

A Background

A.1 Imitation learning

The goal of imitation learning is to learn a behavior policy π^b given access to either the expert policy π^e or trajectories derived from the expert policy τ^e . This work operates in the setting where the agent only has access to observation-based trajectories, i.e. $\tau^e \equiv \{(o_t, a_t)_{t=0}^T\}_{n=0}^N$. Here N and T denote the number of demonstrations and episode timesteps respectively. We choose this specific setting since obtaining observations and actions from expert or near-expert demonstrators is feasible in real-world settings [73, 74] and falls in line with recent work in this area [29, 75, 73, 48, 76].

A.2 Behavior Cloning

Behavior Cloning (BC) [77, 78] corresponds to solving the maximum likelihood problem shown in Eq. 5. Here \mathcal{T}^e refers to expert demonstrations. When parameterized by a normal distribution with fixed variance, the objective can be framed as a regression problem where, given observations o^e , π^{BC} needs to output a^e .

$$\mathcal{L}^{BC} = \mathbb{E}_{(o^e, a^e) \sim \mathcal{T}^e} \|a^e - \pi^{BC}(o^e)\|^2 \quad (5)$$

After training, π^{BC} learns to mimic the actions corresponding to the observations seen in the demonstrations.

A.3 Semantic Correspondence

Finding corresponding points across multiple images of the same scene is a well-established problem in computer vision [79, 80]. Correspondence is essential for solving a range of larger challenges, including 3D reconstruction [81, 82], motion tracking [72, 83, 84, 85], image registration [80], and object recognition [86]. In contrast, semantic correspondence focuses on matching points between a source image and an image of a different scene (e.g., identifying the left eye of a cat in relation to the left eye of a dog). Traditional correspondence methods [80, 79] often struggle with semantic correspondence due to the substantial differences in features between the images. Recent advancements in semantic correspondence utilize deep learning and dense correspondence techniques to enhance robustness [87, 88, 89] across variations in background, lighting, and camera perspectives. In this work, we adopt a diffusion-based point correspondence model, DIFT [71], to establish correspondences between a reference and an observed image, which is illustrated in Figure 4.

A.4 Point Tracking

Point tracking across videos is a problem in computer vision, where a set of reference points are given in the first frame of the video, and the task is to track these points across multiple frames of the video sequence. Point tracking has proven crucial for many applications, including motion analysis [90], object tracking [91], and visual odometry [92]. The goal is to establish reliable correspondences between points in one frame and their counterparts in subsequent frames, despite challenges such as changes in illumination, occlusions, and camera motion. While traditional point tracking methods rely on detecting local features in images, more recent advancements leverage deep learning and dense correspondence methods to improve robustness and accuracy [72, 83, 84]. In this work, we use Co-Tracker [72] to track a set of reference points defined in the first frame of a robot’s trajectory. These points tracked through the entire trajectory are then used to train generalizable robot policies for the real world.

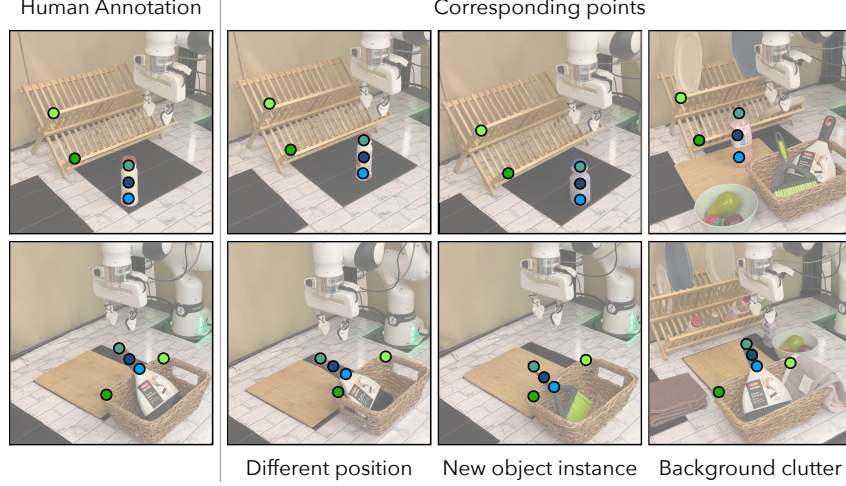


Figure 4: Results of the correspondence model when used for the put bottle on rack and sweep broom tasks. On the left is a frame with human annotations for the object points. On the right, we show that semantic correspondence can identify the same points across different positions, new object instances, and background clutter.

