

## B FAQ

### 1. What are some limitations of SkillGen?

See Appendix C.

### 2. Why might a data generation attempt in a failure?

The transformed human segments (skill segments) during data generation (Sec. 4.4) might result in poses that are difficult or impossible for the motion planner or task-space controller to reach. Small errors can also accumulate during open-loop replay of the skill segments, causing failures during high-precision motions such as insertion. Despite the potential for these failures, proficient agents can be trained from SkillGen datasets.

### 3. Are there concrete examples of situations where SkillGen succeeds in generating data but MimicGen fails?

See Appendix D.

### 4. Is SkillGen compatible with normal teleoperation systems or do I have to use HITL-TAMP?

Yes, SkillGen is compatible with normal teleoperation systems – see Appendix L for results and discussion.

### 5. What are the assumptions made by each HSP policy learning method?

HSP-Reg makes no additional assumptions compared to standard Behavioral Cloning methods. HSP-Class makes similar assumptions to those made during data generation – namely that the sequence of relevant objects that the robot must interact with for a task are known, and we are able to observe or estimate object poses prior to robot interaction (Sec. 3, A2 and A3). Importantly, this does not require full object pose tracking. HSP-TAMP [13] makes the most assumptions. It assumes access to a TAMP system that knows where to move the robot before initiating the learned skill policy and when to terminate the learned skill policy.

### 6. There is a small but significant performance gap between HSP-Reg, and the other HSP methods. Does that mean that policies must use privileged information to get the benefits of the HSP skill formulation?

The results are close between HSP-Reg and the other methods in many cases (Fig. 4, average success rate only lower by 10% to 13%) despite making much fewer assumptions (see FAQ (5) above). However, there are some easy ways to improve performance further (discussion in Appendix Q), including generating more data (Appendix E). Moreover, HSP-Reg might be the only method appropriate for tasks in which, for example, the objects vary.

### 7. Is it necessary for SkillGen data generation rates to be high for policies trained on the generated demo to perform well? If not, why is it beneficial to have higher data generation rates?

There isn't a strict correlation between data generation success rate and trained policy success rate. In many cases, data generation success rates can be very low, especially when using initiation augmentation (Appendix F), compared to the resulting policy success rates. However, higher data generation rates can be beneficial for generating datasets more quickly (in terms of wall clock time), since it will take less time to reach a target amount of data. Even when data generation rates are low, SkillGen can leverage parallelization during data generation to generate data faster (Appendix G.4). Finally, a higher data generation rate can imply better coverage of the task reset distribution in the generated data, but a low data generation rate does not necessarily mean the task reset distribution is not covered well.

### 8. Can SkillGen be used to generate data for different robot arms, like MimicGen?

Yes, see Appendix N for results.

### 9. Explain how SkillGen was used to generate over 24K demonstrations across 18 task variants in simulation from just 60 human demonstrations.

We generated 1000 SkillGen demos for each of the 18 task variants in Fig. 4 and an additional 6 more datasets (1000 demos each) with a different robot arm (Appendix N), using just 10 source human demos collected on the 6 simulation tasks. We do not include the

670 dataset scaling law experiments (Appendix [E](#)), the datasets generated with initiation aug-  
671 mentation, and the datasets generated in the real world, which would increase the total  
672 substantially.

## C Limitations

We discuss limitations of SkillGen that can inform future work, extending Section 7.

1. **Given sequence of skill segments during data generation.** During data generation, the sequence of skill segments (relevant objects that must be manipulated by the robot during each skill) must be provided.
2. **Object pose estimates during data generation.** During data generation, SkillGen assumes access to the object pose at the start of each skill segment, either by direct observation (simulation) or estimation (real world).
3. **Quasi-static tasks with rigid objects.** This paper applies SkillGen to primarily quasi-static tasks with rigid objects.
4. **Better performance when using source human data from HITL-TAMP [13] than from conventional teleoperation systems.** SkillGen obtains better results when using human demonstrations collected with HITL-TAMP than with conventional teleoperation systems (Appendix L). Investigating how more consistent human annotations can reduce this gap is future work.
5. **Limited agent observability and action space for sim-to-real experiments.** Agents used in the sim-to-real experiments only observe changes in robot proprioception, as no pose tracking or visual observations are used during execution. The agent also receives object poses at the start of each episode, but these are never updated. The action space is restricted to position-only control (no rotation). These design choices were made to maximize the possibility of transfer without the need for addressing the gap in perception between simulation and the real world, and without the need for extensive robot controller tuning between simulation and the real world. See Appendix K for more details and discussion.

## D Analysis on Challenging Data Generation Scenarios

In this section, we discuss some challenging data generation scenarios where SkillGen is able to generate data, while MimicGen struggles. We first review some limitations of MimicGen, and then we discuss different data generation scenarios.

### D.1 MimicGen Limitations

**Susceptibility to scene collisions.** MimicGen uses a naive linear interpolation scheme during data generation to connect the end of one transformed object-centric human segment to another one. This approach is not aware of scene geometry, which can result in data generation failures due to collisions between the robot and other objects in the scene. By contrast, SkillGen transit and transfer motions between skill segments are carried out via motion planning.

**Tradeoff between Data Generation Quality and Policy Learning Proficiency.** The use of naive linear interpolation also impacts learning ability. Longer in time (not space) interpolation segments have been shown to be harmful to policies trained from MimicGen data [11], which motivates the use of short interpolation segments with a small number of intermediate waypoints. However, this can lower the data generation success rate, since the end-effector controller might not be capable of accurately tracking waypoints that are far apart, and this also can be unsafe for real-world deployment. Consequently, MimicGen has a fundamental tradeoff with respect to interpolation segments. On one hand, shorter segments are better for policy learning but can result in lower data generation success rates and be unsafe for real-world deployment. On the other, longer segments are more suitable for real-world deployment and for ensuring better data generation throughput but make policy learning more difficult. By contrast, SkillGen has no such tradeoff.

### D.2 Challenging Data Generation Scenarios

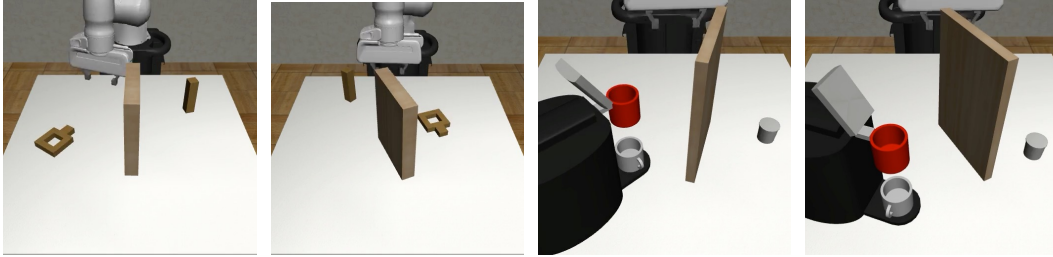


Figure D.1: **Example Configurations for Clutter Tasks.** Example configurations from the clutter task variants of Square and Coffee.

**Presence of Clutter.** SkillGen successfully generates data for scenes with large obstacles, unlike MimicGen. We develop variants of the Square and Coffee tasks that have a large obstruction placed in the workspace (Fig. D.1). The reset distributions for these tasks are identical to their clutter-free counterparts described in Appendix J except for the presence of the obstruction, which has its own reset distribution, and is placed randomly near the center of the workspace. We use the same source demonstrations as before (collected on the clutter-free  $D_0$  variants of these tasks) and perform 200 data generation attempts with both SkillGen and MimicGen. The data generation success rates are presented in Table D.1. We see that SkillGen substantially outperforms MimicGen by margins as large as 58.5%.

Task Variant	MimicGen [11]	SkillGen
Square ( $D_1$ , Clutter)	4.0	<b>62.5</b>
Square ( $D_2$ , Clutter)	14.5	<b>72.0</b>
Coffee ( $D_0$ , Clutter)	16.5	<b>49.0</b>
Coffee ( $D_1$ , Clutter)	14.0	<b>55.0</b>

Table D.1: **Data Generation Rates for Environments with Clutter.** SkillGen is able to generate data for environments with clutter much more effectively than MimicGen.

727 **Large Scene Variations from Source Demos.** SkillGen excels at generating data even when there  
 728 are substantial deviations from where objects were located in the source human demonstrations  
 729 unlike MimicGen, which suffers from having to use short linear interpolation segments during gen-  
 730 eration. For example, MimicGen is unable to produce any data on Coffee Prep  $D_2$ , due to the mug  
 731 and drawer being on opposite sides of the table compared to the source demos ( $D_0$ ) (see Fig. D.2),  
 732 while SkillGen can generate data and train proficient agents on  $D_2$  (Fig. 4). SkillGen also enjoys  
 733 large gains over MimicGen for data generation rates, especially on  $D_2$  task variants, which vary sub-  
 734 stantially from  $D_0$ , where source data was collected. This can be seen in Table F.1 (Appendix F).

