Sequential Object-Centric Relative Placement Prediction for Long-horizon Imitation Learning

Ben Eisner, Eric Cai, Octavian Donca, Teeratham Vitchutripop, David Held
The Robotics Institute
Carnegie Mellon University
{baeisner,eycai,odonca,dheld}@andrew.cmu.edu

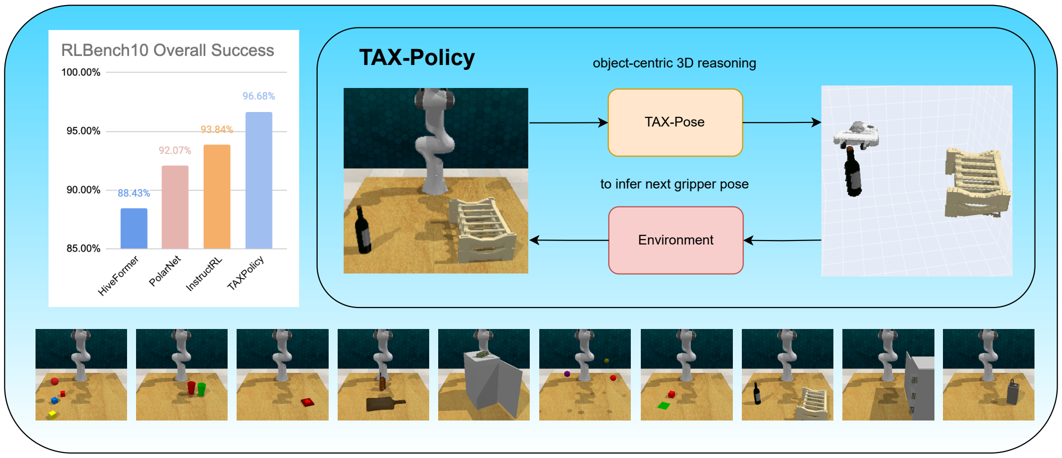

Figure 1: Overview: We present TAX-Policy, an object-centric learning method for imitating long-horizon manipulation trajectories. Our method decomposes long-horizon trajectories into a keyframe-driven sequence of object-centric geometric reasoning queries, which can be used to infer the next desired gripper state to advance the task. Using our approach, we achieve state-of-the-art success rates on the RLBench10 suite of tasks.

Abstract: Learning long-horizon manipulation skills from high-dimensional demonstrations is a critical but challenging task for robots today. Recently, several methods have been proposed to learn such skills by extracting keyframes from the demonstration, learning to predict the robot gripper's pose for the next-closest keyframe, and using motion planning or a learned low-level policy to transition between keyframes. However, these methods can suffer from imprecision and sensitivity to initial conditions. Concurrently, several promising algorithms have been proposed to efficiently learn precise object-object relationships from a small number of demonstrations, enabling generalization to new objects and new configurations. In this work, we introduce a new framework that adapts these object-object relational methods for long-horizon imitation learning. By predicting desired object-object relationships at each keyframe - instead of predicting only a gripper position - our method is able to learn precise sequences of object rearrangements that can be executed by a robot. We demonstrate the power of this technique by evaluating on the RLBench10 suite of tasks, where we achieve state-of-the-art success rates on the benchmark. Supplementary materials are available on our anonymized website.

Keywords: 3D, manipulation, object-centric, relative pose

1 Introduction

Developing robotic agents that can learn to manipulate diverse categories of objects from raw sensor inputs is an important open problem in manipulation. Although incredible progress has been made in the last few years [1], most methods which operate from visual signals still struggle when confronted with long-horizon tasks and tasks with considerable object-level and configuration-level variation [2, 3]. The problem is even more challenging in the few-shot learning case, where an agent should learn how to manipulate a novel object from only a small number of demonstrations.

One approach to imitation learning for long-horizon tasks is to introduce hierarchical structures to decompose tasks into sub-tasks or sub-skills. When considering learning from long-horizon demonstrations from high-dimensional observations, one promising technique is to detect keyframes that divide up a manipulation task into subtasks and then predict a high-level action for each subtask [2]. Past work has framed this high-level action space in terms of prediciting the future *gripper position* [2, 4, 5]. However, when the gripper is holding an object, then reasoning about the future gripper position alone may not be sufficient. For example, consider a robot grasping two different-sized cups; when stacking cups of different sizes, the agent needs to reason about the shape and size of the grasped object to determine where to move the gripper.

Our insight is that we can obtain higher precision by reasoning about the future *pose relationships* between the gripper, the scene, and the object the gripper is manipulating during each particular subtask. By predicting these desired relationships - instead of just predicting the gripper position directly - agents can learn the various geometric and visual features which are critical for task success. Learning to predict such object pose relationships allows the robot to learn to place objects more precisely and better generalize to novel configurations of the task.

In this work, using the common framework of keyframe-based imitation learning, we learn to predict a sequence of object-object pose relationships. From these object-object pose relationships, we can extract the desired robot gripper poses. To accomplish this, we leverage the recently-proposed TAX-Pose framework [6], which has shown the ability to achieve impressive levels of precision in relative prediction tasks from small numbers of demonstrations. By combining sequential relative placement prediction with motion planning between key poses, we can achieve strong performance on difficult imitation learning tasks. Our contributions are as follows:

- We identify the potential for using relative pose prediction methods in long-horizon tasks.
- We propose TAX-Policy, a straightforward, but general, framework for long-horizon imitation learning based on relative pose prediction.
- TAX-Policy achieves state-of-the-art success rates on the RLBench10 suite of tasks and demonstrates improvements over prior work that performs gripper-centric reasoning.

