Constraint-Preserving Data Generation for Visuomotor Policy Learning

Kevin Lin[1,2] **Varun Ragunath**[2*] **Andrew McAlinden**[2*] **Aaditya Prasad**[1]

Jimmy Wu[3] **Yuke Zhu**[2] **Jeannette Bohg**[1]

[1]Stanford University [2]University of Texas at Austin [3]Princeton University
* Denotes equal contribution

cp-gen.github.io

Abstract: Large-scale demonstration data has powered key breakthroughs in robot manipulation, but collecting that data remains costly and time-consuming. We present Constraint-Preserving Data Generation (CP-Gen), a method that uses a single expert trajectory to generate robot demonstrations containing novel object geometries and poses. These generated demonstrations are used to train closed-loop visuomotor policies that transfer zero-shot to the real world and generalize across variations in object geometries and poses. Similar to prior work using pose variations for data generation, CP-Gen first decomposes expert demonstrations into free-space motions and robot skills. But unlike those works, we achieve geometry-aware data generation by formulating robot skills as keypoint-trajectory constraints: keypoints on the robot or grasped object must track a reference trajectory defined relative to a task-relevant object. To generate a new demonstration, CP-Gen samples pose and geometry transforms for each task-relevant object, then applies these transforms to the object and its associated keypoints or keypoint trajectories. We optimize robot joint configurations so that the keypoints on the robot or grasped object track the transformed keypoint trajectory, and then motion plan a collision-free path to the first optimized joint configuration. Experiments on 16 simulation tasks and four real-world tasks, featuring multi-stage, non-prehensile and tight-tolerance manipulation, show that policies trained using our method achieve an average success rate of 77%, outperforming the best baseline which achieves an average of 50%.

1 Introduction

Teaching robots to act by imitating humans is one of the most powerful, yet costly, ideas in robotics. At its best, large-scale imitation learning has enabled robots to tie shoelaces, fix broken grippers, and even cook shrimp [1–4]. Further scaling promises to unlock broad generalization across complex manipulation tasks. Behind these successes, however, lies a steep cost: human labor, months of continuous robot operation, and heavy infrastructure demands. For example, ALOHA Unleashed [3] collected 26,000 demonstrations over 8 months across 10 robots and 35 operators, while DROID [5] gathered 76,000 real-world demonstrations over a year. As the field pushes forward, reducing the burden of demonstration collection will be a key challenge.

Automated data generation promises to reduce the manual effort required in data collection. One line of work [6–10] generates demonstrations for scenes with diverse object poses by collecting a handful of teleoperated demonstrations and algorithmically multiplying them using pose transformations and action replay. Conceptually, these works aim to enable data efficient spatial generalization by exploiting $\mathrm{SE}(3)$ equivariance of robot actions with respect to object poses [9]. However, because they rely on $\mathrm{SE}(3)$ transformations, these works cannot generate demonstrations that adapt to geometric variations (such as different aspect ratios) with a given object instance. For example, robot actions that succeed in hanging a short, wide wine glass onto a rack may fail entirely for a tall, narrow one, even if both are aligned under the same pose transformation.

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

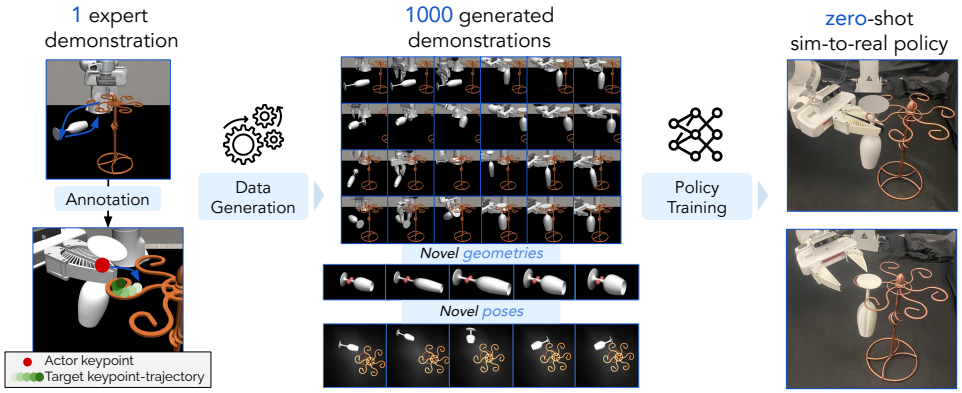


Figure 1: **CP-Gen** uses one expert demonstration and keypoint-trajectory constraints to generate diverse demonstrations in simulation involving novel object geometries and poses, enabling large-scale policy training and zero-shot sim-to-real transfer. For the Wine Glass Spiral Hanging task, the first keypoint-trajectory is anchored to the wine glass and constrains the motion of points on the end effector. The second keypoint-trajectory is anchored relative to the spiral-shaped wine glass rack, and constrains a keypoint on the wine glass stem to track a complex spiral motion.

A complementary strategy to ease the burden on manual data collection is to increase learning efficiency by embedding structure—specifically, equivariance—into policy architectures [11–23]. One line of work focuses on open-loop manipulation, exploiting symmetry to generalize across different object poses [24–27]. More recent work has moved toward closed-loop policy learning, embedding equivariances directly into the policy architecture and directly predicting robot actions [11, 28–32]. However, embedding equivariance bottlenecks can limit policy expressiveness and be insufficient when dealing with scenes that involve multiple interacting objects or symmetry-breaking variations.

In this work, we propose **Constraint-Preserving Data Generation (CP-Gen)**, a geometry-aware data generation method that takes as few as a *single* expert demonstration and generates new demonstrations with both novel object poses and novel geometries. With CP-Gen, we aim to achieve the benefits of equivariant policy architectures without the bottleneck of embedding structure directly into policies.

Our key insight is to bring geometry awareness into the data generation process through a **keypoint-trajectory constraint** formulation. Given an expert demonstration, we segment the trajectory into *free-space motions* (segments replaceable with point-to-point collision-free motion planning) and *robot skills* (segments requiring some interaction with task-relevant objects). We then formulate each skill as a keypoint-trajectory constraint: selected robot or grasped-object keypoints must track a reference trajectory defined in the frame of a task-relevant object. For example, in the *Wine Glass Spiral Hanging* task (Figure 1), grasping is formulated as a trajectory-tracking constraint between keypoints on the gripper tips and a target pre-grasp trajectory in the wine-glass frame. The *spiral* insertion introduces a second constraint: keypoints on the glass stem must track a spiral-like trajectory defined relative to the rack. By anchoring keypoints to the reference frame of task-relevant objects, we can sample new object geometries and poses, adapt the keypoint-trajectory constraints according to the transforms applied to the original object geometry, and generate thousands of demonstrations covering diverse shapes and geometries. Using these demonstrations, we can train a visuomotor policy that naturally generalizes robustly across changes in object pose and geometry, without requiring additional human demonstrations.

Our contributions are threefold. First, we propose CP-Gen, a data generation method that produces robot demonstrations from a single expert demonstration using a keypoint-trajectory constraint formulation. Unlike prior work, and due to this formulation, CP-Gen generates data in a geometry aware manner. Second, we validate our keypoint-trajectory data generation formulation on the MimicGen simulation benchmark. We achieve state-of-the-art performance (85% vs. 63%) on the default, pose-variation only task settings, as well as on our proposed simulation benchmark featuring novel object geometries (70% vs. 37%). Third, we demonstrate successful zero-shot sim-to-real on a set of four real-world tasks featuring multi-stage, non-prehensile and tight-tolerance manipulation.