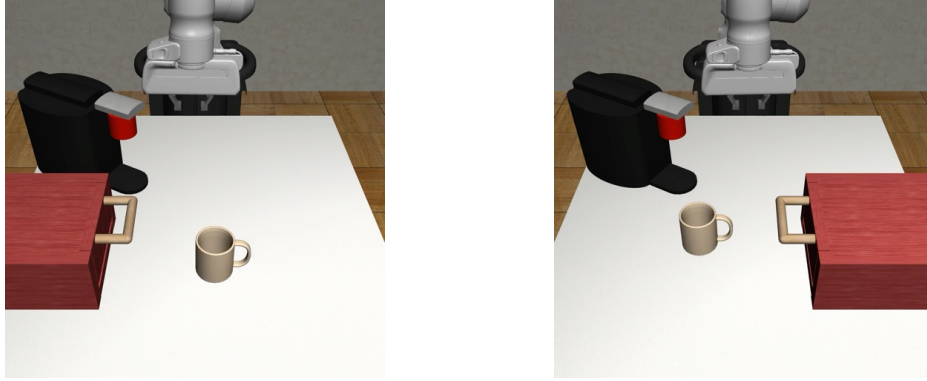


Figure D.2: **Coffee Prep  $D_0$  and  $D_2$ .** Example configurations for two task variants of Coffee Prep. Source demonstrations were collected on  $D_0$ . MimicGen is unable to generate data on  $D_2$  due to the drawer and mug being on opposite ends of the table compared to the source demos, while SkillGen successfully generates data and trains proficient policies for  $D_2$ .

735 **Safe and Proficient Real World Deployment.** SkillGen is able to obtain proficient policies in  
 736 the real world, as shown in Sec. 6.3, unlike MimicGen. MimicGen had to use longer interpolation  
 737 segments in the real world, to enforce safety during execution, which made policy learning results  
 738 suffer (as discussed above).

## E Dataset Scaling Law Analysis

We present results for using different amounts of SkillGen data for policy training, to see how policy success rate scales with amount of data. We present results in Table E.1 (for HSP-TAMP), Table E.2 (for HSP-Class) and Table E.3 (for HSP-Reg). HSP-Reg uses SkillGen with initiation augmentation (Sec. 4.5). All tasks and methods receive a significant increase from 200 to 1000 demos, and some tasks benefit strongly from 1000 to 5000 demos, notably Square  $D_2$  (52% to 72% on HSP-Reg) and Threading  $D_1$  (60% to 76% on HSP-Reg).

Task Variant	Human 200	SkillGen 200	SkillGen 1000	SkillGen 5000
Square ( $D_2$ )	88.0	84.0	<b>94.0</b>	92.0
Threading ( $D_1$ )	32.0	36.0	72.0	<b>84.0</b>
Piece Assembly ( $D_0$ )	<b>98.0</b>	88.0	96.0	96.0
Piece Assembly ( $D_2$ )	60.0	60.0	84.0	<b>92.0</b>

Table E.1: **Policy Training Dataset Comparison with HSP-TAMP [13]**. Table of results corresponding to the comparison in Fig. 4 (upper right). HSP-TAMP agent performance is comparable on 200 SkillGen demos and 200 human demos, despite SkillGen using just 10 human demos for generation. Generating more SkillGen demonstrations can result in significant performance improvement.

Task Variant	SkillGen 200	SkillGen 1000	SkillGen 5000
Square ( $D_2$ )	74.0	94.0	<b>96.0</b>
Threading ( $D_1$ )	34.0	66.0	<b>80.0</b>
Piece Assembly ( $D_0$ )	72.0	80.0	<b>86.0</b>
Piece Assembly ( $D_2$ )	44.0	74.0	<b>78.0</b>

Table E.2: **Policy Training Dataset Comparison with HSP-Class**. Generating more SkillGen demonstrations can result in modest performance improvement.

Task Variant	SkillGen 200	SkillGen 1000	SkillGen 5000
Square ( $D_2$ )	4.0	52.0	<b>72.0</b>
Threading ( $D_1$ )	14.0	60.0	<b>76.0</b>
Piece Assembly ( $D_0$ )	68.0	<b>86.0</b>	82.0
Piece Assembly ( $D_2$ )	2.0	50.0	<b>62.0</b>

Table E.3: **Policy Training Dataset Comparison with HSP-Reg**. Generating more SkillGen demonstrations can result in substantial performance improvement for certain tasks.

## F Data Generation Success Rates

We present data generation rates for the datasets used in our experiments (Table F.1 for simulation tasks and Table F.2 for real-world tasks and the sim-to-real task). In most cases, SkillGen achieves higher data generation rates than MimicGen. One notable exception is when using initiation augmentation (Sec. 4.5) – success rates are much lower in this case. However, this is due to the aggressive noise distribution applied to motion planner targets during the generation process. See Appendix G.3 for more discussion.

Task Variant	MimicGen [11]	SkillGen	SkillGen (+IA)
Square ( $D_0$ )	73.7	<b>99.8</b>	30.7
Square ( $D_1$ )	48.9	<b>91.5</b>	34.3
Square ( $D_2$ )	31.8	<b>87.7</b>	27.5
Threading ( $D_0$ )	51.0	<b>76.2</b>	35.0
Threading ( $D_1$ )	39.2	<b>66.4</b>	27.2
Threading ( $D_2$ )	21.6	<b>74.3</b>	24.9
Piece Assembly ( $D_0$ )	35.6	<b>82.5</b>	5.1
Piece Assembly ( $D_1$ )	35.5	<b>72.7</b>	4.7
Piece Assembly ( $D_2$ )	31.3	<b>69.3</b>	4.6
Coffee ( $D_0$ )	<b>78.2</b>	73.3	9.3
Coffee ( $D_1$ )	63.5	<b>73.6</b>	9.1
Coffee ( $D_2$ )	27.7	<b>70.0</b>	8.5
Nut Assembly ( $D_0$ )	53.0	<b>98.6</b>	15.2
Nut Assembly ( $D_1$ )	30.0	<b>91.7</b>	15.1
Nut Assembly ( $D_2$ )	22.8	<b>69.1</b>	10.6
Coffee Prep ( $D_0$ )	53.2	<b>64.6</b>	1.4
Coffee Prep ( $D_1$ )	<b>36.1</b>	<b>36.8</b>	0.7
Coffee Prep ( $D_2$ )	0.0	<b>59.9</b>	0.6
Average	40.7	<b>75.4</b>	14.7

Table F.1: **Data Generation Rates for Simulation Environments.** SkillGen improves data generation rates over MimicGen substantially. When using initiation augmentation (+IA), data generation rates are much lower, due to the aggressive noise distribution applied to motion planner targets.

Task	MimicGen [11]	SkillGen
Pick-Place-Milk	-	100.0
Cleanup-Butter-Trash	-	95.0
Coffee	52.0	73.0
Nut-Assembly [Sim]	72.6	94.8

Table F.2: **Data Generation Results on Real World Manipulation Tasks and Sim-to-Real Tasks.** SkillGen has high data generation throughput even in the real world, and compares favorably to MimicGen. The bottom part of the table shows the data generation rate in simulation for the task used for sim-to-real transfer.

## G Data Generation Details

In this section, we provide more details on SkillGen data generation. We first describe how reference skill segments are selected, and how they are transformed and executed. We next describe how initiation augmentation can be used to produce more robust closed-loop agents. Finally, we describe how we leveraged parallelization to generate large datasets efficiently with reasonable wall clock times, even when data generation rates were low.

### G.1 Reference Skill Segment Selection

During a data generation attempt, SkillGen adapts existing skill segments to the new scene and executes them sequentially with motion segments (Sec. 4.4). To generate a new skill segment (for skill index  $i$  for a task), SkillGen requires a reference skill segment  $\tau_{is}$  to be selected from the source demonstrations  $D_{src}$ . Since the skill index should match between the source demonstrations and the current skill segment that must be generated, this problem reduces to selecting a source demonstration index  $j \in \{1, 2, \dots, N\}$ . In our experiments, we sample this index randomly for the first skill segment, and then leave it fixed for the rest of the episode. However, more sophisticated selection methods could be used to select a different source demonstration index for each skill index if desired.

### G.2 Skill Segment Execution and Action Noise

During a data generation attempt, after an existing skill segment is selected and transformed to obtain a new sequence of end-effector pose actions  $\tau'_{is} = (T_W^{A'_0}, \dots, T_W^{A'_K})$  (Sec. 4.4), this sequence of actions is executed one by one. However, we found it beneficial to apply additive noise to the pose actions. As in MimicGen, we convert each absolute pose action to a normalized delta pose action (using the current robot end effector pose) and add Gaussian noise  $\mathcal{N}(0, 1)$  with magnitude  $\sigma$  in each dimension, where  $\sigma = 0.05$ . Note that the gripper actuation actions are copied as-is from the source skill segment, and no noise is added. These modified normalized delta pose actions are then executed, and stored in the generated dataset.