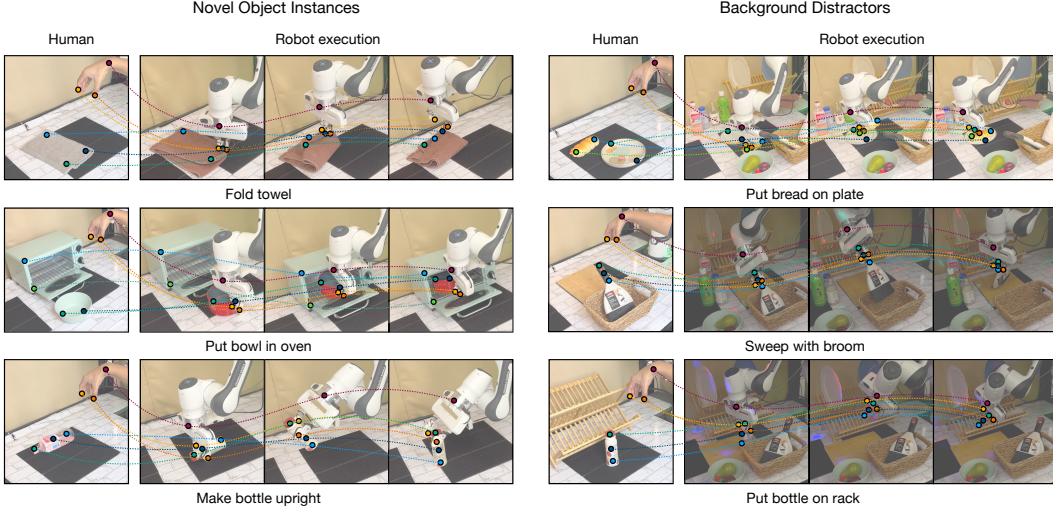


Figure 5: Real-world rollouts showing that Point Policy generalizes to novel object instances and is robust to background distractors.

B Algorithmic Details

B.1 Point Triangulation

Point triangulation is a fundamental technique in computer vision used to reconstruct 3D points from their 2D projections in multiple images. Given n cameras with known projection matrices P_1, P_2, \dots, P_n and corresponding 2D image points x_1, x_2, \dots, x_n , the goal is to find the 3D point X that best explains these observations.

The projection of X onto each image is given by:

$$x_i \sim P_i X$$

where \sim denotes equality up to scale.

One common approach is the Direct Linear Transform (DLT) method:

543 1. For each view i , we can form two linear equations:

$$x_i(p_i^3 \cdot X) - (p_i^1 \cdot X) = 0$$

544

$$y_i(p_i^3 \cdot X) - (p_i^2 \cdot X) = 0$$

545

where p_i^j is the j -th row of P_i .

546

2. Combining equations from all views, we get a system $AX = 0$.

547

3. The solution is the unit vector corresponding to the smallest singular value of A , found via Singular Value Decomposition (SVD).

548

549 For optimal triangulation, we aim to minimize the geometric reprojection error.

550 B.2 Hyperparameters

551 The complete list of hyperparameters is provided in Table 5. Details about the number of demon-
552 strations for each task has been included in Section D.2, and summarized in Table 6. All the models
553 have been trained using a single NVIDIA RTX A4000 GPU. The policy contains a total of 3.3M
554 parameters and requires at most 30 minutes for training (40-50k iterations). However, preprocessing
555 the collected data, where the entire dataset is labelled with semantically meaningful key points using
556 DIFT and Co-Tracker, takes around 1 hour.

Table 5: List of hyperparameters.

Parameter	Value
Learning rate	$1e^{-4}$
Image size	256×256 (for BC, BC w/ Depth, MT- π)
Batch size	64
Optimizer	Adam
Number of training steps	100000
Transformer architecture	minGPT [93] (for BC, BC w/ Depth, P3PO, Point Policy) Diffusion Transformer [16] (for MT- π)
Hidden dim	256
Observation history length	1 (for BC, BC w/ Depth) 10 (for MT- π , P3PO, Point Policy)
Action head	MLP
Action chunk length	20

557 C Implementation Details for MT- π

558 Since the official implementation of MT- π is not yet public available, we adopt the Diffusion Trans-
559 former (DiT) based implementation of a 2D point track prediction model proposed by Bharadhwaj
560 et al. [16]. We modify the architecture such that given a single image observation and robot motion
561 tracks on the image, the model predicts future tracks of the robot points. These robot tracks are then
562 converted to 3D using corresponding tracks for two camera views. The robot action is then computed
563 from the 3D robot tracks using the same rigid-body geometry constraints as Point Policy (described
564 in Section 3.3). MT- π proposes the use of a key point retargeting network in order to convert the
565 human hand and robot key points to the same space. Since we already convert the human hand key

Table 6: Number of demonstrations.