2 Related Work

Long-horizon imitation learning for manipulation: Many prior works have tackled the problem of imitating long-horizon manipulation behaviors. Closest to our approach are keyframing methods that predict the desired gripper pose at the next keyframe [2, 7, 5, 8, 9, 3, 10, 11, 12]. These approaches construct models which consume 2D and/or 3D sensor observations and predict a gripper output. In contrast, our method directly predicts desired object-object pose relationships at every time step, from which a desired gripper pose can be inferred. Our experiments demonstrate the benefits of this object-centric (as opposed to gripper-centric) reasoning.

Relative Placement Prediction: Our work builds directly on TAX-Pose [6], a method for learning SE(3)-equivariant task-specific pose relationships between pairs of entities in a scene (a mug and a mug rack, for instance). Variants of TAX-Pose have been proposed to achieve high-precisision relationships [13] and multimodal relationships [14]. Other methods leverage correspondences to align point clouds [15], energy fields that can be optimized to find a desired object pose [16], or use diffusion to iteratively improve pose predictions [17].

3 Problem Statement

Offline Imitation Learning from Observations: In this work, we consider the setting of offline imitation learning for robot manipulation tasks. For each manipulation task, we assume access to a set of K successful demonstrations, in which the k-th demonstration d_k consists of a sequence of T_k observations $d_k = \{o_1, ...o_{T_k}\}$. We assume that the final observation o_{T_k} completes the task. The objective is to learn a policy $\pi_\theta : \mathcal{O} \rightarrow \mathcal{A}$ from this offline dataset, which completes the tasks conveyed by the demonstrations.

Observation Space: We assume that each element of the observation space $o \in \mathcal{O}$ consists of: RGB images from a set of calibrated cameras, partial world-frame point clouds generated from back-projected depth images from each of the same cameras, gripper end-effector pose $\mathbf{T}_{\text{grip}} \in SE(3)$, and binary gripper finger positions θ_{grip}.

Action Space: The action space is the desired 6-DOF end-effector pose $\mathbf{T}_{\text{grip,goal}} \in SE(3)$ as well as a binary gripper action $a_{\text{grip}} \in \mathcal{G}$ which opens or closes the gripper. For this work, we assume access to a motion planner which moves the arm to the desired end-effector pose. We also can provide the motion planner with a flag $\gamma_{\text{mp}} \in \{0, 1\}$ to disable/enable collision-checking. We assume that the environment will plan a collision-free path to achieve this end-effector position and gripper state if possible, and do nothing if no such path exists.

Assumptions: If the gripper closes around an object stably, we assume that there will be a rigid attachment between the object and the gripper, until the gripper is opened, with a small amount of tolerance for slippage. This assumption is important for converting the object-centric reasoning that we perform in our method to the desired gripper action.

We also assume access to an oracle which can map the text description s to a subset of segmentation labels $l \in \mathcal{L}$ corresponding to objects the agent should manipulate. This oracle can be based on Grounded Segment Anything [18] or other approaches for object segmentation.

We further assume access to a method for extracting keyframes from a demonstration that divides up a log-horizon task into subtasks, similar to the heuristic proposed by James and Davison [2] and used in many concurrent methods [2, 7, 5, 8, 9, 3, 10, 11, 12].

4 Background: TAX-Pose

This work builds on top of TAX-Pose [6], a method for learning task-specific cross-object pose relationships (called the "*cross-pose*" between two objects). We will briefly describe how TAX-Pose predicts precise cross-pose relationships. First, given a point cloud observation $\mathbf{P}$, we apply an extraction function $\mathbf{P}_\mathcal{A}, \mathbf{P}_\mathcal{B} = \texttt{extract}(\mathbf{P})$ which extracts a desired *action* point cloud $\mathbf{P}_\mathcal{A}$ and an *anchor* point cloud $\mathbf{P}_\mathcal{B}$. This extraction function can be implemented based on object segmentation or using a learned task-specific segmentation function. The key desired property of `extract` is to isolate an entity $\mathcal{A}$ which is a set of points (either the gripper or a gripper-object assembly) that the robot will directly move in this subtask, and the remainder of the scene $\mathcal{B}$ that provides context on how this object should be placed. For example, $\mathcal{A}$ can be a mug and $\mathcal{B}$ would be a mug-rack on which the mug will be placed. TAX-Pose then performs object-centric reasoning to determine how the points $\mathbf{P}_\mathcal{A}$ should move relative to $\mathbf{P}_\mathcal{B}$ to achieve this subtask.

To achieve the subtask, we assume that object $\mathcal{A}$ will undergo a rigid transformation $\mathbf{T}_{\mathcal{AB}}$ to move to a desired goal position. To estimate $\mathbf{T}_{\mathcal{AB}}$, we perform the following steps: First, we encode each of $\mathbf{P}_\mathcal{A}, \mathbf{P}_\mathcal{B}$ with per-point feature encoders $\Psi_\mathcal{A} = g_\mathcal{A}(\mathbf{P}_\mathcal{A}), \Psi_\mathcal{B} = g_\mathcal{B}(\mathbf{P}_\mathcal{B})$. Next, these per-point features are transformed with a cross-attention module $g_\mathcal{T}$ and added to the original features to produce final per-point features:

$$\Phi_\mathcal{A} = \Psi_\mathcal{A} + g_{\mathcal{T}_\mathcal{A}}(\Psi_\mathcal{A}, \Psi_\mathcal{B}), \quad \Phi_\mathcal{B} = \Psi_\mathcal{B} + g_{\mathcal{T}_\mathcal{B}}(\Psi_\mathcal{A}, \Psi_\mathcal{B}). \tag{1}$$