### G.3 Initiation Augmentation

As described in Sec. 4.5, SkillGen has the option of adding noise to the skill initiation states  $T_W^{E_0}$ , producing new initiation states  $T_W^{E'_0}$ , to broaden the support of the initiation set and allow the trained closed-loop skill policies to be more robust to incorrect initiation pose predictions. We found this to be very helpful for HSP-Reg agents, which must directly predict initiation poses via regression. Consequently, all of our HSP-Reg agents are trained on datasets with initiation augmentation, unless otherwise noted.

For datasets generated with initiation augmentation, we add uniform translation noise to the target position for each initiation state  $\mathcal{U}[-t, t]$ , where  $t$  is the position noise scale. We also modify the target rotation, by sampling a random rotation axis (random vector on 3D unit sphere), sampling a random angle  $\phi \sim \mathcal{U}[0, r]$ , converting the new sampled axis-angle rotation to a rotation matrix, and multiplying the target rotation by this rotation matrix. The motion planner will attempt to reach the new target pose, and then we will subsequently plan and execute a *recovery segment* consisting of a sequence of pose actions that moves from new pose  $T_W^{E'_0}$  to the original pose  $T_W^{E_0}$ . The recovery segment is added to the transformed skill segment, and is part of the dataset used to train the closed-loop agent.

In our experiments, we chose  $t = 0.08$  meters and  $r = 80$  degrees. We note that this is a very wide and aggressive pose randomization distribution, and that a large portion of sampled poses will be unreachable by the motion planner, due to the pose being in collision with the scene. This is why the data generation rates for the initiation augmentation datasets are significantly lower (Appendix F). This could be addressed with more intelligent sampling mechanisms, but we leave this for future work. Instead, opted to leverage parallelization during data generation to efficiently generate large datasets in a reasonable amount of wall clock time (described below).



#### 801 **G.4 Efficient Data Generation with Parallelization**

802 Datasets generated with initiation augmentation can have low data generation rates due to the broad  
803 noise distribution and rejection sampling process used. To mitigate this, we parallelized data col-  
804 lection across a large number of cpu processes. The SkillGen data generation process is easily  
805 amenable to this type of parallelization.

#### 806 **G.5 Hardware**

807 Data generation runs were batched together and run simultaneously (on a compute cluster) on 8-  
808 GPU nodes consisting of 8 NVIDIA Volta V100 GPUs, 64 CPUs, and 400GB of memory. Real  
809 robot experiments were run on a machine with an NVIDIA GeForce RTX 3090 GPU, 36 CPUs,  
810 32GB of memory, and 1 TB of storage.

## H Policy Learning Details

Here, we describe how policies are trained with SkillGen data. All policies trained on MimicGen data are trained with BC-RNN [1] using the same hyperparameters as in MimicGen [11].

### H.1 Observation Spaces

Every network used camera observations, consisting of a front-view camera and a wrist-view camera, and proprioception consisting of end effector poses and gripper finger positions unless otherwise mentioned. The simulation tasks used an image resolution of 84x84 and the real-world tasks used an image resolution of 120x160. All networks taking image inputs utilize pixel shift randomization [1, 47–50] to shift image pixels by up to 10% of each dimension randomly on each forward pass.

### H.2 Policy Evaluation

Unless otherwise mentioned, policies are evaluated using 50 rollouts per checkpoint during training. The best-performing policy success rate is reported for each training run [1].

### H.3 Training Procedures and Hyperparameters

We outline how each network used by the HSP algorithms described in Sec. 4.6 is trained. All networks are trained with the Adam optimizer [85] with a learning rate of 1e-4. Only one network of each type is used across all skill segments (e.g. we do not train separate networks per skill).

**Policy Network ( $\pi_\theta$ ) (HSP-Reg, HSP-Class, HSP-TAMP):** The policy network is trained with BC-RNN using robomimic [1] using the same default network structure and hyperparameters from their study. This matches the settings used for training policies in MimicGen [11].

**Termination Classifier ( $\mathcal{T}_\theta$ ) (HSP-Reg, HSP-Class):** This is a binary classification network  $\mathcal{T}_\theta : \mathcal{O} \rightarrow \{0, 1\}$  that is trained to predict when the skill policy should be running. The network architecture uses the same observation encoder structure (with different learned weights) as the policy network – each image is encoded using a ResNet-18 network [86] followed by a spatial-softmax layer [87], and these outputs are concatenated directly with the other non-image observations. This is then fed to an MLP with 2 hidden layers of size 1014, which outputs 2 logits. The network is trained with a standard multi-class classification Cross-Entropy loss. Labels to train this network are easily obtained from the SkillGen dataset, as each observation-action pair  $(o, a)$  is labeled with whether it was collected while the motion planner was running or not. We additionally apply data augmentation, and flip the labels on the last 50% of each motion planner segment. This is useful to ensure the termination classifier does not erroneously predict that the policy should terminate at the start of the skill segment. During agent rollouts, we additionally only accept a valid termination prediction when termination has been predicted 5 times – we found this to be a simple mechanism to prevent early termination prediction.

**Initiation Regression Network ( $\mathcal{I}_\theta$ ) (HSP-Reg):** This is a network  $\mathcal{I}_\theta : \mathcal{O} \rightarrow \text{SE}(3)$  that directly predicts an end effector pose corresponding to the initiation condition for the next skill. The architecture is the same as the termination classifier, except for the last layer, which directly predicts a position (3-dim) and a rotation (6-dim rotation representation from Zhou et al [88]). To allow for multimodal predictions, we use a Gaussian Mixture Model (GMM) head, using the same hyperparameters as the BC-RNN-GMM model from robomimic [1]. The position and rotation targets to train the network come from the SkillGen dataset, and correspond to the targets that were sent to the motion planner during data generation. These targets are normalized to lie in  $[-1, +1]$ , using the same procedure from Chi et al. [89]. During agent rollouts, this network directly samples a target pose for the motion planner to reach.

**Initiation Classifier ( $\mathcal{I}_\theta$ ) (HSP-Class):** This is a classification network  $\mathcal{I}_\theta : \mathcal{O} \rightarrow \{1, 2, \dots, N_{\text{src}}\}$  that frames skill initiation condition prediction as a classification problem over initiation states in the source dataset  $\mathcal{D}_{\text{src}}$ . The architecture is the same exact network (with shared weights) as the termination classifier described above ( $\mathcal{T}_\theta$ ) – there is simply an extra classification head added to the output of the network. It is trained to predict the source demonstration in  $\mathcal{D}_{\text{src}}$  that spawned the generated demonstration in  $\mathcal{D}$  using a standard multi-class classification Cross-Entropy loss. During

861 agent rollouts, after predicting a source demonstration label, the corresponding initiation state in the  
862 source demonstration is adapted to the current state using the adaptation procedure from Sec. 4.4 to  
863 obtain a target pose for the motion planner.

#### 864 **H.4 Hardware**

865 Policy learning runs each used a machine (on a compute cluster) with an NVIDIA Volta V100 GPU,  
866 8 CPUs, and 50GB of memory. Real robot experiments were run on a machine with an NVIDIA  
867 GeForce RTX 3090 GPU, 36 CPUs, 32GB of memory, and 1 TB of storage.

## I Planning Details

In this section, we provide additional details on the Motion Planner and the Task and Motion Planner used in our experiments, beyond those in Sec. 5. We used PyBullet [90] for collision checking during motion planning and TAMP. Within the HITL-TAMP system, we used PDDLStream [91] for task and motion planning.

For each motion planning query, we decompose planning into three phases. The first is a short *retreat* motion that moves the robot’s end effector backward. The second is a transit or transfer motion that moves the robot a short distance in front of the query pose. The third is an *approach* motion that moves to the query pose. The retreat and approach motions move the robot out of and into contact respectively. During these short phases, we ignore expected collisions between the robot and any manipulated object along with collisions between manipulated objects and the environment. In our experiments, we used a retreat and approach distance of 5cm in the end effector’s z axis.

Expanding on Section 6.3, in the real world, we assume manipulable object segmentation. While a number of choices on segmentation methods can be made [92,93], we deploy a simple yet effective pipeline which works well in our setup. Specifically, we first perform RANSAC plane fitting to filter the table from the observed point cloud. Then, we use DBSCAN [94] to cluster the objects within the remaining point cloud. In settings where we have shape models for the objects, the object cloud segments are distinguished by comparing to their respective 3D models and we use FoundationPose [83] for object 6D pose estimation. Otherwise, we reconstruct collision volumes for the manipulable objects online by running marching cubes [95] on each segmented point cloud. For transfer motion planning, we detect whether a manipulable object is grasped by checking for contact between both the robot’s fingers and the object in our planning model. While grasped, we assume the object is rigidly attached to the robot, modifying its collision geometry.

## J Tasks and Task Variants

In this section, we provide detailed descriptions of all the tasks (Fig. J.1) and task variants. See the website (<https://skillgen.github.io>) for more visualizations. The action space for all tasks is a delta-pose action space (using an Operational Space Controller [82]) to control the arm), along with a gripper open/close command. Control occurs at 20 hz.

