizations of the various cropping schemes discussed earlier in the paper. (a) The original RGB for reference. (b) The uncropped point cloud. The background is visible behind the robot. (c) Loose cropping. The robot is visible but the background is not. (d) Tight cropping. The table is fully visible but the robot base is not. (e) Tight cropping + hand cropping. We use the robot's proprioception to remove all points behind the end effector.

Table 15: The effect of world frame cropping on Adapt3R. 'None' means no cropping, 'Loose' means that the background is cropped out but objects near the table, such as the robot, are not cropped out (see Figure 13c). 'Tight' means that nearer objects such as the robot base are cropped out. 'No EE' omits the end effector cropping discussed in Section 3.2, and all other experiments use it. All experiments are from combining Adapt3R with BAKU.

	None	Loose	Tight, No EE	Tight
Orig.	0.917	0.914	0.913	**0.919**
UR5e	0.772	0.790	0.778	**0.801**
Kinova3	0.648	0.641	**0.674**	0.653
IIWA	0.695	0.715	0.700	**0.718**
Small	**0.858**	0.849	0.834	0.847
Medium	0.830	**0.830**	0.822	0.829
Large	0.802	0.813	0.821	**0.829**

- No cropping
- Loose cropping, which removes the faraway background but not nearby objects such as the robot base.
- Tight cropping, which removes nearer objects such as the robot base but does not remove the table.
- EE cropping, which crops out points behind the end effector.

Results are presented in Table 15. We see that cropping leads to a modest improvement in in-distribution performance. Surprisingly it does not lead to a substantial improvement for the change of robot experiments, with the exception that end effector cropping leads to a modest improvement. We see that cropping does lead to a noticeable improvement for the camera change experiments. This makes sense since these experiments introduce many out of distribution faraway points that are subsequently cropped out.

D.6.3 Downsampling

In this section we study the downsampling strategies discussed in Section 3.2. Specifically, we compare the position-based downsampling used by DP3 [25] and iDP3 [26] to downsampling based on image features in F, as in [27].

First, we compare quantitative results in Table 16. We see that feature-based downsampling leads to strong improvements across the board, especially when generalizing to novel camera viewpoints. This is likely because changing camera positions introduces a large number of out of distribution points on the table, and downsampling according to features, which removes many repetitive points with the same features corresponding to points on the table, helps to mitigate their influence on the final policy.

Table 16: How does the choice of metric space for downsampling affect downstream performance? In this table we compare performance with different choices of metric space for farthest point sampling. We see that sampling according to ℓ_2 distance between features in F (right) outperforms sampling based on distances between Cartesian coordinates (left). Based on combining Adapt3R with BAKU.

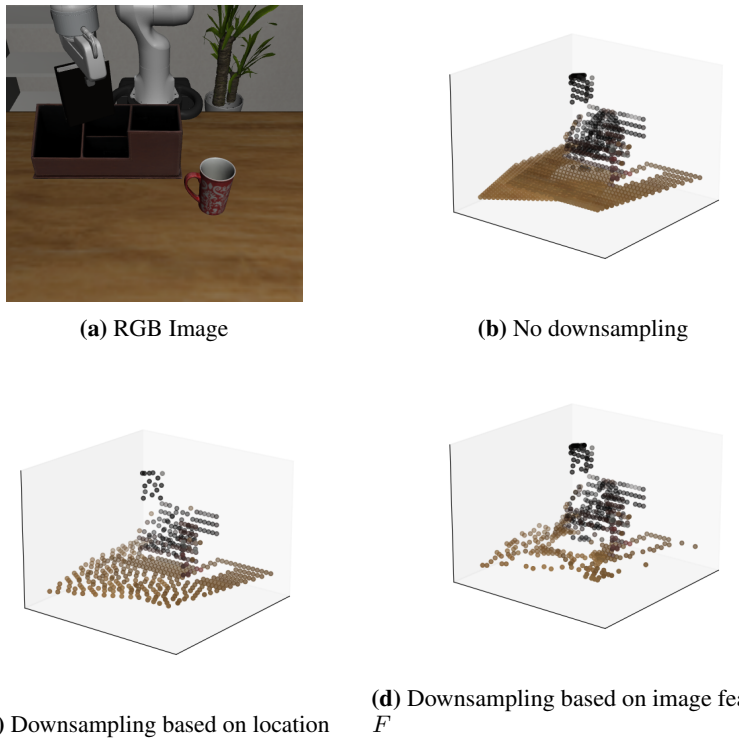
	Position-Based FPS	Feature-Based FPS
Orig.	0.916	**0.919**
UR5e	0.768	**0.801**
Kinova3	0.596	**0.653**
IIWA	0.642	**0.718**
Small	0.717	**0.847**
Medium	0.651	**0.829**
Large	0.709	**0.829**

Next, we study a qualitative comparison of the two strategies in Figure 14. We see in Figure 14c that position-based downsampling samples many points on the table, whereas in Figure 14d, the points are concentrated more around the objects in the scene.

Finally, we study the effect of p, the hyperparameter controlling the number of points in the downsampled point cloud, in Table 17. We see that it is not particularly sensitive to this hyperparameter, with a slight improvement for $p = 512$.

D.6.4 Proprioception

In Table 18 we study the effect of proprioceptive inputs on the final LIBERO results. The logic in removing it is that, because the point cloud is in the end effector's coordinate frame, the proprioceptive information is arguably baked into the observation. We see that removing the proprioception leads to a modest benefit for in-distribution and change of embodiment experiments, but leads to a massive degradation in performance when changing camera poses. Thus, for our final model we include proprioception.



(a) RGB Image

(b) No downsampling

(c) Downsampling based on location

(d) Downsampling based on image features F

Figure 14: In this figure we present visualizations of the various downsampling schemes discussed earlier in the paper. (a) RGB image of the scene for reference. (b) The original point cloud after cropping. (c) The point cloud after downsampling based on Cartesian coordinates. (d) Downsampling based on image features F.

Table 17: In this table we vary the number of points in the downsampled point cloud, p, when combining Adapt3R with BAKU. We see that while performance is not particularly sensitive to this hyperparameter, there is slightly better performance for $p \in \{256, 512\}$.

	$p = 128$	$p = 256$	$p = 512$	$p = 1024$
Orig.	0.917	**0.919**	**0.919**	0.918
UR5e	0.785	**0.802**	0.801	0.776
Kinova3	0.643	**0.681**	0.653	0.643
IIWA	0.725	**0.745**	0.718	0.672
Small	0.828	0.843	**0.847**	0.835
Medium	0.802	0.804	**0.829**	0.803
Large	0.814	0.804	**0.829**	0.799

Table 18: with and without proprioception

Algorithm	Orig.	UR5e	Kinova3	IIWA	Small	Medium	Large
ACT + Adapt3R without	**0.916**	**0.824**	**0.796**	**0.760**	0.671	0.472	0.452
ACT + Adapt3R with	0.912	0.789	0.773	0.731	**0.779**	**0.702**	**0.760**
DP + Adapt3R without	**0.899**	**0.761**	**0.568**	**0.522**	0.568	0.419	0.398
DP + Adapt3R with	0.896	0.737	0.528	0.488	**0.783**	**0.794**	**0.782**
BAKU + Adapt3R without	**0.931**	**0.813**	**0.757**	0.696	0.747	0.601	0.570
BAKU + Adapt3R with	0.920	0.787	0.654	**0.701**	**0.854**	**0.831**	**0.840**