We can interpret these features for the purpose of computing a soft cross-object correspondence (see Pan et al. [6] for details) that assigns a normalized similarity score for each point on $\mathcal{A}$ to $\mathcal{B}$,

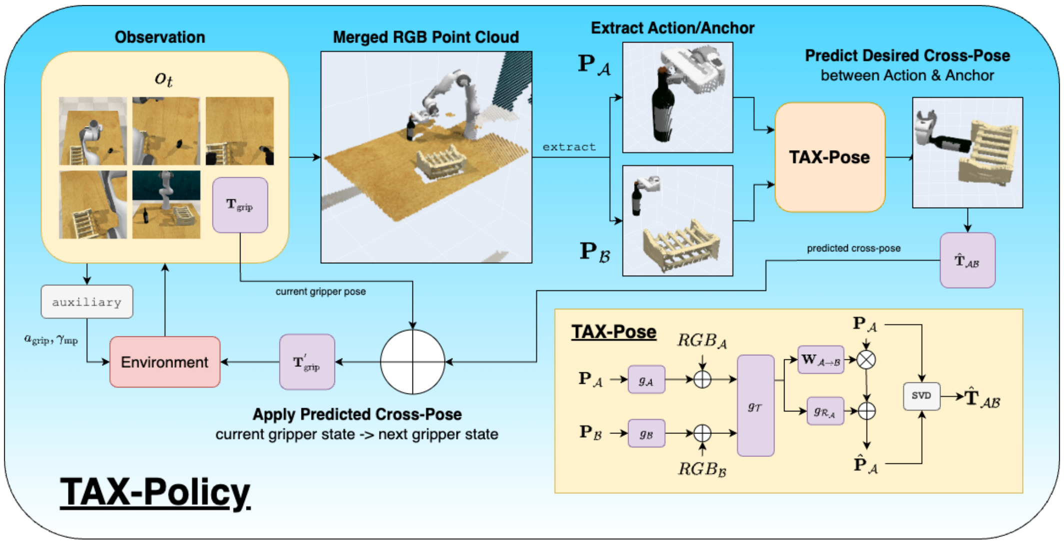
Observation
o_t
$\mathbf{T}_{grip}$
Merged RGB Point Cloud
extract
Extract Action/Anchor
$\mathbf{P}_{\mathcal{A}}$
$\mathbf{P}_{\mathcal{B}}$
Predict Desired Cross-Pose between Action & Anchor
TAX-Pose
predicted cross-pose
$\hat{\mathbf{T}}_{\mathcal{AB}}$
current gripper pose
auxiliary
a_{grip}, γ_{mp}
Environment
$\mathbf{T}'_{grip}$
Apply Predicted Cross-Pose
current gripper state -> next gripper state
TAX-Policy
TAX-Pose
$RGB_{\mathcal{A}}$
$RGB_{\mathcal{B}}$
$g_{\mathcal{A}}$
$g_{\mathcal{B}}$
$g_{\mathcal{T}}$
$\mathbf{W}_{\mathcal{A}\to\mathcal{B}}$
$g_{\mathcal{R}_{\mathcal{A}}}$
SVD
$\hat{\mathbf{P}}_{\mathcal{A}}$

Figure 2: Full system diagram. TAX-Policy first 1) synthesizes multi-camera observations into a unified RGB point cloud, 2) extracts a set of action points and anchor points, 3) leverages a multi-phase TAX-Pose model to predict a relative transform $\hat{\mathbf{T}}_{\mathcal{AB}}$ which should advance the system to the next critical keyframe relationship, 4) applies this relative transformation to the current end-effector pose $\mathbf{T}_{\text{grip}}$ to get the next end-effector state $\mathbf{T}'_{\text{grip}}$, and 5) commands the environment to execute this action, alongside auxiliary gripper and motion planning commands, producing the next observation.

written in matrix form as $\mathbf{W}_{\mathcal{A}\to\mathcal{B}} = s(\Phi_{\mathcal{A}}, \Phi_{\mathcal{B}})$. These weightings can be used to infer a set of soft corresponding points inside the convex hull of $\mathcal{B}$, which can further be corrected using a set of predicted residual vectors to yield a set of predicted locations $\hat{\mathbf{P}}_{\mathcal{A}}$ representing where each point in $\mathbf{P}_{\mathcal{A}}$ should be transported to achieve the desired pose relationship:

$$\hat{\mathbf{P}}_{\mathcal{A}} = \mathbf{W}_{\mathcal{A}\to\mathcal{B}}\mathbf{P}_{\mathcal{B}} + g_{\mathcal{R}_{\mathcal{A}}}(\Phi_{\mathcal{A}}) \tag{2}$$

Finally, we can extract the desired transform $\mathbf{T}_{\mathcal{AB}}$ by solving a least-squares orthogonal Procrustes problem, which has a well known closed-form solution: $\mathbf{T}_{\mathcal{AB}} = \texttt{SVD}(\mathbf{P}_{\mathcal{A}}, \hat{\mathbf{P}}_{\mathcal{A}})$. A key property of this algorithm is that it is provably translation-invariant, and can be trained to be nearly SE(3)-equivariant through data augmentation. Furthermore, this method has been shown across several settings [6, 13] to learn to precisely predict object placement from a small number of examples.

5 Method: TAX-Policy

We will now introduce our framework to construct an agent for multi-step manipulation tasks using object-centric reasoning, which we call TAX-Policy. Below, we describe how we can convert keyframes into a dataset suitable for training a precise relative pose estimator like TAX-Pose [6]. We then describe how we can construct a policy purely out of these relative pose predictions. See Figure 2 for a detailed overview of our system.