2 Related Work

Data Generation for Robotics. One approach to addressing data scarcity in robotics is to generate new demonstrations from teleoperated source data [6–8, 10, 33]. MimicGen [6] generates demonstrations by resetting object poses and replaying pose-transformed versions of the source trajectory. IntervenGen [33] extends MimicGen with human interventions mid-trajectory to correct failures, while DexMimicGen [8] adapts MimicGen for bimanual and dexterous settings. SkillGen [7] segments demonstrations into skills and motions, alternating between learned policies and motion planning to improve robustness. At the core of these methods is a relative-pose formulation and the exploitation of $\mathrm{SE}(3)$-equivariance [9]: given an $\mathrm{SE}(3)$ transformation applied to an object, they apply the same $\mathrm{SE}(3)$ transformation to robot actions to replicate the original effect. As a result, they particularly struggle to generate demonstrations with new object geometries for tasks requiring tight-tolerance manipulation. CP-Gen addresses this limitation by introducing a keypoint-trajectory constraint formulation, representing skills in terms of robot or grasped-object keypoints tracking a reference trajectory. This formulation enables demonstration generation not only across novel object poses, but also across significantly different object geometries.

Equivariance for Robotic Manipulation. Another strategy to improve generalization is to embed geometric structure into policy inference and architecture. Open-loop methods design $\mathrm{SE}(2)$ or $\mathrm{SE}(3)$ equivariant feature representations on point clouds or images [11–23], achieving strong sample efficiency but often relying on expensive inference-time optimization [24–27] and lacking reactivity to dynamic changes. To address these limitations, recent work has moved toward closed-loop policies, beginning with planar $\mathrm{SO}(2)$ symmetries and extending to full $\mathrm{SIM}(3)$ equivariance [11, 28–32], where $\mathrm{SIM}(3)$ captures rotation, translation, and uniform scaling transformations. While $\mathrm{SIM}(3)$-equivariant policies enable more flexible closed-loop control compared to $\mathrm{SE}(3)$-equivariant policies, they still fundamentally assume that object geometry variations can be captured by uniform scaling, and for larger geometry variations, they rely on policy generalization. These closed-loop equivariance works [30–32] also tend to focus on single-object to robot equivariance, instead of the multi-object scenario. In CP-Gen, rather than using architectural equivariance, we leverage data generation to generalize to both novel poses and arbitrary geometry variations via a keypoint-trajectory constraint formulation.

3 Problem Setting

We are given a single expert demonstration in the form of a trajectory $\tau_{\text{src}} = \{o_t, a_t\}_{t=1}^{H}$ consisting of H observations o, and H actions a. Actions $a_t = \{a_{\text{eef}}, a_{\text{grip}}\}$ contain end-effector poses $a_{\text{eef}} \in \mathrm{SE}(3)$ along with gripper actions a_{grip} for a given manipulation task. Observations $o_t = \{I_{\text{hand}}, I_{3\text{pv}}, q_t\}$ at each time step consist of a wrist-mounted camera image I_{hand}, a third-person view camera image $I_{3\text{pv}}$, and robot proprioception data q_t. We aim to train a visuomotor policy $\pi_\theta(a|o)$ mapping observations $o \in O$ to actions $a \in A$.

Similar to prior data generation methods using source demonstrations [6, 7, 33], we assume that an expert demonstration τ_{src} can be decomposed into a sequence of alternating free-space motion segments $\tau^{(i)}_{\text{motion}}$ and robot skill segments $\tau^{(i)}_{\text{skill}}$: $\tau_{\text{src}} = [\tau^{(1)}_{\text{motion}}, \tau^{(1)}_{\text{skill}}, \tau^{(2)}_{\text{motion}}, \tau^{(2)}_{\text{skill}}, \ldots]$. For each skill segment, we additionally assume access to a set of annotated keypoints on the robot gripper or a grasped object that are relevant to the completion of the skill. We refer to these as *actor keypoints* and denote them by $\mathscr{K}_{\text{actor}}(t) = \{k_1^{\text{actor}}(t), \ldots, k_N^{\text{actor}}(t)\}$, where each keypoint $k_i^{\text{actor}}(t) \in \mathbb{R}^3$ is defined in the local frame of the robot gripper or the grasped object at time t. Although in this work we manually selected these keypoints, in future work, keypoints can be predicted using vision-language models [34–36], or inferred via task-specific keypoint detectors [37–40].

4 CP-Gen: Constraint-Preserving Data Generation

We present **Constraint-Preserving Data Generation (CP-Gen)**, a data generation method that enables one-shot visual imitation learning by adapting a source demonstration trajectory to scenes with novel object geometries and poses (see Figure 2). Our approach begins by decomposing the original demonstration trajectory τ_{src} into a sequence of skill segments τ_{skill} interleaved with collision-free motion segments τ_{motion}, such that $\tau_{\text{src}} = [\tau_{\text{motion}}, \tau_{\text{skill}}, \ldots, \tau_{\text{motion}}, \tau_{\text{skill}}]$. A skill segment is a segment that cannot be replaced by point-to-point collision-free motion planning, without affecting