### J.1 Simulation Tasks

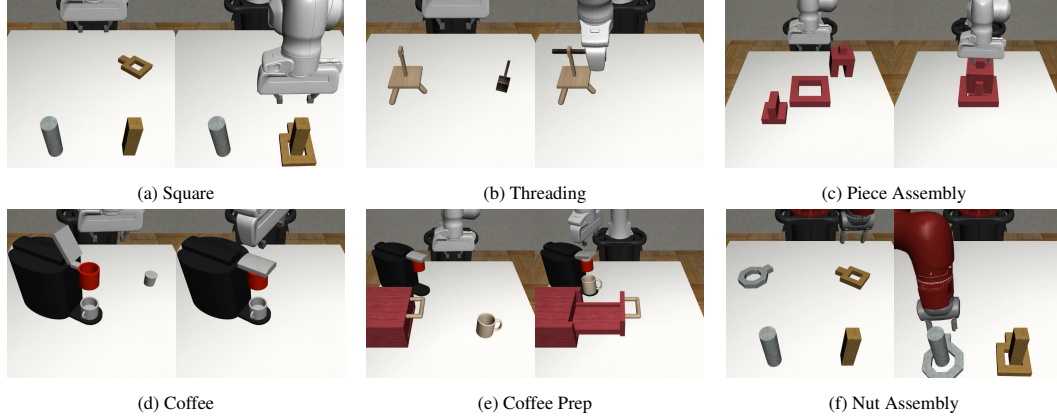


Figure J.1: **Simulation Tasks.** We deploy SkillGen on 6 simulation tasks (18 task variants). These tasks include fine-grained and long-horizon manipulation.

All tasks and task variants are taken from the MimicGen paper [11], with the exception of Nut Assembly ( $D_1$ ,  $D_2$ ) and Coffee Prep ( $D_2$ ), which were newly implemented. For each task, we describe the goal, the task variants, and the skill segments.

- **Square.** The robot must pick a square nut and place it on a peg. ( $D_0$ ) The peg never moves, and the nut is placed in small ( $0.005\text{m} \times 0.115\text{m}$ ) region with a random top-down rotation. ( $D_1$ ) The peg and the nut are initialized in large regions, but the peg rotation is fixed. The peg is initialized in a  $0.4\text{m} \times 0.4\text{m}$  box and the nut is initialized in a  $0.23\text{m} \times 0.51\text{m}$  box. ( $D_2$ ) The peg and the nut are initialized in larger regions ( $0.5\text{m} \times 0.5\text{m}$  box of initialization for both) and the peg rotation also varies. There are 2 skill segments (grasp nut, place onto peg).
- **Threading.** The robot must pick a needle and thread it through a hole on a tripod. ( $D_0$ ) The tripod is fixed, and the needle moves in a modest region ( $0.15\text{m} \times 0.1\text{m}$  box with 60 degrees of top-down rotation variation). ( $D_1$ ) The tripod and needle move in large regions on the left and right portions of the table respectively. The needle is initialized in a  $0.25\text{m} \times 0.1\text{m}$  box with 240 degrees of top-down rotation variation and the tripod is initialized in a  $0.25\text{m} \times 0.1\text{m}$  box with 120 degrees of top-down rotation variation. ( $D_2$ ) The tripod and needle are initialized on the right and left respectively (reversed from  $D_1$ ). The size of the regions is the same as  $D_1$ . There are 2 skill segments (grasp needle, thread into tripod).
- **Coffee.** The robot must pick a coffee pod, insert the pod into the coffee machine, and close the machine hinge. ( $D_0$ ) The machine never moves, and the pod moves in a small ( $0.06\text{m} \times 0.06\text{m}$ ) box. ( $D_1$ ) The machine and pod move in large regions on the left and right portions of the table respectively. The machine is initialized in a  $0.1\text{m} \times 0.1\text{m}$  box with 90 degrees of top-down rotation variation and the pod is initialized in a  $0.25\text{m} \times 0.13\text{m}$  box. ( $D_2$ ) The machine and pod are initialized on the right and left respectively (reversed from  $D_1$ ). The size of the regions is the same as  $D_1$ . There are 2 skill segments (grasp pod, insert-into and close machine).
- **Three Piece Assembly.** The robot must pick one piece, insert it into the base, then pick the second piece, and insert into the first piece to assemble a structure. ( $D_0$ ) The base never moves, and both pieces move around base with fixed rotation in a  $0.44\text{m} \times 0.44\text{m}$  region. ( $D_1$ ) All three pieces move in the workspace ( $0.44\text{m} \times 0.44\text{m}$  region) with fixed rotation.

- ( $D_2$ ) All three pieces can rotate (the base has 90 degrees of top-down rotation variation, and the two pieces have 180 degrees of top-down rotation variation). There are 4 skill segments (grasp piece 1, place into base, grasp piece 2, place into piece 2).
- **Nut Assembly.** Similar to Square, but the robot must place both a square nut and round nut onto two different pegs. ( $D_0$ ) Each nut is initialized in a small box (0.005m x 0.115m region with a random top-down rotation). ( $D_1$ ) The nuts are initialized in a large box (0.23m x 0.51m region) with random top-down rotation, and the pegs are initialized in a large box (0.4m x 0.4m) with a fixed rotation. ( $D_2$ ) The nuts and pegs are initialized in a larger box (0.5m x 0.5m) with random top-down rotations. There are 4 skill segments (grasp each nut and place onto each peg).
  - **Coffee Prep.** A more comprehensive version of Coffee — the robot must load a mug onto the coffee machine, open the machine, retrieve the coffee pod from the drawer and insert the pod into machine. ( $D_0$ ) The mug moves in a modest (0.15m x 0.15m) region with fixed top-down rotation and the pod inside the drawer moves in a 0.06m x 0.08m region while the machine and drawer are fixed. ( $D_1$ ) The mug is initialized in a larger region (0.35m x 0.2m box with random top-down rotation) and the machine also moves in a modest region (0.1m x 0.05m box with 60 degrees of top-down rotation variation). ( $D_2$ ). Same task as  $D_0$  but the drawer is placed on the right side of the table, and the mug is initialized on the left side of the table, instead of the right. There are 5 skill segments (grasp mug, place onto machine and open lid, open drawer, grasp pod, insert into machine and close lid).

## 947 J.2 Real-World Tasks

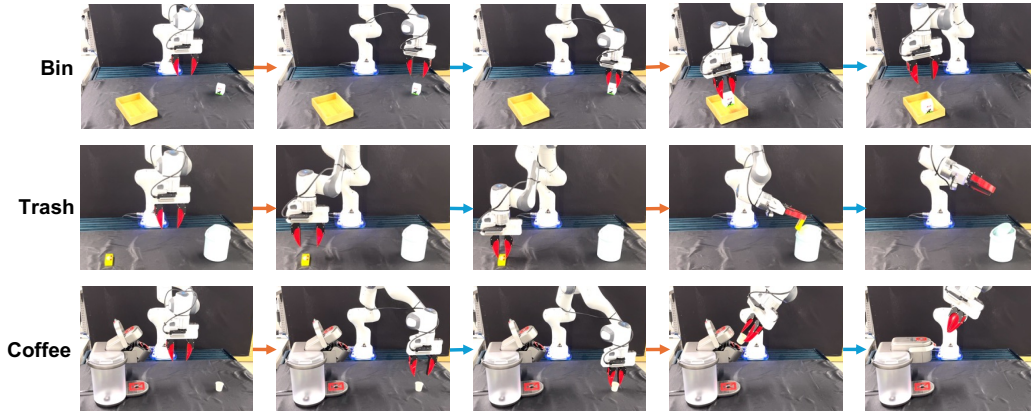


Figure J.2: **Real-World Task Executions.** The 1) initial state, 2) pick initiation state, 3) pick termination state, 4) placement or insertion initiation state, and 5) placement or insertion termination state for an example episode of the Milk-Bin, Butter-Trash, and Coffee tasks. The orange arrows indicate a transition facilitated by motion planning, and the blue arrows indicate a transition conducted by a learned skill policy.

Figure J.2 and Figure J.3 display example task executions and the initial state distributions respectively for the Pick-Place-Milk, Cleanup-Butter-Trash, and Coffee tasks introduced in Section 6.3. For each task, we describe the goal, the initialization regions for the objects, and the skill segments.

- **Pick-Place-Milk.** The robot must pick the milk and place it in the bin. The milk and bin objects are randomly placed anywhere on the table, with random orientations that are within  $\pm 45$  degrees of yaw from their nominal orientations. There are two skill segments: pick milk and place into bin.
- **Cleanup-Butter-Trash.** The robot must pick the butter and insert it into the trash can by pushing the trash can's lid. The butter and trash can are placed randomly on the left and right sides of the table respectively, with random orientations that are within  $\pm 45$  degrees of yaw from their nominal orientations. There are two skill segments: pick butter and insert into trash can.
- **Coffee.** The robot must pick the coffee pod, insert it into the coffee machine, and then close the machine's lid. The pod is initialized in a 0.44m x 0.35m box (as in MimicGen [11]).