Past approaches predict the next keyframe gripper pose: In Section 3, we explained how for each demonstration d_k, we could extract a set of H keyframes $\{d_{k,1}^{(key)}, \ldots, d_{k,H}^{(key)}\}$, and then extract the gripper pose at each keyframe $T_{k,h}$. Prior works [2, 7, 5, 8, 9, 3, 10, 11, 12] trained models which consume the current observation $o_{k,h}$ and predict the desired gripper pose of the next keyframe as $\hat{T}_{k,h+1} = f_\theta(o_{k,h})$. These propsed models perform gripper-centric reasoning, predicting from the observation the desired gripper pose. In contrast, our approach, described below, uses object-centric reasoning; we predict object-object relationships, which we then uses to predict the next keyframe gripper pose.

TAX-Policy predicts keyframe object-object relationships: TAX-Pose and followup work [6, 19] have demonstrated that the pose relationships between two objects can be precisely predicted from point cloud observations, by predicting cross-object correspondences (see Section 4). TAX-Pose consumes a pair of (action, anchor) point clouds $(\mathbf{P}_{\mathcal{A}}, \mathbf{P}_{\mathcal{B}})$ expressed in any coordinate frame, and predicts a transform $\mathbf{T}_{\mathcal{AB}}$ which transports object $\mathcal{A}$ to the desired "Cross-Pose" with respect to object $\mathcal{B}$ in that coordinate frame. As such, TAX-Pose is directly applicable to the problem of predicting keyframe gripper poses. To train a TAX-Pose model to represent these demonstration relationships, we need to extract the following point clouds from our keyframe dataset:

- **Current action point cloud, $\mathbf{P}_{\mathcal{A}}$**: The *action* point cloud includes all points that the robot will move in each subtask. This always includes the robot gripper; Furthermore, in keyframes in which an object is currently being grasped, this object's points are also included in the action point cloud.
- **Current anchor point cloud, $\mathbf{P}_{\mathcal{B}}$**: The *anchor* point cloud is the set of points that define the objects in the environment that the action object should move relative to for this subtask. We simply include the entire scene point cloud as the anchor point cloud $\mathbf{P}_{\mathcal{B}}$, after filtering out the background (e.g. table) and non-gripper robot points. Crucially, we include the gripper in the anchor point cloud when it is present in the scene because in many phases, the gripper moves relative to its prior position (i.e. pre-grasp to grasp).

We train TAX-Pose to predict the "cross-pose", e.g. the transformation $\mathbf{T}_{\mathcal{AB}}$ by which the points in object $\mathcal{A}$ need to move to achieve the goal configuration relative to object $\mathcal{B}$. Thus we train a function to predict this transform from the action and anchor point clouds:

$$\hat{\mathbf{T}}_{\mathcal{AB}} = f_\theta^{TP}(\mathbf{P}_{\mathcal{A}}, \mathbf{P}_{\mathcal{B}}).$$

To supervise this function, we reason as follows: if we assume the points in object $\mathcal{A}$ are rigidly attached to the robot gripper, then the object transform is the same as the gripper transform. Given the current gripper pose $T_{k,h}$ and the next gripper pose $T_{k,h+1}$, we compute the ground-truth transform:

$$\mathbf{T}^*_{\mathcal{AB}} = T_{k,h+1}T_{k,h}^{-1}.$$

We train with a similar loss as from prior work (see [6] for details). Given the predicted cross-pose $\mathbf{T}_{\mathcal{AB}}$, we can then predict the next gripper pose as

$$\hat{T}_{k,h+1} = T_{k,h} \cdot \hat{\mathbf{T}}_{\mathcal{AB}}.$$

Compared to prior work that predicts the next gripper pose $\hat{T}_{k,h+1}$ directly, we learn to predict the cross-pose transform $\mathbf{T}_{\mathcal{AB}}$ based on geometrically reasoning about the relationship between the object point clouds $\mathbf{P}_{\mathcal{A}}$ and $\mathbf{P}_{\mathcal{B}}$ (see Section 4). This object-centric reasoning allows our method to make much more precise predictions about the future gripper pose compared to prior work. See Algorithm 1 for a pseudocode implementation of a full policy rollout.

Modifying TAX-Pose: In order to train a TAX-Pose model with all of the necessary context required to correctly predict a precise cross-pose, we make the below modifications to the TAX-Pose architecture and algorithm:

1. **Adding (optional) RGB channels to TAX-Pose**: Several of the tasks in the RLBench benchmark condition on the color of objects (i.e. `pick_up_cup`), where distractor objects may be identical geometrically. To disambiguate, in tasks that require color (identifiable in the prompt string s) we incorporate an additional color channel processing head. See Appendix B for details.
2. **Including a context vector corresponding to the timestep**: We inject a one-hot encoding c_{phase} of the current phase (i.e. subtask number) for a particular task, to help TAX-Pose disambiguate between keyframes that have small geometric variations.
3. **Multi-relationship training**: In the original TAX-Pose paper, a single TAX-Pose model is only trained to predict a single relationship (i.e. mug-on-rack). In this work, a single model is used for all the desired relationships for a task; the model must disambiguate which relationship should currently be predicted.