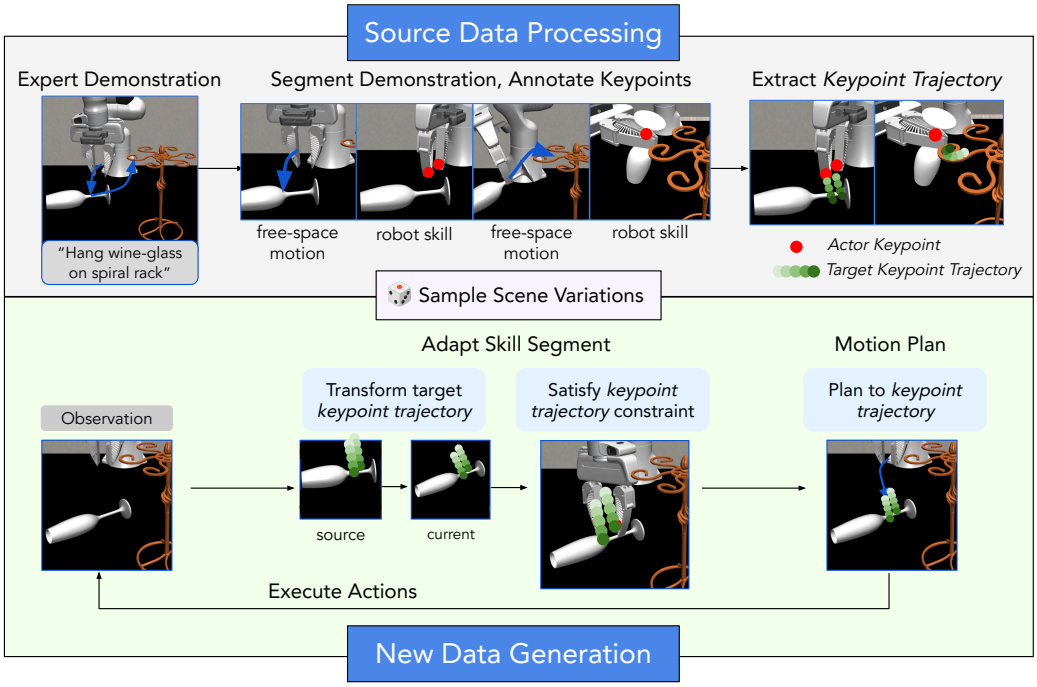


Figure 2: **CP-Gen Method**. In the *Source Data Processing* stage (top), starting from an expert demonstration τ_{src}, we (a) segment the trajectory into free-space motion and skill segments, (b) annotate keypoints on the robot or grasped object ● (dubbed *actor* keypoints), and (c) convert each skill segment into a *keypoint-trajectory constraint* by extracting a keypoint-trajectory expressed in the frame of a task-relevant object. After processing the source demonstration, in the *New Data Generation* stage (bottom), we 🎲 *sample scene variations* by applying geometry and pose transforms to every task-relevant object. For each sampled scene, we (a) adapt our source skill segment to the current observation by transforming the extracted target keypoint-trajectory with the current object transforms and solving for robot configurations that satisfy the updated keypoint-trajectory constraint, and (b) motion plan a collision-free path that from the current robot joint configuration to the first configuration in the solved configuration trajectory. The process iterates over all segments to generate a demonstration for the new scene, which can subsequently be used to train a visuomotor policy $\pi_\theta(a \mid o)$.

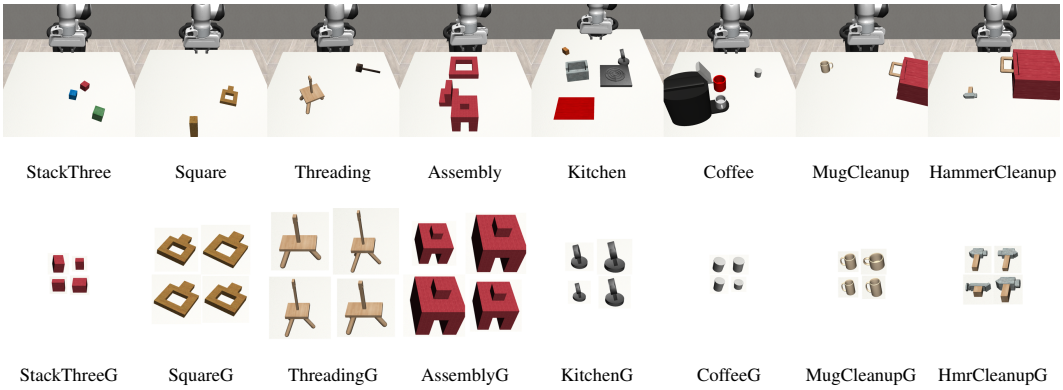
overall task success, while a motion segment is one that can be replaced by collision-free motion planning. Each skill segment τ_{skill} is formulated as a *keypoint-trajectory constraint*: keypoints on the robot or grasped object, *actor keypoints* ${}^{O}\mathcal{K}_{\text{actor}}(t)$, must track a reference trajectory defined relative to a task-relevant object. This reference trajectory is represented as a time-varying sequence of *target keypoints* ${}^{O}\mathcal{K}_{\text{target}}(t)$ expressed in the local frame of the object O. We extract this reference keypoint-trajectory (Section 4.1) by transforming the actor keypoints into the object frame at each timestep. We assume that adapting and preserving this keypoint-trajectory constraint to a new scene leads to successful execution. To generate a new demonstration in a novel scene, we first sample *pose* and *geometry* transforms for each task-relevant object. We then apply the sampled transformations to the objects (and their associated keypoint trajectories) to produce a new scene reset. For the new scene, we iterate through two phases of skill segment adaptation (Section 4.2) and motion planning (Section 4.3). Finally, after filtering out any failed demonstrations using a success detector (which we assume exists for each task), we use imitation learning to train a visuomotor policy on the generated demonstrations.

4.1 Keypoint-Trajectory Constraint Extraction

To extract the target keypoint-trajectory for a skill segment τ_{skill} from the original demonstration τ_{src}, we transform the actor keypoints ${}^{A}\mathcal{K}_{\text{actor}} = \{{}^{A}k_1, \ldots, {}^{A}k_N\}$, defined in the actor's local frame A (robot gripper or grasped object), into the task-relevant object frame O via a frame transform from actor to world frame, then world to task-relevant object frame:

$$ {}^{O}k_i(t) = \underbrace{{}^{O}T_W(t)}_{\text{object pose}^{-1}} \cdot \underbrace{{}^{W}T_A(t)}_{\text{actor pose}} \cdot {}^{A}k_i, \tag{1} $$

Applying Eq. 1 across all t yields target keypoint-trajectory: ${}^{O}\mathcal{K}_{\text{target}}(t) = \{{}^{O}k_1(t), \ldots, {}^{O}k_N(t)\}$.



Property	Stack Three	Square	Threading	Assembly	Kitchen	Coffee	Mug Cleanup	Hammer Cleanup
Object(s)	block1, block2, block3	nut, peg	needle, tripod	base, piece_1, piece_2	bread, pot, stove, button	pod, machine	drawer, object	hammer, drawer
Scale Range	[0.6,1.4]	[1.0,1.0]	[0.9,1.1]	[0.7,1.3]	[0.7,1.4]	[0.6,1.1]	[0.7,1.3]	[0.7,1.2]

Figure 3: **Simulation Tasks and Geometry Generalization Variants**. *Top*: Tasks from the MimicGen benchmark [6] and object geometries sampled from our proposed *Geometry Generalization* task variants. *Bottom*: Uniform scale sampling ranges applied in *Geometry Generalization* task variants.

4.2 Skill Segment Adaptation

Given a skill segment τ_{skill} and a current scene observation, our goal is to adapt the original keypoint-trajectory constraint to new task-relevant object geometries and poses. Doing so requires updating both the actor keypoints and the target keypoint-trajectory.

Update Keypoint-Trajectory Constraint for New Geometry. Suppose that the task-relevant object undergoes a geometric transformation $\mathbf{X}$ (e.g., non-uniform scaling) in its own local frame O. Let the original local-frame target keypoint-trajectory be ${}^{O}\mathscr{K}_{\text{target}}(t) = \{{}^{O}k_1(t), \ldots, {}^{O}k_N(t)\}$. We therefore apply $\mathbf{X}$ to transform the target keypoint-trajectory:

$$ {}^{O}k'_i(t) = \mathbf{X} \cdot {}^{O}k_i(t), \quad \forall t. $$

If the actor keypoints ${}^{A}\mathscr{K}_{\text{actor}}(t) = \{{}^{A}k_1(t), \ldots, {}^{A}k_N(t)\}$ are anchored to a grasped object that has been geometrically transformed, we similarly apply the object's local-frame geometric transformation $\mathbf{X}_A$ to the actor keypoints: ${}^{A}k'_i = \mathbf{X}_A \cdot {}^{A}k_i$. These updated keypoints ${}^{A}k'_i$ are then used in the optimization process to match the newly transformed target trajectory.

Solve for Robot Configuration via Keypoint Matching. Given the updated actor keypoints ${}^{A}k'_i$ (in the actor's local frame) and the updated target keypoint-trajectory ${}^{O}k'_i(t)$ (in the object's local frame), our goal is to solve for the robot joint configuration q^*_t at each timestep t such that the transformed actor keypoints match the target keypoints in the world frame.