Task	Number of object instances	Total number of demonstrations
Close drawer	1	20
Put bread on plate	1	30
Fold towel	1	20
Close oven	1	20
Sweep broom	1	20
Put bottle on rack	2	30
Put bowl in oven	1	20
Make bottle upright	2	30

points to the corresponding robot points for Point Policy, we directly use these converted robot points instead of learning a separate keypoint retargeting network.

To ensure the correctness of our implementation, we evaluate MT- π in a setting identical to the one described in their paper. We conduct this evaluation on the *put bread on plate* task. We use 30 robot teleoperated demonstrations in addition to the human demonstrations, resulting in a total of 60 demonstrations. We observed a performance of 18/20, thus, confirming the correctness of the implementation.

D Experiments

D.1 Experimental Setup

Our experiments utilize a Franka Research 3 robot equipped with a Franka Hand gripper, operating in a real-world environment. We use the Deoxys [53] real-time controller for controlling the robot. The policies utilize RGB and RGB-D images captured using Intel RealSense D435 cameras from two third-person camera views. The action space encompasses the robot’s end effector pose and gripper state. We collect a total of 190 human demonstrations across 8 real-world tasks, featuring diverse object positions and types. Additionally, for studying the effect of co-training with robot data (Section D.4), we collect a total of 100 robot demonstrations for 4 tasks (Section D.4) using a VR-based teleoperation framework [74]. All demonstrations are recorded at a 20Hz frequency and subsequently subsampled to approximately 6Hz. For methods that directly predict robot actions, we employ absolute actions during training, with orientation represented using a 6D rotation representation [94]. This representation is chosen for its continuity and fast convergence properties. The learned policies are deployed at a 6Hz frequency during execution.

D.2 Task Descriptions

We experiment with manipulation tasks with significant variability in object position, type, and background context. Figure 3 depicts rollouts for all of our tasks. For each task, we collect data across various object sizes and appearances. During evaluations, we add novel object instances that are unseen during training. Illustrations of the variations in positions and object instances have been depicted in Appendix D.5. We provide a brief description along with details about the number of demonstrations and the evaluation setting for each task below.

Close drawer The robot arm is tasked with pushing close a drawer placed on the table. The position of the drawer varies for each evaluation. We collect 20 demonstrations for a single drawer and run evaluations on the same drawer.

Table 7: Policy performance of Point Policy with teleoperated robot data on in-domain object instances.

Demonstrations	Put bread on plate	Fold towel	Sweep broom	Make bottle upright
Human	19/20	9/10	9/10	16/20
Robot	18/20	9/10	4/10	12/20
Human + Robot	20/20	9/10	8/10	8/20

Put bread on plate The robot arm picks up a piece of bread from the table and places it on a plate. The positions of the bread and the plate are varied for each evaluation. We collect 30 demonstrations for the task of a single bread-plate pair. During evaluations, we introduce two new plates.

Fold towel The robot arm picks up a towel placed on the table from a corner and folds it. The position of the towel varies for each evaluation. We collect 20 demonstrations for a single towel. During evaluations, we introduce two new towels.

Close oven The robot arm is tasked with closing the door of an oven. The position of the oven varies for each evaluation. We collect 20 demonstrations for the task on a single oven and run evaluations on the same oven.

Sweep broom The robot arm picks up a broom and sweeps the table. The position and orientation of the broom are varied across evaluations. We collect 20 demonstrations for a single broom. During evaluations, we introduce a new broom.

Put bottle on rack The robot arm picks up a bottle from the table and places it on the lower level of a kitchen rack. The position of the bottle is varied for each evaluation. We collect 15 demonstrations for 2 different bottles, resulting in a total of 30 demonstrations for the task. During evaluations, we introduce three new bottles.

Put bowl in oven The robot arm picks up a bowl from the table and places it inside an oven. The position of the bowl varies for each evaluation. We collect 20 demonstrations for the task with a single bowl. During evaluations, we introduce a new bowl.