Algorithm 1: TAX-Policy: Policy Based on Precise Placement with TAX-Pose

```
Input: Initial observation $o_0$
done ← False;
$o_t \leftarrow o_0$;
while !done do
    $P_A, P_B, T_{\text{grip}} \leftarrow$ extract($o_t$) ; // (1) Extract the action and anchor point
                                  // clouds and the gripper pose from the
                                  // observation.
    $\hat{\mathbf{T}}_{\mathcal{AB}} \leftarrow f_\theta^{TP}(P_A, P_B)$ ;        // (2) Predict the cross-pose transformation.
    $T'_{\text{grip}} \leftarrow \hat{\mathbf{T}}_{\mathcal{AB}} \cdot T_{\text{grip}}$ ;           // (3) Transform the gripper pose by the
                                  // predicted cross-pose.
    $a_{\text{grip}}, \gamma_{\text{mp}} \leftarrow$ auxiliary($o_t$) ;      // (4) Get the next gripper open-close
                                     // and the motion planner parameters
                                     // (e.g.  collision-checking).
    $o_t$, done ← step_fn($T'_{\text{grip}}, a_{\text{grip}}, \gamma_{\text{mp}}$) ;    // (5) Execute the motion planner to
                                         // move the gripper to the
                                         // predicted gripper pose $T'_{\text{grip}}$.
end
```

6 Experiments

In this section, we describe our experiments to evaluate the value of our object-centric approach compared to the gripper-centric approach of prior work.

6.1 RLBench10 Setting

RLBench10 Tasks: Tasks defined in the RLBench [4] framework all involve controlling a simulated Franka Emika Panda 7-DOF robot arm with a parallel-jaw gripper to accomplish various object rearrangement tasks in scenes with randomized initial conditions. Each task in the RLBench suite offers its own binary success criteria, most of which determine whether a specific object or set of objects have been rearranged within some positional threshold of goal positions. In this work, we evaluate on the **RLBench10** suite of tasks, a subset of tasks initially proposed by Guhur et al. [8]. Visualizations of these tasks can be found in Figure 3. More details can be found in Appendix A.

Demonstration Dataset: For each task, we generate a set of 100 training episodes with a single task variation using the standard RLBench dataset generation script, with minor patches from Shridhar et al. [5] to also record whether collision-checking is needed at each state. We apply the standard keyframing used in several methods [5, 2]. See Appendix C for more details.

Training and Evaluation Details: We train a single TAX-Pose model for each task on all 100 demonstrations, and evaluate on 500 unseen initial configurations using the same variation as seen during training. See Appendix D for more details on architecure choices, hyperparameters, etc.

6.2 Adapting TAX-Policy for RLBench10

Motion Planning Modifications: During the course of training and evaluating models in the RL-Bench suite of environments, we encountered several flaws in the environments' motion-planning and low-level control that violate our policy's assumptions. First, we find that the motion planner provided in RLBench's `EndEffectorPoseViaPlanning` action space does not always find valid collision-free trajectories when feasible ones exist. Second, we find that the provided low-level controller does not always faithfully follow collision-free trajectories proposed by the motion-planner, particularly when the robot is close to its own joint limits, is near a colliding surface, or is grasping an object. A significant fraction of the time, these errors cause task failure; as such, we make several minor heuristic changes to the execution of the next-gripper commands proposed in TAX-Policy:

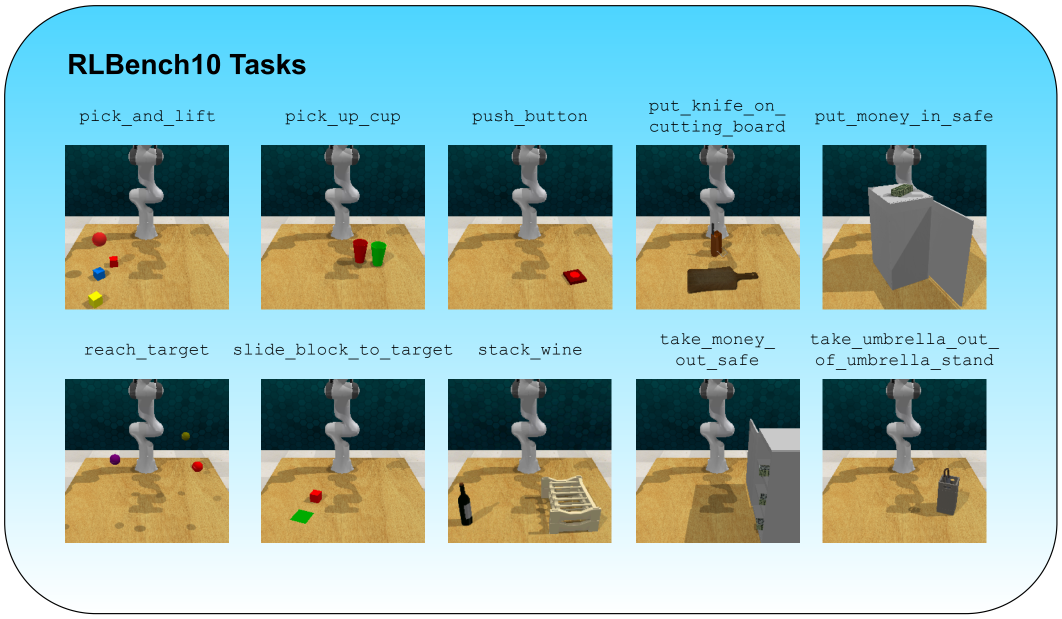

Figure 3: The RLBench10 suite of tasks

- **Action repeat**: Following Xian et al. [3], if the gripper does not actually reach the commanded position by the time the next observation is obtained, repeat the same action until the gripper reaches the desired pose (up to a maximum number of repeats).
- **Random action pertubations**: If the motion planner fails to find a trajectory to an initial proposed gripper position, randomly perturb the proposed gripper position by a small amount until motion planning succeeds.
- **Post-grasp z-offsets**: When motion planning with a grasped object, we modify the motion planning trajectory to include a small positive z-offset waypoint directly above the commanded position, before the final position is achieved. This corrects the behavior in which the controller does not properly perform gravity compensation, resulting in a collision.