962 There are two skill segments: pick pod and both insert pod into machine as well as close  
963 lid. Here, the insertion and closing are treated as a single learned skill.

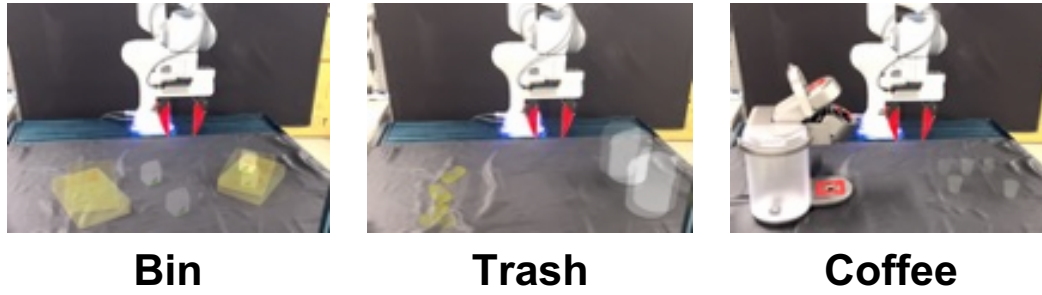


Figure J.3: **Real-World Reset Distributions.** The initial states of the Pick-Place-Milk, Cleanup-Butter-Trash, and Coffee tasks each overlaid onto a single image.



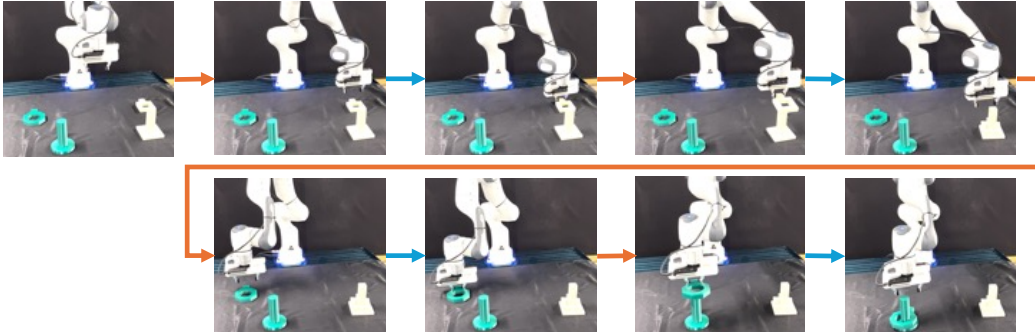


Figure K.1: **Real-World Nut Assembly Execution.** An example execution of the real-world Nut Assembly task. The task involves four skill segments (in blue) and four motion planning segments (in orange). The skill segments are 1) pick the Square Nut, 2) place the Square Nut on the Square Peg, 3) pick the Round Nut, 4) place the Round Nut on the Round Peg.

965 As described in Section 6.3, we performed an experiment to explore SkillGen’s ability at facilitating  
 966 zero-shot sim-to-real transfer. In this section, we provide further details omitted in the main text.  
 967 We considered the “Nut Assembly” task (Figure K.1) where the robot must pick a Square Nut, place  
 968 the Square Nut on the Square Peg, pick a Round Nut, and place the Round Nut on the Round Peg.  
 969 This task is long-horizon in that it involves four skill stages; additionally, each place stage require  
 970 precise manipulation to fit each nut on its associated peg.

971 **Insertion Tolerance.** We designed a simulated analog of the task, “Nut Assembly [Sim]”, in ro-  
 972 bosuite [78]. In the real world, we replicate the CAD model of each nut and peg and 3D printed  
 973 them so that the real and simulated geometries matched. The square peg is 3.2cm on each side,  
 974 the square hole on the nut is 4.6cm on each side, the round peg is 4cm in diameter, and the round  
 975 hole is approximately 6.8cm in diameter, leaving only a couple centimeters of tolerance for each nut  
 976 insertion.

977 **Initialization Bounds.** In simulation, the nuts and pegs are each initialized randomly in non-  
 978 overlapping 21cm x 41.5m quadrants of the table, with fixed orientation. In the real world, the  
 979 initialization region for the objects are as follows: square nut (18cm x 26cm), round nut (20cm x  
 980 20cm), square peg (14cm x 40cm) and round peg (16cm x 30cm). The simulation bounds were  
 981 intentionally designed to be more extensive than the real world initialization bounds.

982 **Observation and Action Space.** As mentioned in Section 7, we made several assumptions specifi-  
 983 cally for this experiment. First, we trained pose-based rather than image-based policies. As a result,  
 984 there is no visual sim-to-real transfer. Second, because pose estimation during robot execution  
 985 can be challenging, for example, due to the robot occluding the camera, each policy observes only  
 986 the initial object poses. They do however consume up-to-date robot proprioception measurements,  
 987 consisting of the end effector position and width of the gripper fingers. We make an additional sim-  
 988 plification, and provide the end effector position with respect to the initial object position for all 4  
 989 items, instead of providing the end effector position and the object poses separately. Consequently,  
 990 the final observation consumed by the agent is simply the robot end effector position with respect  
 991 to the initial square nut position, square peg position, round nut position, and round peg position,  
 992 as well as the width of the gripper fingers. Additionally, we simplified the agents action space by  
 993 fixing the orientation of the end effector, which results in a 4-DOF position-only action space (one  
 994 extra dim for gripper actuation).

995 **Policy Training Details.** We mostly follow the procedure described in Appendix H for HSP-Class  
 996 training and the procedure from MimicGen [11] for training the MimicGen policies. We use an  
 997 increased learning rate of 1e-3 for the closed-loop policy network. We also change the RNN policy  
 998 to make it “open-loop” over the RNN horizon by repeating the first observation in the sequence  
 999 instead of providing the current observation – this is equivalent to the action chunking described in  
 1000 Zhao et al. [19].

1001 **Experiment Summary.** Ultimately, through SkillGen, we were able amplify a single simulation  
 1002 source demonstration into 1000 simulation demonstrations on Nut Assembly [Sim], train a pose-



1003 based HSP-Class policy, and deploy it using SkillGen without any real-world data, where it achieved  
1004 35% success rate, while the MimicGen agent could not complete the full task, and achieved 5%  
1005 success rate on the first square nut insertion.

## L Results with Conventional Teleoperation Source Demonstrations

Task Variant	MimicGen [11]	SkillGen
Square ( $D_0$ )	73.7	<b>87.3</b>
Square ( $D_1$ )	48.9	<b>73.8</b>
Square ( $D_2$ )	31.8	<b>65.1</b>
Threading ( $D_0$ )	<b>51.0</b>	43.6
Threading ( $D_1$ )	<b>39.2</b>	36.7
Threading ( $D_2$ )	21.6	<b>36.9</b>
Piece Assembly ( $D_0$ )	35.6	<b>48.7</b>
Piece Assembly ( $D_1$ )	35.5	<b>48.4</b>
Piece Assembly ( $D_2$ )	31.3	<b>53.8</b>
Coffee ( $D_0$ )	78.2	<b>81.5</b>
Coffee ( $D_1$ )	63.5	<b>75.4</b>
Coffee ( $D_2$ )	27.7	<b>59.8</b>
Average	44.8	<b>59.3</b>

Table L.1: **Data Generation Rates from using Conventional Teleoperation Source Data.** SkillGen improves data generation rates over MimicGen substantially for most tasks, particularly the  $D_2$  variants.

Task Variant	MimicGen [11]	HSP-Class	HSP-Reg
Square ( $D_0$ )	90.7	<b>100.0</b>	84.0
Square ( $D_1$ )	73.3	<b>84.0</b>	58.0
Square ( $D_2$ )	49.3	<b>68.0</b>	46.0
Threading ( $D_0$ )	<b>98.0</b>	94.0	94.0
Threading ( $D_1$ )	<b>60.7</b>	46.0	56.0
Threading ( $D_2$ )	38.0	34.0	<b>50.0</b>
Piece Assembly ( $D_0$ )	<b>82.0</b>	80.0	74.0
Piece Assembly ( $D_1$ )	<b>62.7</b>	48.0	52.0
Piece Assembly ( $D_2$ )	13.3	<b>42.0</b>	36.0
Coffee ( $D_0$ )	<b>100.0</b>	98.0	<b>100.0</b>
Coffee ( $D_1$ )	90.7	<b>100.0</b>	94.0
Coffee ( $D_2$ )	77.3	<b>92.0</b>	90.0
Average	70.0	<b>73.8</b>	69.5

Table L.2: **Agent Performance on Datasets Generated from Conventional Teleoperation Source Data.** Across the tasks, the average SkillGen policy learning results are comparable to MimicGen, but HSP-Class slightly outperforms the MimicGen baseline.

The experiments presented in Sec. 6 used demonstrations collected with HITL-TAMP [13], a teleoperation system where humans only demonstrate select skill segments of each task. A TAMP system plans and executes the rest of the task, in between skill demonstrations. In this section, we analyze how SkillGen’s performance changes when using source demonstrations from a conventional teleoperation system instead of HITL-TAMP.