First, we compute the world-frame position of each actor keypoint (recall that actor keypoints are defined on the robot or a grasped object, and that we assume a fixed end-effector to grasped object frame transformation) via forward kinematics: ${}^{W}k'_i(q) = f_{\text{FK}}(q, {}^{A}k'_i)$. Next, we compute the world-frame position of each target keypoint by transforming the updated local-frame keypoints using the current object pose ${}^{W}T_O(t)$ relative to the world frame: ${}^{W}k_i^{\text{target}}(t) = {}^{W}T_O(t) \cdot {}^{O}k'_i(t)$.

Finally, we solve the following optimization problem for each timestep of the robot skill segment:

$$ q^*_t = \arg\min_q \underbrace{\sum_{i=1}^{N} \left\| f_{\text{FK}}(q, {}^{A}k'_i) - \left({}^{W}T_O(t) \cdot {}^{O}k'_i(t)\right) \right\|_2^2}_{\text{match keypoints}} + \underbrace{\lambda \left\| q - q^*_{t-1} \right\|_2^2}_{\text{temporal smoothness penalty}} \tag{2} $$

where $\lambda \geq 0$ trades off keypoint matching and temporal smoothness in joint space. This optimization encourages a robot trajectory where the transformed actor keypoints ${}^{A}k'_i$ track the updated keypoint-

Method	StackThree	Square	Thread	Assembly	Kitchen	Coffee	Mug	Hammer	**Avg**
MimicGen [RGB]	**0.98±0.01**	0.72±0.04	0.38±0.04	0.32±0.04	**0.98±0.01**	0.58±0.04	0.78±0.03	0.64±0.04	0.67
DemoGen [RGB]	0.79±0.03	**0.87±0.03**	**0.87±0.03**	**0.85±0.03**	0.88±0.03	0.98±0.01	**0.89±0.03**	0.79±0.03	0.87
CP-Gen [RGB]	0.82±0.03	**0.87±0.03**	**0.87±0.03**	**0.85±0.03**	0.96±0.01	**1.00±0.00**	0.86±0.03	**0.80±0.03**	**0.88**
MimicGen [D+S]	0.52±0.04	0.76±0.03	0.44±0.04	0.40±0.04	**0.94±0.02**	0.56±0.04	0.46±0.04	0.64±0.04	0.59
DemoGen [D+S]	**0.56±0.04**	0.89±0.03	**0.85±0.03**	**0.80±0.03**	0.92±0.02	**1.00±0.00**	**0.81±0.03**	0.73±0.04	**0.82**
CP-Gen [D+S]	**0.56±0.04**	**0.91±0.02**	**0.85±0.03**	0.79±0.03	0.92±0.02	0.99±0.01	0.72±0.03	**0.79±0.03**	**0.82**

Method	StackThreeG	SquareG	ThreadG	AssemblyG	KitchenG	CoffeeG	MugG	HammerG	Avg
MimicGen [RGB]	0.60±0.04	0.42±0.04	0.10±0.02	0.16±0.03	0.74±0.04	0.00±0.00	0.38±0.04	0.40±0.04	0.35
DemoGen [RGB]	0.53±0.04	0.82±0.03	0.11±0.02	0.52±0.04	0.42±0.04	0.03±0.01	0.44±0.04	0.62±0.04	0.44
CP-Gen [RGB]	**0.68±0.04**	**0.88±0.03**	**0.79±0.03**	**0.55±0.04**	**0.83±0.03**	**0.58±0.04**	**0.82±0.03**	**0.67±0.04**	**0.73**
MimicGen [D+S]	0.22±0.03	0.64±0.04	0.05±0.02	0.38±0.04	0.84±0.03	0.00±0.00	0.50±0.04	0.45±0.04	0.39
DemoGen [D+S]	0.17±0.03	0.74±0.04	0.11±0.02	0.52±0.04	0.32±0.04	0.04±0.02	0.42±0.04	0.59±0.04	0.36
CP-Gen [D+S]	**0.36±0.00**	**0.83±0.03**	**0.73±0.04**	**0.79±0.03**	**0.86±0.03**	**0.63±0.04**	**0.76±0.04**	**0.74±0.04**	**0.67**

Table 1: **CP-Gen achieves state-of-the-art results on the MimicGen simulation benchmark**. **Top**: On the original MimicGen benchmark (default *Pose Only* task variants), CP-Gen and DemoGen perform similarly (average success rate of 85% and 84.5% respectively), compared to MimicGen's 63% which uses neither a motion planner nor object-object data generation. **Bottom**: On our custom benchmark containing *Geometry Generalization* task variants which feature novel object geometries (denoted as TaskG), CP-Gen achieves an average success rate of 70%, outperforming MimicGen and DemoGen by a margin of 33% and 30% respectively. These results highlight CP-Gen's strong generalization not only to pose variations but also to challenging geometric variations. Bolded numbers indicate the best-performing method within each modality group. RGB denotes policies taking RGB image inputs; D+S denotes policies taking depth maps and segmentation mask inputs.

trajectory ${}^{O}k'_i(t)$ in the world frame under new object geometries and object poses. We solve the optimization problem using the L-BFGS-B [41] gradient-based optimizer from SciPy [42].

4.3 Motion Planning

After obtaining the optimized robot configurations $Q^* = [q_1^*, \ldots, q_H^*]$ from the skill segment adaptation described above, we plan a collision-free trajectory from the robot's current joint configuration to the first joint configuration of the upcoming skill segment q_1^*. We query a collision-free motion planner [43, 44] with the robot's start configuration, the first joint configuration of the upcoming skill, and provide collision geometries extracted directly from our simulation environment. We use an inverse-kinematics controller to track trajectories in our end-effector pose action space.

5 Experiments

Through our experiments, we wish to answer the following questions:

Q1: Can a policy trained on data generated from a *single demonstration* using CP-Gen generalize to *unseen object geometries and poses*?

Q2: How does CP-Gen's *keypoint-trajectory formulation* affect *data generation* success rates on tasks with *novel object geometries*?

Q3: How does training policies on *diverse object geometries* affect performance in various settings?

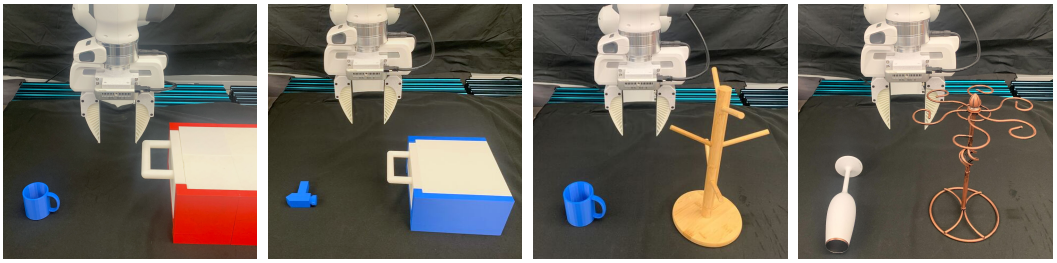
Q4: Can *pixel-based policies* trained with CP-Gen in simulation *transfer zero-shot* to the real world?

5.1 Simulation Experiments

Tasks. We evaluate all methods on single-arm Franka Panda tasks from the MimicGen [6] benchmark. We evaluate under two environment reset distributions. The default *Pose Only* distribution varies only object poses, and corresponds to distribution "D1" in the MimicGen benchmark. The custom *Geometry Generalization* distribution introduces shape variations via non-uniform scaling and retains the same pose reset range as the default *Pose Only* distribution (Figure 3).