Together, these modifications enable the agent to find and accomplish collision-free motion plans from the current to the commanded position, significantly reducing the number of rollout failures.

Auxiliary Inputs: One key implementation detail is that for the RLBench experiments, to determine whether to use collision checking or whether to open-close the gripper (the `auxiliary` function in Algorithm 1) we simply replay the a_{grip} and γ_{mp} values seen in one arbitrarily-selected demonstration for the evaluated task, entirely open-loop. We could also train a function to predict these values as was done in prior work [2].

6.3 Results

Baselines: We compare our method against several recent approaches which attempt to predict the gripper state at the next closest keyframe [8, 10, 20]; see Appendix E for details. Unlike these baselines, we require a method to extract the "action object" which will move at each phase from the rest of the scene. In these experiments, we use ground-truth segmentation to obtain the action object; this can be replaced in future work by a segmentation module (e.g. Segment Anything [21]). We report the success rates published in each paper in Table 1.

Although both Act3D [9] and ChainedDiffuser [3] evaluate on RLBench, we do not directly compare against them quantitatively. Act3D does not report results on RLBench10. ChainedDiffuser is not comparable because their focus is on how to estimate low-level trajectories using diffusion instead of using the provided motion planner; we consider the contributions of ChainDiffuser orthogonal to ours, as they build on top of keyframe pose estimation techniques and can be combined with our work instead of using motion planning. Nevertheless, we match ChainedDiffuser's performance across the benchmark and match or exceed their performance on most individual tasks.

Analysis: Our method performs extremely well on this benchmark; see Table 1 for detailed results. We achieve a 96.7% success rate on average, which is state-of-the-art across the baselines (and matches ChainedDiffuser's performance). Moreover, we achieve SOTA performance on 6 of the 10 tasks. Qualitatively, the predictions that TAX-Policy makes are nearly always highly precise, and the motion-planner + controller usually successfully accomplish the commanded gripper states with high precision. Our policy's strong performance on this task provides support for the hypothesis that object-centric reasoning can be a powerful inductive bias for manipulation tasks. Additional ablations evaluated the impact of various modifications both to the TAX-Pose inputs and the RLBench policy; see Appendix F. The few remaining failure modes are described in Appendix G.

	Overall (mean)	pick_and_lift	pick_up_cup	push_button	put_knife_on...	put_money_in_safe	reach_target	slide_block_to_target	stack_wine	take_money_out_safe	take_umbrella_out...
HiveFormer [8]	88.4%	92.2%	77.1%	99.6%	69.7%	96.2%	100.0%	95.4%	81.9%	82.1%	90.1%
PolarNet [10]	92.1%	96.7%	91.9%	**99.9%**	82.5%	96.1%	99.9%	93.5%	94.1%	68.6%	97.5%
InstructRL [20]	93.8%	**97.8%**	84.5%	99.5%	84.5%	**98.7%**	**100.0%**	97.5%	93.2%	89.8%	92.94%
TAX-Policy (Ours)	**96.7%**	94.0%	**94.6%**	95.8%	**97.6%**	96.8%	97.0%	**98.6%**	**97.6%**	**95.4%**	**99.4%**

Table 1: Success rates on the RLBench10 Tasks. We find that the combination of our use of TAX-Pose for predicting precise relative relationships, as well as small tweaks to the policy to improve motion-planning fidelity, yield state-of-the-art results on the RLBench10 suite of tasks.

7 Limitations

There are a few key assumptions that we leverage in this work, which each limit its generality to a different degree. The largest limitation is that TAX-Policy (and TAX-Pose itself) assumes that the action objects can be transported rigidly, meaning deformables cannot be adequately represented, nor can grasped objects which may experience significant slippage while grasped. Keyframing is another limitation: some tasks do not have well-defined stop/start points which make for consistent keyframing. Requiring segmentations (ground-truth or learned) is not a large limitation, as there are many works that are effective at segmenting objects and scenes, even based on text prompts. Finally, some tasks may need more than just geometric reasoning to solve them - particularly high-contact and dynamic tasks. Additionally, the TAX-Pose training algorithm and architecture we selected does not currently handle multimodal relationships very gracefully, and predicts just a single mode.

8 Conclusion

In this work, we explore how sequential object-centric geometric reasoning can be applied to imitation learning for long-horizon manipulation tasks. We propose a policy, TAX-Policy, which can consume a set of long-horizon demonstrations and learn a function which can predict relative pose relationships that correspond to each keyframe relationship in that demonstration. We demonstrate that this approach is extremely effective when applied to the RLBench10 set of tasks, where we achieve state-of-the-art performance on the benchmark. In future work, we would like to incorporate a generative version of TAX-Pose into this policy to handle multimodality more precisely. Additionally, we would like to incorporate more low-level goal-conditioned control (similar to ChainedDiffuser [3]), to capture desired behaviors more precisely and handle the placement of deformable objects.

Acknowledgments

If a paper is accepted, the final camera-ready version will (and probably should) include acknowledgments. All acknowledgments go at the end of the paper, including thanks to reviewers who gave useful comments, to colleagues who contributed to the ideas, and to funding agencies and corporate sponsors that provided financial support.