We use the same source demonstrations as MimicGen, and annotate the skill phases (Sec. 4) to enable data generation with SkillGen. Table L.1 shows that the average data generation rate is higher by 15% over MimicGen. However, Table L.2 shows that the average policy learning results are comparable to MimicGen (compared to the substantial improvements over MimicGen from using HITL-TAMP source data in Fig. 4). This shows that SkillGen performance is higher when using HITL-TAMP source data. One potential reason is due to the variability in motion planner poses when using manual annotations compared to the consistent annotations based on pre-conditions coming from the HITL-TAMP system. This variability can pose a challenge for learning methods [61]. Analyzing this gap further is a valuable avenue for future work.

## 1021 M Ablations

Task Variant	H-TAMP	H-TAMP(+T)	H-Class	H-Class(-T)	H-Reg	H-Reg(-T)	H-Reg(-R)
Square ( $D_0$ )	<b>100.0</b>	<b>100.0</b>	<b>100.0</b>	<b>100.0</b>	94.0	98.0	80.0
Square ( $D_2$ )	94.0	90.0	94.0	<b>96.0</b>	52.0	46.0	40.0
Threading ( $D_0$ )	<b>100.0</b>	<b>100.0</b>	92.0	94.0	94.0	92.0	<b>100.0</b>
Threading ( $D_1$ )	<b>72.0</b>	68.0	66.0	58.0	60.0	66.0	58.0
Piece Assembly ( $D_0$ )	<b>96.0</b>	94.0	80.0	80.0	86.0	80.0	80.0
Piece Assembly ( $D_2$ )	<b>84.0</b>	82.0	74.0	76.0	50.0	40.0	14.0
Coffee ( $D_0$ )	<b>100.0</b>	<b>100.0</b>	<b>100.0</b>	<b>100.0</b>	<b>100.0</b>	<b>100.0</b>	<b>100.0</b>
Coffee ( $D_2$ )	94.0	<b>100.0</b>	<b>100.0</b>	98.0	98.0	96.0	56.0

Table M.1: **Ablation of Key Components.** To understand the difficulty of predicting policy termination, we modify HSP-TAMP to use a termination classifier (HSP-TAMP (+T)), and modify HSP-Class and HSP-Reg to use TAMP to handle termination instead of the termination classifier (-T variants). We see that performance is largely unchanged, indicating that learning termination is relatively easy. To understand the value of initiation augmentation (Sec. 4.5), we train HSP-Reg on dataset generated without it. The large performance regressions demonstrate it can be critical.

### 1022 M.1 Difficulty of Predicting Policy Termination

1023 To understand the difficulty of predicting policy termination, we make the following changes to  
 1024 each method. HSP-TAMP (+T): we modify HSP-TAMP to use the same policy termination clas-  
 1025 sifier  $\mathcal{T}_\psi(o_t)$  from HSP-Class and HSP-Reg, and use it instead of TAMP to determine when agent  
 1026  $\pi_\theta$  should terminate and cede control back to TAMP. HSP-Class (-term) and HSP-Reg (-T): we use  
 1027 the same conditions as HSP-TAMP to dictate when to cede control from the agent  $\pi_\theta$  back to the  
 1028 motion planner, instead of using the classifier. We see that the performance of HSP-TAMP (+T) is  
 1029 at most 4% below and 6% above HSP-TAMP, showing that predicting policy termination is not very  
 1030 difficult. Comparing HSP-Class (-term) to HSP-Class (8% lower to 2% higher) and HSP-Reg (-T)  
 1031 to HSP-Reg (10% lower to 6% higher) corroborates this claim. By comparison, the significant dif-  
 1032 ference in performance between HSP-Class and HSP-Reg on a select few tasks (analyzed in Sec. 6)  
 1033 demonstrates that motion planner target prediction is significantly more challenging. This suggests  
 1034 that the key bottleneck for improving HSP-Reg performance is improving its ability to predict mo-  
 1035 tion planner target poses – there is consequently an opportunity for future work to improve this by  
 1036 integrating models that utilize 3D information [58, 59] or exploring alternative model architectures.  
 1037 See Appendix Q for further discussion.

### 1038 M.2 Value of Initiation Augmentation

1039 To show the value of initiation augmentation (Sec. 4.5), we train HSP-Reg on datasets generated  
 1040 without motion augmentation (HSP-Reg (-R)) and compare with HSP-Reg. Removing motion aug-  
 1041 mentation can cause significant performance drops (e.g. 40% drop on Coffee  $D_2$ , 26% on Three  
 1042 Piece Assembly  $D_2$ ), showing that it can be critical to enable better performance by allowing agents  
 1043 to recover from incorrect motion planner target predictions.

## 1044 N Robot Transfer

1045 We apply SkillGen to generate datasets and train agents for a robot arm that is different than the one  
 1046 the human collected source demonstrations on (Fig. N.1). We use the same source demonstrations  
 1047 as those used in our main experiments, collected on the Panda arm, and generate demonstrations  
 1048 for the Sawyer arm. The results are presented in Table N.1 (data generation) and Table N.2 (policy  
 1049 learning). We see that data generation rates are substantially higher for SkillGen than MimicGen,  
 1050 and that HSP-Class policies trained on SkillGen data are higher performing than their MimicGen  
 1051 counterparts.



Figure N.1: **Data Generation for Sawyer Robot Arm.** Example configurations for task variants where SkillGen generated data for the Sawyer robot arm, using source human data collected on the Panda robot arm.

Task Variant	MimicGen [11]	SkillGen
Square ( $D_0$ ) (Panda)	73.7	<b>99.8</b>
Square ( $D_0$ ) (Sawyer)	55.8	<b>95.2</b>
Square ( $D_1$ ) (Panda)	48.9	<b>91.5</b>
Square ( $D_1$ ) (Sawyer)	38.8	<b>94.0</b>
Threading ( $D_0$ ) (Panda)	51.0	<b>76.2</b>
Threading ( $D_0$ ) (Sawyer)	28.8	<b>68.2</b>
Threading ( $D_1$ ) (Panda)	39.2	<b>66.4</b>
Threading ( $D_1$ ) (Sawyer)	23.7	<b>62.5</b>
Nut Assembly ( $D_0$ ) (Panda)	53.0	<b>98.6</b>
Nut Assembly ( $D_0$ ) (Sawyer)	34.7	<b>86.1</b>
Nut Assembly ( $D_1$ ) (Panda)	30.0	<b>91.7</b>
Nut Assembly ( $D_1$ ) (Sawyer)	22.1	<b>78.3</b>

Table N.1: **Data Generation Rates for Generating Datasets for Different Robots.** We use SkillGen to produce datasets on the Sawyer robot arm using the same 10 source demos collected on the Panda arm. SkillGen improves data generation rates substantially over MimicGen.

Task Variant	MimicGen [11]	HSP-Class
Square ( $D_0$ ) (Panda)	90.7	<b>100.0</b>
Square ( $D_0$ ) (Sawyer)	86.0	<b>96.0</b>
Square ( $D_1$ ) (Panda)	73.3	<b>98.0</b>
Square ( $D_1$ ) (Sawyer)	60.7	<b>98.0</b>
Threading ( $D_0$ ) (Panda)	<b>98.0</b>	92.0
Threading ( $D_0$ ) (Sawyer)	88.7	<b>94.0</b>
Threading ( $D_1$ ) (Panda)	60.7	<b>66.0</b>
Threading ( $D_1$ ) (Sawyer)	50.7	<b>54.0</b>
Nut Assembly ( $D_0$ ) (Panda)	60.0	<b>92.0</b>
Nut Assembly ( $D_0$ ) (Sawyer)	74.0	<b>88.0</b>
Nut Assembly ( $D_1$ ) (Panda)	16.0	<b>78.0</b>
Nut Assembly ( $D_1$ ) (Sawyer)	8.0	<b>62.0</b>

Table N.2: **Agent Performance on Generated Datasets for Different Robot Arms.** We use SkillGen to produce datasets on the Sawyer robot arm using the same 10 source demos collected on the Panda arm. HSP-Class policies trained on SkillGen data significantly outperform agents trained on MimicGen data.

## 1052 O Algorithm Pseudocode

1053 Algorithm 1 provides the pseudocode for the data generation process described in Section 4.4. For  
 1054 each skill trajectory in the source demonstration, SkillGen first estimates the current pose of the  
 1055 object that the skill manipulates. This is used to transform the stored initiation state. Then, MOTION-  
 1056 PLANNER solves for a robot configuration that reaches this pose and plans a joint-space path to the  
 1057 configuration, executed with a joint space controller. Finally, each end-effector action is adapted to  
 1058 the world frame and executed using task-space control.