Evaluation Metrics. We report the average success rate achieved by a particular policy network on the given task, along with the standard error of this metric. For computing standard errors, we use the best checkpoint rolled out for three seeds of 50 trials each.

Baselines. We compare CP-Gen to MimicGen [6] and DemoGen [10]. MimicGen is a data generation method that produces demonstrations in which object poses differ from those seen in source



(a) Mug Cleanup (b) Hammer Cleanup (c) Mug Hanging (d) Wine Glass Spiral Hanging

Figure 4: **Real World Tasks**. We evaluate on four challenging real-world tasks and show that policies trained on CP-Gen generated simulation datasets can transfer zero-shot to the real world.

demonstrations. MimicGen first parses each source demonstration into object-centric subtask segments. To generate a new demonstration, it selects a reference segment, transforms it to match the new scene's object pose, runs linear interpolation to reach the start of the transformed reference segment and executes the resulting end-effector trajectory using a controller. Finally, MimicGen filters out failed runs using a custom success checker, in the same way as CP-Gen.

The authors additionally add Gaussian action noise to increase state coverage at the cost of the number of successfully generated demonstrations. In our case, we use 0.01 noise by default, but set noise to 0 if the number of successfully generated demonstrations drops below 10% for a given task. We use MimicGen to generate 1000 successful demonstrations per task from 10 human demonstrations.

DemoGen generates synthetic demonstrations for novel object poses using a TAMP-based action generation method and 3D pointcloud editing. As the original setup does not leverage physics simulation, we adapt DemoGen to leverage a physics simulator for a fair comparison. Like CP-Gen, DemoGen assumes a fixed grasp transform between the end-effector and object if the previous robot skill segment involves grasping an object. DemoGen with physics can be viewed as CP-Gen without the keypoint trajectory constraints, and is implemented as such for our baseline.

Policy Training. For each method and task, we train a Diffusion Policy [45], specifically the DDPM variant, on the corresponding generated dataset that contains 1000 successful demonstrations. We use the same success classifier for CP-Gen, MimicGen and DemoGen to filter for successful demonstrations. Policy input consists of end-effector pose, gripper width, and observations from a third-person and wrist-mounted camera. For camera observations, we use either RGB images or a depth map with segmentation masks.

Results. From Table 1, we see that policies trained with data generated by CP-Gen (which takes a single source demonstration) outperform those trained using data generated using MimicGen and DemoGen across both observation input modalities. On *Pose Only* task variants, we see that CP-Gen outperforms policies trained using MimicGen data (85% vs. 63%). We hypothesize that the policy performance drop is due to dataset quality, and from two reasons. First, CP-Gen uses a collision-free motion planner, and thus contains collision-avoidance behaviors, while MimicGen data does not have explicit collision-avoidance behaviors. Second, CP-Gen uses *actor keypoints* (keypoints on the robot or grasped object) during data generation which enables the robot to adapt actions based on how an object is grasped, while MimicGen only preserves relative $SE(3)$ transforms between the robot and a reference object. For example, on the Threading task, regardless of how the robot grasps the needle, MimicGen would generate the same end-effector actions in the threading phase, based on the tripod's pose. In contrast, CP-Gen would adjust the robot's actions to align the actor keypoints on the needle tip with the tripod. In the *Pose Only* setting, CP-Gen is equivalent to our implementation of DemoGen and thus achieves similar success rates. On the *Geometry Generalization* task variants, CP-Gen's advantage is more pronounced (70% vs. 37% and 40%). This finding is unsurprising, as CP-Gen generated demonstrations adapted to changes in object geometries, while MimicGen and DemoGen do not consider such geometry changes during data generation.

Method	Mug Cleanup	Hammer Cleanup	Mug Hanging	Wine Glass Spiral Hanging	**Average**
MimicGen	0.20 ± 0.13	0.00 ± 0.00	0.80 ± 0.13	0.60 ± 0.16	0.40
CP-Gen	$\mathbf{0.80 \pm 0.13}$	$\mathbf{0.80 \pm 0.13}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.70 \pm 0.15}$	**0.83**

Table 2: **CP-Gen policies successfully transfer zero-shot sim2real**. Policies take in depth and segmentation masks, and tasks feature both object-geometry and object pose variations.

(a) Data Generation			(b) Policy Evaluation		
Method	**Pose**	**Geometry**	**Method**	**Pose**	**Geometry**
MimicGen	0.58	0.23	CP-Gen (Pose)	0.88	0.45
CP-Gen	0.89	0.62	CP-Gen (Geometry)	0.72	0.73

Table 3: **Ablation Results.** (a) CP-Gen outperforms MimicGen in data generation under both *pose*-only and *geometry* generalization settings. (b) Generating varied geometries (*geometry*) boosts policy generalization to novel objects (73% vs. 45%).

5.2 Real World Experiments

Tasks. We evaluate CP-Gen and baselines on four real-world manipulation tasks (Figure 4): (1) *Mug Cleanup*: Open a drawer, pick and place mug into a drawer, and close the drawer. (2) *Hammer Cleanup*: Same as *Mug Cleanup*, except with a small hammer, introducing grasping challenges. Cleanup tasks test multi-stage, non-prehensile manipulation involving articulated objects. (3) *Mug Hanging*: Pick and insert a mug onto a branch-like hook. This task tests constrained insertion through a small opening. (4) *Wine Glass Spiral Hanging*: Pick, reorient and hang a wine glass by threading the stem through a spiral-shaped rack. This task tests tight tolerance path following and orientation control. For each task, we manually construct a digital twin in simulation, generate 1000 successful demonstrations using a single source demonstration, and train a Diffusion Policy on the dataset. Policies are evaluated on two novel object geometries for each task.

Policy Inputs. We use end-effector poses, gripper width and two camera observations. Specifically, we use depth and segmentation masks for the third-person-view camera, and segmentation masks for the hand camera. We use depth and segmentation masks to reduce the sim-to-real gap [46].

Results. From Table 2, we see that CP-Gen achieves strong zero-shot sim-to-real transfer on all four manipulation tasks. In contrast, MimicGen struggles with novel geometries, as it generates fewer successful demos there. In the *Cleanup* tasks, failures arise from misalignment during drawer opening. In the *Hanging* tasks, failures are caused by suboptimal grasp positioning, which prevents proper insertion. Task failures did not come from collisions with non-relevant objects, so we posit that the performance gain came from the geometry-aware demonstration data generated by CP-Gen.

5.3 Ablations

We conduct a series of ablations to study two aspects of CP-Gen: the use of the keypoint-trajectory constraint formulation, and the effect of training on diverse object geometries. First, to assess the impact of the keypoint-trajectory constraint formulation, we compare data generation success rates between CP-Gen and MimicGen (which uses a relative pose data generation formulation) across both *Pose Only* and *Geometry Generalization* task variants on eight simulation tasks. For each task, we run 50 data generation trials. In Table 3(a), CP-Gen achieves consistently higher success rates (89% vs. 58%, 62% vs. 23%), especially on the *Geometry Generalization* reset distribution, highlighting the value of the geometry-aware data generation that our keypoint-trajectory constraint formulation enables. Note that our comparison with DemoGen [10] in Table 1 also serves as an ablation for using keypoint trajectory constraints versus only using pose transforms, as that is the difference between CP-Gen and our DemoGen baseline. Keypoint-trajectory constraints lead to a 30% increase in policy success rates for CP-Gen vs. DemoGen in the Geometry Generalization setting. Second, we evaluate how training on diverse object geometries affects policy generalization. In Table 3(b), we report policy success rates when training with and without geometry diversity, and evaluated on the *Pose* and *Geometry Generalization* environment reset settings. As expected, training with diverse object geometries substantially improves policy generalization to novel geometries (73% vs. 45%). We also observe that training and evaluating on the same distribution yields the best evaluation performance (73% vs. 45% and 88% vs. 72%).