References

[1] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems XIX*. Robotics: Science and Systems Foundation, July 2023.

[2] S. James and A. J. Davison. Q-Attention: Enabling efficient learning for Vision-Based robotic manipulation. *IEEE Robotics and Automation Letters*, 7(2):1612–1619, Apr. 2022.

[3] Z. Xian, N. Gkanatsios, T. Gervet, T.-W. Ke, and K. Fragkiadaki. ChainedDiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2323–2339. PMLR, 2023.

[4] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison. RLBench: The robot learning benchmark learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, Apr. 2020.

[5] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-Actor: A Multi-Task transformer for robotic manipulation. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 785–799. PMLR, 2023.

[6] C. Pan, B. Okorn, H. Zhang, B. Eisner, and D. Held. TAX-Pose: Task-Specific Cross-Pose estimation for robot manipulation. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1783–1792. PMLR, 2023.

[7] S. James, K. Wada, T. Laidlow, and A. J. Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022.

[8] P.-L. Guhur, S. Chen, R. G. Pinel, M. Tapaswi, I. Laptev, and C. Schmid. Instruction-driven history-aware policies for robotic manipulations. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 175–187. PMLR, 2023.

[9] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki. Act3D: 3D feature field transformers for Multi-Task robotic manipulation. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 3949–3965. PMLR, 2023.

[10] S. Chen, R. G. Pinel, C. Schmid, and I. Laptev. PolarNet: 3D point clouds for Language-Guided robotic manipulation. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 1761–1781. PMLR, 2023.

[11] S. Liu, S. James, A. Davison, and E. Johns. Auto-Lambda: Disentangling dynamic task relationships. Apr. 2022.

[12] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox. RVT: Robotic view transformer for 3D object manipulation. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 694–710. PMLR, 2023.

[13] B. Eisner, Y. Yang, T. Davchev, M. Vecerik, J. Scholz, and D. Held. Deep SE(3)-equivariant geometric reasoning for precise placement tasks. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*, Apr. 2024.

[14] J. Wang, O. Donca, and D. Held. Learning distributional demonstration spaces for Task-Specific Cross-Pose estimation. May 2024.

[15] Y. Wang and J. M. Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3523–3532, 2019.

[16] A. Simeonov, Y. Du, Y.-C. Lin, A. R. Garcia, L. P. Kaelbling, T. Lozano-Pérez, and P. Agrawal. SE(3)-Equivariant relational rearrangement with neural descriptor fields. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 835–846. PMLR, 2023.

[17] A. Simeonov, A. Goyal, L. Manuelli, Y.-C. Lin, A. Sarmiento, A. R. Garcia, P. Agrawal, and D. Fox. Shelving, stacking, hanging: Relational pose diffusion for multi-modal rearrangement. In J. Tan, M. Toussaint, and K. Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2030–2069. PMLR, 2023.

[18] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, Z. Zeng, H. Zhang, F. Li, J. Yang, H. Li, Q. Jiang, and L. Zhang. Grounded SAM: Assembling Open-World models for diverse visual tasks. Jan. 2024.

[19] B. Eisner, Y. Yang, T. Davchev, M. Vecerik, J. Scholz, and D. Held. Deep SE(3)-Equivariant geometric reasoning for precise placement tasks. In *The Twelfth International Conference on Learning Representations*, 2024.

[20] H. Liu, L. Lee, K. Lee, and P. Abbeel. Instruction-Following agents with multimodal transformer. Oct. 2022.

[21] 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.

[22] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[23] G. Qian, Y. Li, H. Peng, J. Mai, H. Hammoud, M. Elhoseiny, and B. Ghanem. PointNeXt: Revisiting PointNet++ with improved training and scaling strategies. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 23192–23204. Curran Associates, Inc., 2022.

Appendix

Table of Contents

A More details on RLBench

One of the core benefits of the RLBench framework (from a practitioner's perspective) is that it comes with a demonstration-generation engine: for each new task specified using the logical success condition primitives defined by the framework, the engine can generate motion plans which reliably accomplish all success conditions given any initial scene configuration, provided those conditions are achievable. The resulting plans, however, are empirically extremely consistent - while the motion planning results controlling the arm's movement through free-space have some diversity depending on scene configuration, there are two consistent features of each demonstration for a single task:

1. **Consistent keyframes $\implies$ consistent, identifiable task phases**: Throughout the course of each demonstration, the expert agent always executes a consistent number of discrete motions, where the gripper roughly comes to rest before transitioning to the next motion (and is detectable by measuring the end-effector velocity $\dot{\theta}_{\text{ee}}$ in the observation space of the demonstrations). These pauses occurs both when the end-effector moves through free space without holding an object and when the end-effector has already grasped an object, and mark transition points between discernable "phases" for each task. This property has been widely reported in the literature (TODO cite a few keyframe rlbench papers), where each pause frame is usually referred to as a **keyframe**. Importantly, the keyframes (and the semantic interpretation of motions between keyframes) are semantically aligned across all demonstrations, and can be denoted as unique phases h_i. For example, in the `stack_wine` task, there are 5 distinct phases demarcated by keyframes, which appear in exactly the same order in *every* demonstration, without exception. Thus, every demonstration has H phases, and using the widely-used keyframe detection strategy `detect_keyframes`, a demonstration d_k can be parsed into H distinct keyframes:
$$d_k^{(key)} = \texttt{detect\_keyframes}(d_k) = \{o_{h_1}, \dots, o_{h_H}\}$$
2. **Each keyframe achieves one of a unique discrete set of relative relationships**: One key property of these consistent keyframes is that, across demonstrations for a task, for a given phase h_i the gripper always achieves one of a discrete number of pose relationships $T_{h_i}^{(j)} \in \{T_{h_i}^{(1)} \dots T_{h_i}^{(J)}\}$ relative to the pose of a (consistent) specific object in the scene. There are a small, discrete number of such relative poses if expressed in the object frame. *Usually, there is only a single relative pose relationship achieved across all demonstrations.*

This condition is only violated when there is perfect object-level symmetry (as occurs in a handful of RLBench tasks).