---

### Algorithm 1 Demonstration Generation

---

```

procedure GENERATE-DATA( $\tau$ )
  for  $\tau_{is} \in \tau$  do
     $T_W^{O'_i} \leftarrow \text{ESTIMATE-POSE}()$ 
     $T_W^{E'_0} \leftarrow T_W^{O_i} \tau_{is}[0]$ 
     $q_0 \leftarrow \text{CURRENT-CONFIG}()$ 
     $q_* \leftarrow \text{INVERSE-KINEMATICS}(T_W^{E'_0})$ 
    for  $q \in \text{MOTION-PLANNER}(q_0, q_*)$  do
       $\text{JOINT-SPACE-CONTROL}(q)$ 
    for  $T_{O_i}^{E_t} \in \tau_{is}$  do
       $T_W^{E'_t} \leftarrow T_W^{O'_i} T_{O_i}^{E_t}$ 
       $\text{TASK-SPACE-CONTROL}(T_W^{E'_t})$ 

```

---

1059 Algorithm 2 provides the pseudocode for HSP deployment, which was described Section 4.6. The  
 1060 structure has some global similarity with Algorithm 1, but critically, it operates over skills instead of  
 1061 trajectories and does not require pose estimation. For each skill in a provided sequence of skills  $\Psi$ ,  
 1062 DEPLOY-HSP predicts the initiation pose using the current observation  $o$ . Then, it plans and executes  
 1063 joint-space motions to the initiation pose. Until the termination network predicts to terminate, the  
 1064 skill queries its policy for the next task-space action.

---

### Algorithm 2 HSP Deployment

---

```

procedure DEPLOY-HSP( $\Psi$ )
  for  $\langle O, \mathcal{I}_\theta, \pi_\theta, \mathcal{T}_\theta \rangle \in \Psi$  do
     $o \leftarrow \text{OBSERVE}()$ 
     $T_W^{E'_0} \leftarrow \mathcal{I}_\theta(o)$ 
     $q_0 \leftarrow \text{CURRENT-CONFIG}()$ 
     $q_* \leftarrow \text{INVERSE-KINEMATICS}(T_W^{E'_0})$ 
    for  $q \in \text{MOTION-PLANNER}(q_0, q_*)$  do
       $\text{JOINT-SPACE-CONTROL}(q)$ 
    while  $\mathcal{T}_\theta(o) \neq \text{True}$  do
       $T_W^{E'_t} \leftarrow \pi_\theta(o)$ 
       $\text{TASK-SPACE-CONTROL}(T_W^{E'_t})$ 
       $o \leftarrow \text{OBSERVE}()$ 

```

---

## 1065 P Comparison with HITL-TAMP

1066 As we described in Section 2, HITL-TAMP [13] is a prior system that integrates BC and planning to  
1067 improve both data collection efficiency and policy success rates. Within SkillGen, we optionally use  
1068 HITL-TAMP to both collect a handful of source demonstrations (Section 4.3) and deploy learned  
1069 skills at test time through HSP-TAMP (Section 4.6). However, when compared directly, SkillGen  
1070 has several advantages over HITL-TAMP.

1071 **Fewer Assumptions.** HITL-TAMP requires a model to plan the TAMP segments. Specifying one  
1072 requires defining Planning Domain Definition Language (PDDL) actions, including their parame-  
1073 ters, preconditions, and effects, along with sampling procedures that generate continuous action pa-  
1074 rameter values [91]. In contrast, SkillGen only requires a skill plan at data generation time, namely  
1075 the sequence of objects that will be acted upon. At test time, SkillGen can even reduce this assump-  
1076 tion by learning a single skill that encompasses all learned segments without explicitly conditioning  
1077 on any objects. HITL-TAMP also requires pose observation or estimation during all its phases, for  
1078 example, to define TAMP-gated hand-off regions from TAMP to a learned policy. Through directly  
1079 learning initiation sets, which can be viewed as learning-gated conditions, SkillGen not only uses  
1080 learning to transfer control but also avoids pose estimation in its HSP-Reg configuration.

1081 **Lower Human Effort.** Although HITL-TAMP partially automates the demonstration process, a  
1082 human must still manually teleoperate a portion of each episode. Thus, the amount of human effort  
1083 required scales linearly with the number of demonstrations. In contrast, SkillGen only requires a  
1084 fixed amount of human effort and can spawn an arbitrarily large number of demonstrations. Further-  
1085 more, policy learning results can be comparable given a similar amount of SkillGen demonstrations  
1086 and HITL-TAMP demonstrations (Sec. 6.2), with just a fraction of the human effort.

1087 **Object Grasp Segments Delegated to Learned Policies.** HITL-TAMP, TAMP is responsible for  
1088 carrying out object grasps, while SkillGen defers all object interaction to learned agents. For exam-  
1089 ple, in the real-world Coffee task, TAMP controls the arm to grasp the coffee pod and approach the  
1090 insertion point on the machine, and a learned policy is only responsible for the insertion segment. In  
1091 SkillGen, there are two skill segments that must be learned – one for pod grasping and one for pod  
1092 insertion. Despite this increased difficulty, a policy trained on SkillGen data performs comparably.  
1093 HSP-Class obtains 65% on the Coffee task with 100 SkillGen demos generated from just 3 human  
1094 demos, in comparison to HSP-TAMP achieving 74% from 100 HITL-TAMP demos.

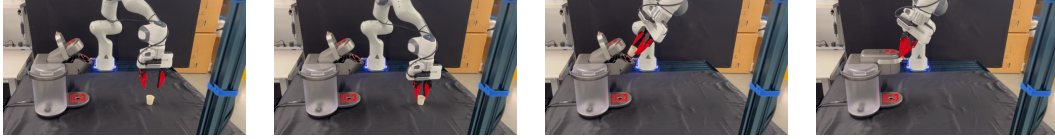


Figure P.1: **Comparison Between Skill Segments Learned by SkillGen and HITL-TAMP.** The experiments in HITL-TAMP [13] assumed that TAMP carries out object grasps (the left two frames for the Coffee task shown above) – consequently, the trained agent was responsible for less portions of each task (e.g. the right two frames for the Coffee task above). By contrast, SkillGen is responsible for all segments shown above.

## 1095 Q Discussion on HSP-Reg Results

1096 HSP-Reg makes the fewest assumptions out of the three HSP methods presented in this work  
1097 (Sec. 4.6). However, while the average task success rate is only lower by 10% to 13% than the  
1098 other methods, there can still be a significant gap in policy performance depending on the specific  
1099 task. In this section, we provide some reasons to be optimistic that HSP-Reg performance can be  
1100 increased significantly.

1101 **Using more SkillGen data.** In this work, our main experiments (Fig. 4) used 1000 SkillGen demon-  
1102 strations – this number was chosen for consistency with prior work [11]. However, we found that  
1103 using more demonstrations can significantly boost HSP-Reg results (Appendix E). Some notable  
1104 performance increases from 1000 SkillGen demos to 5000 SkillGen demos include Square  $D_2$  (52%  
1105 to 72% on HSP-Reg) and Threading  $D_1$  (60% to 76% on HSP-Reg).

1106 **Improving agent observability.** HSP-Reg is responsible for directly predicting a 6-DoF target  
1107 pose for the motion planner to reach – this can be the key bottleneck for improving performance  
1108 (corroborated by ablations in Appendix M and the performance gap between HSP-Reg and HSP-  
1109 Class). This can be a difficult prediction problem when using just a front-view and wrist-view image  
1110 for this prediction. Consequently, we ran an experiment to see if adding a third, side-view image  
1111 would improve results. We used 5000 SkillGen demos with front-view, wrist-view, and side-view  
1112 observations, and obtained our best HSP-Reg results – 82% for Square  $D_2$  (compared to the 52% in  
1113 Fig. 4), 86% for Threading  $D_1$  (compared to 60%), and 74% for Piece Assembly  $D_2$  (compared to  
1114 50%). These results also demonstrate that adding depth information for the pose prediction can be  
1115 beneficial, as used in prior work [58, 59].



## 1116 **R Skill Segments and Annotations**

1117 In order to amplify a set of source demonstrations in a targeted manner, SkillGen requires annotation  
1118 of the start and end of each skill that should be learned on the demonstrations. Even when using  
1119 HITL-TAMP to gather source demonstrations, the TAMP model must specify action preconditions  
1120 and effects (Appendix P), which loosely correspond to skill initiation and termination conditions.

1121 The choice of what skills to learn and how fine-grained they should be can be customized by a  
1122 human supervisor. For example, in the Coffee task displayed Fig. J.2, the robot must insert the pod  
1123 and then close the coffee machine lid. We choose to model and learn both behaviors as a single skill  
1124 rather than split them into two separate skills, connected by transit motion planning. This imposes a  
1125 larger burden on learning but reduces the execution time by not requiring motion planning between  
1126 the two behaviors.

1127 Ultimately, the motivation of Sec. 4 is our primary recommendation with respect to modeling prin-  
1128 ciples. Motion planning is a safe and reliable technique for addressing contact-adverse segments of  
1129 tasks, which are often substantial in many common tasks. If learned policies are able to replicate  
1130 these attributes for a given task, then it makes sense to incorporate more learning. Otherwise, defer-  
1131 ring learning to primarily the contact-rich task segments, where motion planning is ineffective is the  
1132 wiser strategy.