6 Conclusion

In this work, we propose CP-Gen, a data generation framework for generalizable visual imitation learning given just one expert demonstration. CP-Gen generates geometry-aware demonstrations using the insight that robot skills can be formulated as keypoint-trajectory constraints: keypoints on the robot or grasped object must track a reference trajectory defined relative to a task-relevant object. We demonstrate that CP-Gen's approach to data generation from just one source demonstration achieves state-of-the-art success rates on the MimicGen simulation benchmark, and outperforms baselines on a custom benchmark featuring diverse object geometries. Finally, we show successful zero-shot sim-to-real transfer of policies trained with CP-Gen data to challenging real-world tasks.

Limitations. CP-Gen requires manual annotation of robot skill segments and keypoints. The simulation environment and reward functions are also manually defined; future work could incorporate foundation model based success classifiers. The method assumes a fixed skill sequence, limiting task level generalization. Incorporating task level planning via foundation models [47–50] or task and motion planner methods [51–53] can further improve data generation success. Current results are limited to a single-arm Panda robot, though the method can be extended to multi-arm, mobile base, and legged systems [8]. CP-Gen does not handle the situation where one may want demonstrations generated for a specific object instance within the same category for which a source demonstration is available. For example, we may have a source demonstration for a mug with a square handle, but wish to generate demonstrations for a specific mug that might have a round handle. CP-Gen would need require the geometric transformation from the square handle mug to the round handle mug, which may be non-trivial to obtain. Future work may explore the use of neural descriptor fields [25] to assist in this setting.

Acknowledgments

Toyota Research Institute provided funds to support this work. Additionally, this work was partially supported by the National Science Foundation (FRR-2145283, EFRI-2318065), the Office of Naval Research (N00014-24-1-2550), the DARPA TIAMAT program (HR0011-24-9-0428), and the Army Research Lab (W911NF-25-1- 0065). It was also supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government (MSIT) (No. RS-2024-00457882, National AI Research Lab Project). Finally, we thank Timothy Chen, Wil Thomason, Zak Kingston, Tyler Lum, Priya Sundaresan, Fanyun Sun, Yifeng Zhu, Zhenyu Jiang, Mingyo Seo, Megan Hu, William Chong, Marion Lepert, and Brent Yi for helpful discussions throughout the project.

References

[1] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi:10.15607/RSS.2023.XIX.016.

[2] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning (CoRL)*, 2024.

[3] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, K. Ghasemipour, C. Finn, and A. Wahid. Aloha unleashed: A simple recipe for robot dexterity. *arXiv preprint arXiv:2410.13126*, 2024.

[4] J. Aldaco, T. Armstrong, R. Baruch, J. Bingham, S. Chan, K. Draper, D. Dwibedi, C. Finn, P. Florence, S. Goodrich, et al. Aloha 2: An enhanced low-cost hardware for bimanual teleoperation. *arXiv preprint arXiv:2405.02292*, 2024.

[5] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

[6] 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.

[7] C. R. Garrett, A. Mandlekar, B. Wen, and D. Fox. Skillgen: Automated demonstration generation for efficient skill learning and deployment. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=YOFrRTDC6d.

[8] Z. Jiang, Y. Xie, K. Lin, Z. Xu, W. Wan, A. Mandlekar, L. Fan, and Y. Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[9] E. Ameperosa, J. A. Collins, M. Jain, and A. Garg. Rocoda: Counterfactual data augmentation for data-efficient robot learning from demonstrations. *arXiv preprint arXiv:2411.16959*, 2024.

[10] Z. Xue, S. Deng, Z. Chen, Y. Wang, Z. Yuan, and H. Xu. Demogen: Synthetic demonstration generation for data-efficient visuomotor policy learning. *arXiv preprint arXiv:2502.16932*, 2025.

[11] D. Wang, R. Walters, and R. Platt. SO(2)-Equivariant Reinforcement Learning. In *International Conference on Learning Representations*, 2022.

[12] D. Wang, R. Walters, X. Zhu, and R. Platt. Equivariant Q Learning in Spatial Action Spaces. In *5th Annual Conference on Robot Learning*, 2021.

[13] H. Huang, D. Wang, A. Tangri, R. Walters, and R. Platt. Leveraging Symmetries in Pick and Place. *The International Journal of Robotics Research*, 2023.

[14] 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 *Conference on Robot Learning*, pages 835–846. PMLR, 2023.

[15] C. Pan, B. Okorn, H. Zhang, B. Eisner, and D. Held. TAX-Pose: Task-Specific Cross-Pose Estimation for Robot Manipulation. In *Conference on Robot Learning*, pages 1783–1792. PMLR, 2023.

[16] H. Huang, D. Wang, X. Zhu, R. Walters, and R. Platt. Edge Grasp Network: A Graph-Based SE(3)-invariant Approach to Grasp Detection. In *International Conference on Robotics and Automation (ICRA)*, 2023.

[17] S. Liu, M. Xu, P. Huang, X. Zhang, Y. Liu, K. Oguchi, and D. Zhao. Continual Vision-based Reinforcement Learning with Group Symmetries. In *Conference on Robot Learning*, pages 222–240. PMLR, 2023.

[18] M. Jia, D. Wang, G. Su, D. Klee, X. Zhu, R. Walters, and R. Platt. SEIL: Simulation-augmented Equivariant Imitation Learning. In *International Conference on Robotics and Automation (ICRA)*, 2023.

[19] S. Kim, B. Lim, Y. Lee, and F. C. Park. Se (2)-equivariant pushing dynamics models for tabletop object manipulations. In *Conference on Robot Learning*, pages 427–436. PMLR, 2023.

[20] C. Kohler, A. S. Srikanth, E. Arora, and R. Platt. Symmetric models for visual force policy learning. *arXiv preprint arXiv:2308.14670*, 2023.

[21] H. H. Nguyen, A. Baisero, D. Klee, D. Wang, R. Platt, and C. Amato. Equivariant reinforcement learning under partial observability. In *Conference on Robot Learning*, pages 3309–3320. PMLR, 2023.

[22] H. Nguyen, T. Kozuno, C. C. Beltran-Hernandez, and M. Hamaya. Symmetry-aware reinforcement learning for robotic assembly under partial observability with a soft wrist. *arXiv preprint arXiv:2402.18002*, 2024.

[23] 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. URL https://openreview.net/forum?id=2inBuwTyL2.

[24] H. Huang, D. Wang, R. Walters, and R. Platt. Equivariant Transporter Network. In *Robotics: Science and Systems*, 2022.

[25] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. In *ICRA*, 2022.

[26] H. Ryu, H. in Lee, J.-H. Lee, and J. Choi. Equivariant Descriptor Fields: SE(3)-Equivariant Energy-Based Models for End-to-End Visual Robotic Manipulation Learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[27] H. Huang, O. Howell, X. Zhu, D. Wang, R. Walters, and R. Platt. Fourier Transporter: Bi-Equivariant Robotic Manipulation in 3D. *arXiv preprint arXiv:2401.12046*, 2024.