Make bottle upright The robot arm pick up a bottle from the table and places it in an upright position. The position of the bottle varies for each evaluation. We collect 15 demonstrations for 2 different bottles, resulting in a total of 30 demonstrations for the task. During evaluations, we introduce two new bottles.

D.3 Discussion of Failure Modes in Baselines

As shown in Table 1 (in-domain evaluation) and Table 2 (novel object instances), Point Policy substantially outperforms all baseline methods. We analyze their failure modes as follows:

Behavior Cloning (BC) & BC w/ Depth struggle due to a visual domain gap. During training, these methods observe human-hand images but receive robot-arm frames during inference. The morphological mismatch between human and robot end-effectors creates a distribution shift, impairing their ability to generalize.

MT- π [18] partially mitigates this issue by incorporating both scene images and 2D robot key points as input, demonstrating the value of key point representations (see Appendix D.7). However, its reliance on RGB images limits human-to-robot transfer compared to Point Policy.

P3PO [59] fails catastrophically (0% success) despite using key points. This stems from noisy depth estimates from RGB-D sensors: thin human fingers yield unreliable depth measurements, propagating errors to the 3D hand key points which in turn results in noisy hand poses used for action supervision. As detailed in Section 4.7, replacing sensor depth with triangulated depth boosts P3PO’s performance

634 to 72% success, yet it remains 18% below Point Policy. This gap highlights Point Policy’s advantage
 635 in leveraging 3D point track prediction rather than direct pose regression.

636 D.4 Can Point Policy be improved with robot demonstrations?

637 Table 7 investigates whether Point Policy’s performance can be enhanced through co-training with
 638 teleoperated robot data, collected using a VR-based teleoperation framework [74]. We conduct this
 639 study on four tasks - *put bread on plate*, *fold towel*, *sweep broom*, and *make bottle upright*. For each
 640 task, we collect an equal number of robot demonstrations as human demonstrations, resulting in 30,
 641 20, 20, and 30 demonstrations respectively. Interestingly, our findings reveal that for tasks involving
 642 complex motions, such as *sweep broom* and *make bottle upright*, policies trained solely on robot data
 643 perform poorly with the same amount of data as compared to those trained exclusively on human data.
 644 This drop in performance stems from the complex motions in these tasks making it harder to collect
 645 robot data using VR teleoperation, resulting in noisy demos. These results highlight an important
 646 consideration: humans and robots may execute the same task in different ways. Consequently,
 647 co-training with both human and robot data requires the development of algorithms capable of dealing
 648 with these differences effectively.

649 D.5 Illustration of Spatial Generalization and Novel Object Instances

650 Figure 6 and Figure 7 illustrate the variations in object positions and novel object instances used for
 651 each task, respectively.

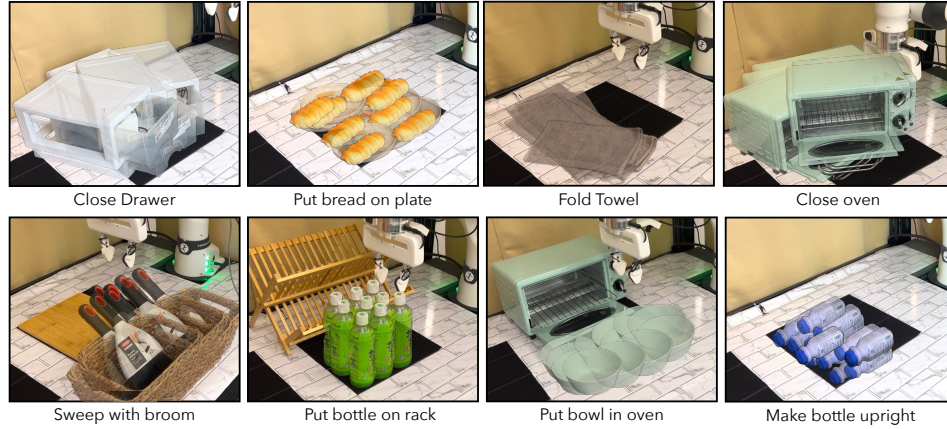


Figure 6: Illustration of spatial variation used in our experiments.



Figure 7: Illustration of objects used in our experiments. For each task, on the left are in-domain objects while on the right are novel objects used in our generalization experiments.