B Color Channel architecture modifications

The basic formulation of TAX-Pose does not include any channels for color information. To address this, we make a small modification. We simple take the dense per-point encoded features for each object ψ_A, ψ_B (without color), concatenate color features (RGB) as additional channels to each objects' shape features, and run each point cloud through a per-point MLP in order to mix in color information to obtain new per-point features ψ'_A and ψ'_B. These features are then passed as normal to the cross-attention module as shown in Equation 1.

C Additional Dataset Details

We use standard dataset generation techniques for RLBench, using the procedural expert provided with the original RLBench dataset. In addition to using a minor fork of the original code to extract whether the expert used collision checking [5], we do something slightly different many of the other RLBench imitation learning methods: to extract a sufficient number of visual points on the manipulated objects, we generate demos with 256x256 RGB-D images instead of the default 128x128 size. Otherwise, dataset generation is the same.

D Training Details

We train our model on 3080 Ti GPUs for 30 hours each. We have observed that, although training seems to converge after roughly 3 hours for each task, we only achieve peak prediction precision performance after training for roughly 30 hours. We use DGCNN [15] as the action/anchor backbone network, following the default from TAX-Pose [6].

E Baseline Descriptions

- HiveFormer [8]: This method encodes a history of time steps, where RGB-D and proprioception information are each processed with a UNet [22] and then synthesized together with a transformer, before being decoded into the next gripper state.
- PolarNet [10]: This method processes point clouds using PointNeXt [23], passes these features through a transformer, and decodes positions using a spatial heatmap and rotation using a separate MLP head.
- InstructRL [20]: This method uses a pretrained multimodal encoder to process RGB + proprioception observations for a set of history observations, and then mixes these encoded features via a transformer, which is then decoded into actions.

F Ablation Descriptions

We evaluated the impact of various modifications both to the TAX-Pose inputs and the RLBench policy modifications. No single change was responsible for a major performance increase; see Appendix F for details. See Table 2 for quantitative ablation results. The models we consider are:

- Basic TAX-Policy: A variant of our final method without any of the below additions.
- Adding RGB: We add RGB to the observation space, so that tasks with geometrically-identical (but differently-colored) objects can be distinguished.
- Action Repeats: We command the same action in the environment until the environmnet has accomplished the commanded end-effector pose, following [3].

	Overall (mean)	pick_and_lift	pick_up_cup	push_button	put_knife_on...	put_money_in_safe	reach_target	slide_block_to_target	stack_wine	take_money_out_safe	take_umbrella_out...
Basic TAXPolicy	47.0%	-%	3.0%	88.0%	92.0%	-%	9.0%	0.0%	61.0%	-%	78.0%
+ RGB	62.0%	13.0%	33.0%	49.0%	80.0%	-%	10.0%	76.0%	53.0%	-%	95.0%
+ repeat-action	70.0%	24.0%	59.0%	38.0%	87.0%	-%	**100.0**%	93.0%	61.0%	-%	**100.0**%
+ wrist cam	95.0%	86.0%	**97.0**%	-%	95.0%	96.0%	96.0%	95.0%	-%	**97.0**%	99.0%
Gripper-only	81.1%	4.0%	94.0%	90.0%	**100.0**%	**99.0**%	53.0%	96.0%	89.0%	88.0%	98.0%
TAX-Policy (Ours)	**96.7**%	**94.0**%	94.6%	**95.8**%	97.6%	96.8%	97.0%	**98.6**%	**97.6**%	95.4%	99.4%

Table 2: Success rates of methodological ablations for TAXPolicy on the RLBench10 suite of tasks. Column entries marked with "-%" indicate training or evaluation runs where there was a disruption during execution, or where a bug was found after evaluation which invalidated the result. Overall metrics are calculated by averaging valid entries (excluding the invalidated entries).

- Including Wrist Camera: We additionally include wrist viewpoints, which are critical for observing the action object in some degenerate occlusion configurations.
- Gripper-only Action Object: We evaluate whether removing the manipulated object from the action point cloud (leaving only the gripper) impacts the policy quality.

G Failure Modes

Nearly all of the failure modes that we observe arise from one of two issues:

- **Motion planning and control errors**: Despite the adjustments made to the motion planner, we occasionally encounter control errors where the gripper incidentally knocks into objects or the environment (despite the expectation of collision-free motion), causing the observations to deviate dramatically from the training trajectories. Without a stronger controller or low-level action control (as proposed by Xian et al. [3]), it may not be feasible to get a behavior-cloned agent predicting keyframe poses to be robust to these errors.
- **Multimodality and Symmetry**: In three tasks (`pick_and_lift`, `pick_up_cup`, and `reach_target`), there are several identical objects in the scene, which causes the demonstrations to be highly multimodal. TAX-Pose has been known to not perfectly represent symmetric and multimodal scenes because of the way SE(3) equivariance is achieved [6], and although a generative version of TAX-Pose has been proposed [14], we were unable to successfully get a generative version to learn precise relationships suitable for solving this task. We expect that a precise, generative version of TAX-Pose would be able to remove the other class of failures.