[28] X. Zhu, D. Wang, O. Biza, G. Su, R. Walters, and R. Platt. Sample Efficient Grasp Learning Using Equivariant Models. In *Robotics: Science and Systems*, 2022.

[29] X. Zhu, D. Wang, G. Su, O. Biza, R. Walters, and R. Platt. On robot grasp learning using equivariant models. *Autonomous Robots*, 47(8):1175–1193, 2023.

[30] D. Wang, S. Hart, D. Surovik, T. Kelestemur, H. Huang, H. Zhao, M. Yeatman, J. Wang, R. Walters, and R. Platt. Equivariant diffusion policy. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=wD2kUVLT1g.

[31] J. Yang, Z.-a. Cao, C. Deng, R. Antonova, S. Song, and J. Bohg. Equibot: Sim(3)-equivariant diffusion policy for generalizable and data efficient learning. In *8th Annual Conference on Robot Learning*, 2024.

[32] J. Yang, C. Deng, J. Wu, R. Antonova, L. Guibas, and J. Bohg. Equivact: Sim(3)-equivariant visuomotor policies beyond rigid object manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9249–9255. IEEE, 2024.

[33] R. Hoque, A. Mandlekar, C. Garrett, K. Goldberg, and D. Fox. Intervengen: Interventional data generation for robust and data-efficient robot imitation learning, 2024.

[34] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=9iG3SEbMnL.

[35] 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.

[36] OpenAI. Gpt-4 technical report, 2023.

[37] L. Manuelli, W. Gao, P. Florence, and R. Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *ISRR*, 2019.

[38] Z. Qin, K. Fang, Y. Zhu, F.-F. Li, and S. Savarese. Keto: Learning keypoint representations for tool manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 7278–7285. IEEE, 2020.

[39] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, M. Laskey, K. Stone, J. E. Gonzalez, and K. Goldberg. Learning rope manipulation policies using dense object descriptors trained on synthetic depth data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 9411–9418. IEEE, 2020.

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

[41] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995. doi:10.1137/0916069. URL https://doi.org/10.1137/0916069.

[42] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi:10.1038/s41592-019-0686-2.

[43] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. V. Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, and D. Fox. curobo: Parallelized collision-free minimum-jerk robot motion generation, 2023.

[44] W. Thomason, Z. Kingston, and L. E. Kavraki. Motions in microseconds via vectorized sampling-based planning. In *IEEE International Conference on Robotics and Automation*. URL http://arxiv.org/abs/2309.14545.

[45] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[46] M. Dalal, M. Liu, W. Talbott, C. Chen, D. Pathak, J. Zhang, and R. Salakhutdinov. Local policies enable zero-shot long-horizon manipulation. *International Conference of Robotics and Automation*, 2025.

[47] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[48] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[49] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg. Text2motion: From natural language instructions to feasible plans. *Autonomous Robots*, 47(8):1345–1365, 2023.

[50] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

[51] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pages 1470–1477, 2011. doi:10.1109/ICRA.2011.5980391.

[52] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 440–448, 2020.

[53] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021.

Supplementary Material for: Constraint-Preserving Data Generation for Visuomotor Policy Generalization

1 Method Details

1.1 Source Demonstration Collection

For all CP-Gen data generation, we collect one expert demonstration per task using a custom teleoperation system based on the Gello framework [1]. We use the Gello 'leader' arm to control the end-effector pose of the Franka Emika Panda robot in simulation. The underlying controller is a Whole-Body Inverse Kinematics controller provided by Robosuite [2].

1.2 Keypoint Annotation

We annotate "actor keypoints"—task-relevant 3D points on the gripper and/or the manipulated object—to extract keypoint-trajectory constraints that enable geometry aware data generation. When the actor keypoints are located on the gripper (e.g., fingertips or center of grasp), we reuse the same set of predefined gripper keypoints across all environments. When the keypoints are located on the object, we use the MuJoCo [3] viewer interface to select 3D keypoints on the task relevant object's geometry. As described in the main text, though we opt for manual annotation of keypoints in this paper, automatic methods are also possible [4].

1.3 Data Generation Noise Details

To improve robustness, we augment expert demonstrations with *system noise*. Starting from a successful demonstration, we inject Gaussian noise into each action, where actions are parameterized as absolute end-effector commands with respect to the robot base. Given state s and original action a, we execute in the environment a perturbed action

$$a' = a + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2 I),$$

while still recording the original a for training. This setup differs from prior data generation approaches (e.g., MimicGen [5, 6]) that directly train on noised actions a', which we found to produce jerkier motions.

We initialize σ at a high noise level and iteratively decrease it until the rollout completes successfully. For each source demonstration, we retain the successful trajectory with the highest feasible noise level. By decoupling execution (a') from training supervision (a), the system explores a wider range of perturbed states while maintaining smoother action labels.

1.4 Policy Training Details

For both simulation and real-world experiments, we leverage the DDPM implementation from the Diffusion Policy [7] paper. We use the U-Net architecture conditioned on a sequence of RGB images to predict future actions. We adopt a fixed diffusion horizon and use default policy hyperparameters across all tasks unless otherwise stated. Key training settings are summarized in Table 1.

For simulation tasks that include depth and segmentation inputs, we use 84×84 image resolutions and provide both depth and segmentation masks from both robot arm cameras. To mimic the real-world setup, we do not use depth from the hand-mounted camera.

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

Table 1: Training details for diffusion policy.

Category	Detail
Horizon	16 steps (2 observation steps, 8 action steps)
Backbone	ResNet-18 (GroupNorm, no pretrained weights)
Crop Size	76×76 (random crop from fixed center crop)
Diffusion Steps	100 (DDPM with `squaredcos_cap_v2` schedule)
U-Net Config	Downsampling dimensions: [512, 1024, 2048]; kernel size: 5; 8 GroupNorm groups
Optimizer	AdamW (learning rate 1e$-$4, betas = [0.95, 0.999], weight decay 1e$-$6)
Batch Size	64
Learning Rate Schedule	Cosine decay with 500-step warmup
EMA	Enabled (inv_gamma = 1.0, power = 0.75)

For real-world experiments, raw observations are first resized to 90×160, followed by a center crop to 90×90. We then apply the same random cropping procedure as in simulation to produce the final 76×76 input.

2 Simulation Experiment Details

We evaluate CP-Gen on eight simulated manipulation tasks introduced in MimicGen [5], covering short and long-horizon behaviors such as stacking, insertion, drawer manipulation, and object assembly. All tasks are executed in MuJoCo [3] using the Robosuite [2] framework with an inverse kinematics controller at 20 Hz. The action space is an absolute end-effector pose and gripper joint.

Default Task Setting. For all tasks, we use the MimicGen-defined D_1 variant, which introduces modest spatial variation in object placement and top-down rotation. Below we describe each of the eight MimicGen [5] tasks used in our study:

- **Stack Three**: The robot must sequentially stack a red block onto a green block and a blue block onto the red one. There are 4 subtasks in total.
- **Square**: The robot picks up a square nut and places it onto a peg. There are 2 subtasks: grasping the nut and inserting it onto the peg.
- **Threading**: The robot must grasp a needle and thread it through a tripod hole. The task requires precise control and involves 2 subtasks.
- **Coffee**: The robot picks up a coffee pod, inserts it into a coffee machine, and closes the hinge. This task involves 2 subtasks.
- **Three Piece Assembly**: The robot picks and inserts two separate pieces into a base structure to assemble a composite object. The task contains 4 subtasks.
- **Hammer Cleanup**: The robot opens a drawer, picks up a hammer, places it inside the drawer, and closes the drawer. There are 4 subtasks in total.
- **Mug Cleanup**: Similar to Hammer Cleanup, but with a mug object. The task includes object-centric variations involving different mug geometries. It contains 4 subtasks.
- **Kitchen**: The robot must switch the stove on, pick and place a pot onto the stove, pick and place bread into the pot, move the pot to the serving region, and then turn the stove off. This long-horizon task involves 7 sequential subtasks.