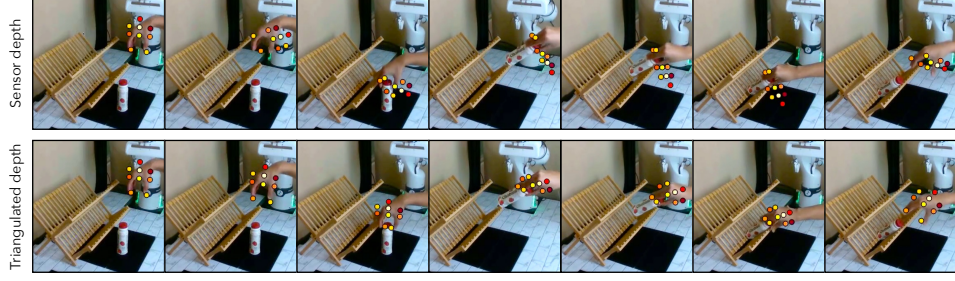


Figure 8: Illustration of discrepancy in actions obtained from sensor depth and triangulated depth for the task of putting a bottle on the rack.

Table 8: In-domain policy performance

Method	Close drawer	Put bread on plate	Fold towel	Close oven	Sweep broom	Put bottle on rack	Put bowl in oven	Make bottle upright
MT- π [18]	2/10	2/20	0/10	4/10	0/10	8/30	0/10	0/20
MT- π + object points	8/10	1/20	6/10	1/20	4/10	0/10	0/10	2/20
Point Policy (Ours)	10/10	19/20	9/10	9/10	9/10	26/30	8/10	16/20

D.6 Illustration of Depth Discrepancy

Figure 8 provides an illustration of the discrepancy in actions obtained from sensor depth and triangulated depth for the task of putting a bottle on the rack. We observe that the noise in sensor depth leads to noise in robot points which is turn results in unreliable actions.

D.7 Significance of Object Points

While Point Policy uses robot and object key points as input to the policy, MT- π [18], the best-performing baseline in Table 1, only uses robot key points and obtains information about the rest of the scene through an input image. We hypothesize that using object points can improve policy learning performance, especially when there is a morphology gap between data collection and inference. Table 8 and Table 9 test this hypothesis by providing object points in addition to the robot points already passed as input into MT- π , for in-domain objects and novel object instances, respectively. We observe that adding object points improves the performance of MT- π on select tasks. Nevertheless, Point Policy outperforms both methods by 68% across all tasks, emphasizing the efficacy of predicting 3D key points rather than 2D key points in image space.

E Discussion of Failure Modes and Future Directions

We recognize a few limitations in this work:

1. Point Policy’s reliance on existing vision models makes it susceptible to their failures. For instance, failures in hand pose detection or point tracking under occlusion have a detrimental effect on performance. However, with continued advances in computer vision, we believe that frameworks such as Point Policy will become stronger over time.
2. Point-based abstractions enhance generalization capabilities, but sacrifice valuable scene context information, which is crucial for navigating through cluttered or obstacle-rich environments. Future research focusing on developing algorithms that preserve sparse contextual cues in addition to the point abstractions in Point Policy might help address this.
3. While all our experiments are from a fixed third-person camera view, a large portion of human task videos on the internet are from an egocentric view [95, 96]. Extending Point Policy to egocentric

Table 9: Policy performance on novel object instances

Method	Put bread on plate	Fold towel	Sweep broom	Put bottle on rack	Put bowl in oven	Make bottle upright
MT- π [18]	1/20	0/20	0/10	0/30	0/10	0/20
MT- π + object points	2/20	0/20	0/20	1/10	0/10	1/20
Point Policy (Ours)	18/20	15/20	4/10	27/30	9/10	9/20

camera views can help us utilize these vast repositories of human videos readily available on the internet.

4. From Table 2, we observe a drop in performance for the *sweep broom* task for a new broom. This is because this novel object is shorter in length than the train object and hence, in some cases, the learned model either fails to grab the broom or fails to go down enough to reach the table. Including object instances of multiple sizes and shapes during training can help alleviate this problem.
5. While the key points are tracked through a trajectory, occlusion due to the human hand or the robot covering the object will result in failure of the vision models. We argue that image-based policies will face the same issue with occlusions and looking into other sensing modalities such as touch can present a potential solution for this.
6. In this work, we address the morphology gap between the human hand and the two-fingered robot gripper through appropriate hand-to-robot retargeting. Hence, the demonstrations are collected assuming a two-fingered gripper will be performing the task. We leave the transferring of arbitrary finger poses to a two-fingered gripper for future work.