Geometry Generalization Benchmark. To evaluate the robustness of CP-Gen to changes in object geometry, we introduce a new benchmark involving geometry generalization. Specifically, we keep the same pose reset distributions from the default tasks settings, but apply non-uniform scaling transforms to task relevant objects. We show samples scene resets for our custom benchmark in Figure 1.

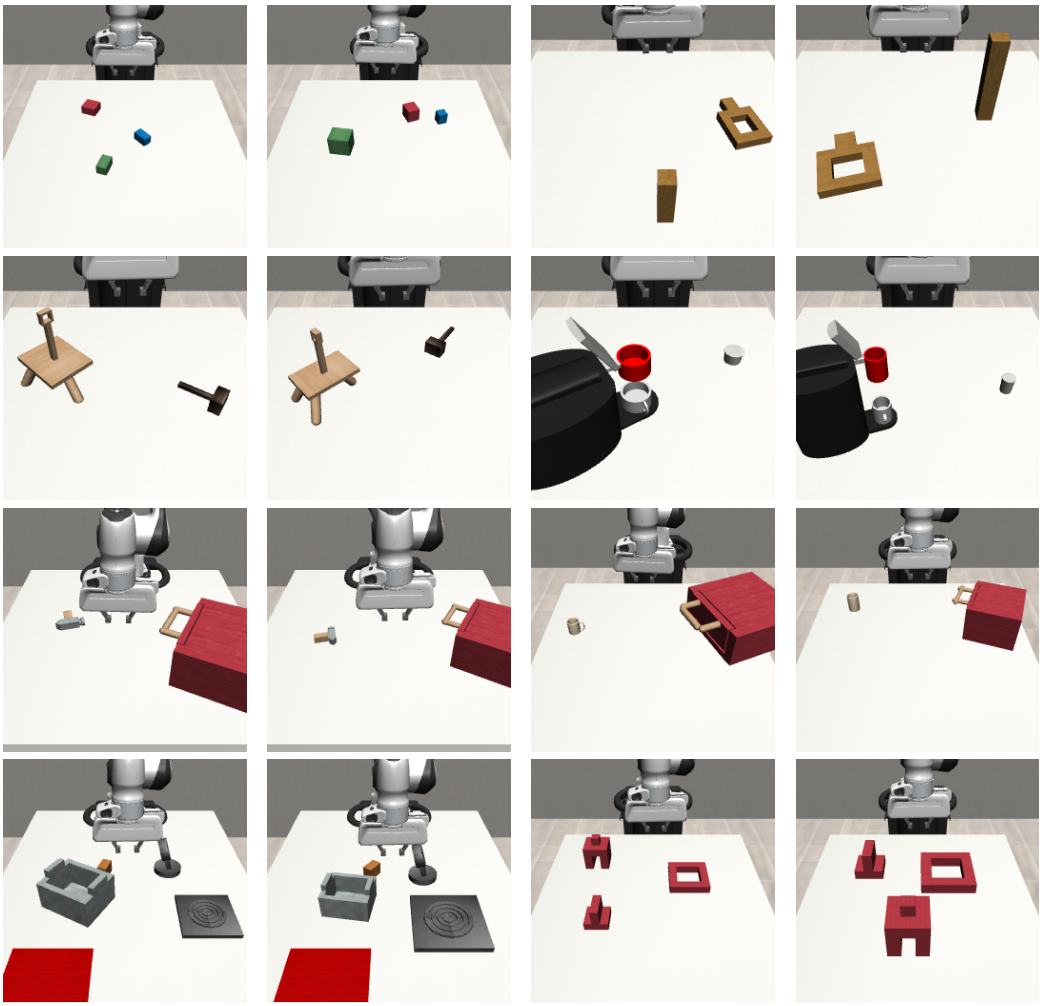


Figure 1: **Examples of Geometry Generalization.** For each task, we show two examples of scale factors applied to each axis on the task-relevant objects.

3 Real-World Experiment Details

Real-World Setup. For our real-world robot, we use a Franka Emika Panda arm, with a UMI gripper [8] attached to the panda gripper. For our cameras, we use two Intel RealSense cameras: a 3rd-person view (3PV) camera statically mounted, and an eye-in-hand (hand) camera mounted to the wrist.

Digital Twin Visualizations. For each real-world task, we construct a simulation-based digital twin environment. Figure 2 shows RGB, depth map and binary segmentation mask renderings for all four real-world tasks, from both third-person and eye-in-hand perspectives.

Segmentation Masks. To obtain segmentation masks in the real world, we first use the Segment Anything Model (SAM) [9] to generate binary segmentation masks. Then, we apply adaptive color thresholding using Otsu's method [10] within the selected SAM masks to isolate foreground objects from the black table background.

References

[1] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel, *Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators*, 2023.

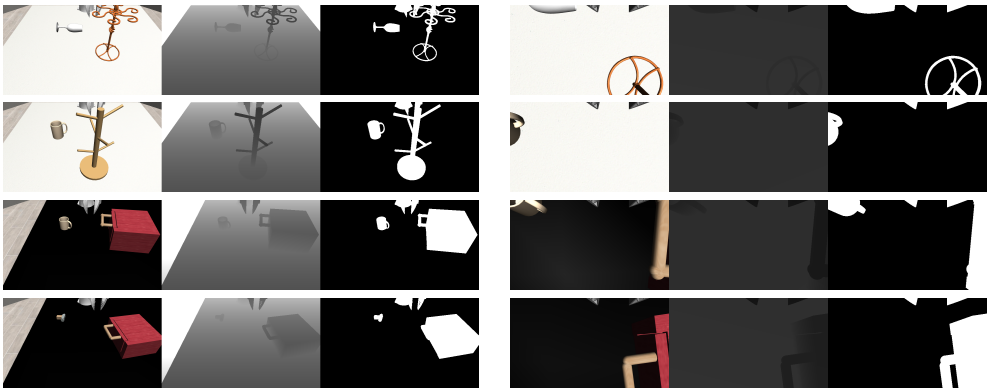


Figure 2: Digital twin environments for each real-world task. Left: third-person agentview. Right: eye-in-hand view.

[2] Y. Zhu *et al.*, "Robosuite: A modular simulation framework and benchmark for robot learning," in *arXiv preprint arXiv:2009.12293*, 2020.

[3] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[4] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, "Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation," in *8th Annual Conference on Robot Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=9iG3SEbMnL.

[5] A. Mandlekar *et al.*, "Mimicgen: A data generation system for scalable robot learning using human demonstrations," in *7th Annual Conference on Robot Learning*, 2023.

[6] Z. Jiang *et al.*, "Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[7] C. Chi *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[8] C. Chi *et al.*, "Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots," in *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[9] N. Ravi *et al.*, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024. [Online]. Available: https://arxiv.org/abs/2408.00714.

[10] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979. DOI: 10.1109/TSMC.1979.4310076.