## References

- [1] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, “What matters in learning from offline human demonstrations for robot manipulation,” in *Conference on Robot Learning (CoRL)*, 2021.
- [2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, “Rt-1: Robotics transformer for real-world control at scale,” *arXiv preprint arXiv:2212.06817*, 2022.
- [3] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg, and P. Abbeel, “Deep imitation learning for complex manipulation tasks from virtual reality teleoperation,” *arXiv preprint arXiv:1710.04615*, 2017.
- [4] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei, “RoboTurk: A Crowdsourcing Platform for Robotic Skill Learning through Imitation,” in *Conference on Robot Learning*, 2018.
- [5] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine, “Bridge data: Boosting generalization of robotic skills with cross-domain datasets,” *arXiv preprint arXiv:2109.13396*, 2021.
- [6] 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.
- [7] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, “Bc-z: Zero-shot task generalization with robotic imitation learning,” in *Conference on Robot Learning*. PMLR, 2022, pp. 991–1002.
- [8] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence, “Interactive language: Talking to robots in real time,” *arXiv preprint arXiv:2210.06407*, 2022.
- [9] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, “Recent advances in robot learning from demonstration,” *Annual review of control, robotics, and autonomous systems*, vol. 3, pp. 297–330, 2020.
- [10] M. Dalal, A. Mandlekar, C. Garrett, A. Handa, R. Salakhutdinov, and D. Fox, “Imitating task and motion planning with visuomotor transformers,” *arXiv preprint arXiv:2305.16309*, 2023.
- [11] 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,” *arXiv preprint arXiv:2310.17596*, 2023.
- [12] R. Hoque, A. Mandlekar, C. R. Garrett, K. Goldberg, and D. Fox, “Interventional data generation for robust and data-efficient robot imitation learning,” in *First Workshop on Out-of-Distribution Generalization in Robotics at CoRL 2023*, 2023. [Online]. Available: <https://openreview.net/forum?id=ckFRoOaA3n>
- [13] A. Mandlekar, C. R. Garrett, D. Xu, and D. Fox, “Human-in-the-loop task and motion planning for imitation learning,” in *Conference on Robot Learning*. PMLR, 2023, pp. 3030–3060.
- [14] A. Mandlekar, J. Booher, M. Spero, A. Tung, A. Gupta, Y. Zhu, A. Garg, S. Savarese, and L. Fei-Fei, “Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity,” *arXiv preprint arXiv:1911.04052*, 2019.
- [15] A. Mandlekar, D. Xu, R. Martín-Martín, Y. Zhu, L. Fei-Fei, and S. Savarese, “Human-in-the-loop imitation learning using remote teleoperation,” *arXiv preprint arXiv:2012.06733*, 2020.
- [16] A. Tung, J. Wong, A. Mandlekar, R. Martín-Martín, Y. Zhu, L. Fei-Fei, and S. Savarese, “Learning multi-arm manipulation through collaborative teleoperation,” *arXiv preprint arXiv:2012.06738*, 2020.
- [17] J. Wong, A. Tung, A. Kurenkov, A. Mandlekar, L. Fei-Fei, S. Savarese, and R. Martín-Martín, “Error-aware imitation learning from teleoperation data for mobile manipulation,” in *Conference on Robot Learning*. PMLR, 2022, pp. 1367–1378.
- [18] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel, “Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators,” 2023.

- [19] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, “Learning fine-grained bimanual manipulation with low-cost hardware,” *arXiv preprint arXiv:2304.13705*, 2023.
- [20] 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.
- [21] Z. Fu, T. Z. Zhao, and C. Finn, “Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation,” *arXiv preprint arXiv:2401.02117*, 2024.
- [22] A. Iyer, Z. Peng, Y. Dai, I. Guzey, S. Haldar, S. Chintala, and L. Pinto, “Open teach: A versatile teleoperation system for robotic manipulation,” *arXiv preprint arXiv:2403.07870*, 2024.
- [23] S. Dass, W. Ai, Y. Jiang, S. Singh, J. Hu, R. Zhang, P. Stone, B. Abbatematteo, and R. Martin-Martin, “Telemoma: A modular and versatile teleoperation system for mobile manipulation,” *arXiv preprint arXiv:2403.07869*, 2024.
- [24] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song, “Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots,” *arXiv preprint arXiv:2402.10329*, 2024.
- [25] H. Fang, H.-S. Fang, Y. Wang, J. Ren, J. Chen, R. Zhang, W. Wang, and C. Lu, “Low-cost exoskeletons for learning whole-arm manipulation in the wild,” *arXiv preprint arXiv:2309.14975*, 2023.
- [26] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, “Rlbench: The robot learning benchmark & learning environment,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.
- [27] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani *et al.*, “Transporter networks: Rearranging the visual world for robotic manipulation,” *arXiv preprint arXiv:2010.14406*, 2020.
- [28] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan, “Vima: General robot manipulation with multimodal prompts,” *arXiv preprint arXiv:2210.03094*, 2022.
- [29] J. Gu, F. Xiang, X. Li, Z. Ling, X. Liu, T. Mu, Y. Tang, S. Tao, X. Wei, Y. Yao *et al.*, “Maniskill2: A unified benchmark for generalizable manipulation skills,” *arXiv preprint arXiv:2302.04659*, 2023.
- [30] H. Ha, P. Florence, and S. Song, “Scaling up and distilling down: Language-guided robot skill acquisition,” *arXiv preprint arXiv:2307.14535*, 2023.
- [31] Y. Wang, Z. Xian, F. Chen, T.-H. Wang, Y. Wang, K. Fragkiadaki, Z. Erickson, D. Held, and C. Gan, “Robogen: Towards unleashing infinite data for automated robot learning via generative simulation,” *arXiv preprint arXiv:2311.01455*, 2023.
- [32] D. A. Pomerleau, “Alvin: An autonomous land vehicle in a neural network,” in *Advances in neural information processing systems*, 1989, pp. 305–313.
- [33] A. J. Ijspeert, J. Nakanishi, and S. Schaal, “Movement imitation with nonlinear dynamical systems in humanoid robots,” *Proceedings 2002 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1398–1403 vol.2, 2002.
- [34] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, “One-shot visual imitation learning via meta-learning,” in *Conference on robot learning*. PMLR, 2017, pp. 357–368.
- [35] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, “Robot programming by demonstration,” in *Springer Handbook of Robotics*, 2008.
- [36] S. Calinon, F. D’halluin, E. L. Sauser, D. G. Caldwell, and A. Billard, “Learning and reproduction of gestures by imitation,” *IEEE Robotics and Automation Magazine*, vol. 17, pp. 44–54, 2010.
- [37] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, “Learning to generalize across long-horizon tasks from human demonstrations,” *arXiv preprint arXiv:2003.06085*, 2020.
- [38] C. Wang, R. Wang, D. Xu, A. Mandlekar, L. Fei-Fei, and S. Savarese, “Generalization through hand-eye coordination: An action space for learning spatially-invariant visuomotor control,” *arXiv preprint arXiv:2103.00375*, 2021.

- [39] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, “Learning latent plans from play,” in *Conference on Robot Learning*, 2019.
- [40] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim, “Demonstration-guided reinforcement learning with learned skills,” in *Conference on Robot Learning*, 2021.
- [41] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum, “Opal: Offline primitive discovery for accelerating offline reinforcement learning,” in *International Conference on Learning Representations*, 2021.
- [42] K. Hakhamaneshi, R. Zhao, A. Zhan, P. Abbeel, and M. Laskin, “Hierarchical few-shot imitation with skill transition models,” in *International Conference on Learning Representations*, 2021.
- [43] Y. Zhu, P. Stone, and Y. Zhu, “Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4126–4133, 2022.
- [44] S. Nasiriany, T. Gao, A. Mandlekar, and Y. Zhu, “Learning and retrieval from prior data for skill-based imitation learning,” in *Conference on Robot Learning (CoRL)*, 2022.
- [45] A. Kumar, A. Singh, F. Ebert, Y. Yang, C. Finn, and S. Levine, “Pre-training for robots: Offline rl enables learning new tasks from a handful of trials,” *arXiv preprint arXiv:2210.05178*, 2022.
- [46] P. Mitrano and D. Berenson, “Data augmentation for manipulation,” *arXiv preprint arXiv:2205.02886*, 2022.
- [47] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, “Reinforcement learning with augmented data,” *arXiv preprint arXiv:2004.14990*, 2020.
- [48] I. Kostrikov, D. Yarats, and R. Fergus, “Image augmentation is all you need: Regularizing deep reinforcement learning from pixels,” *arXiv preprint arXiv:2004.13649*, 2020.
- [49] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto, “Visual imitation made easy,” *arXiv e-prints*, pp. arXiv–2008, 2020.
- [50] A. Zhan, P. Zhao, L. Pinto, P. Abbeel, and M. Laskin, “A framework for efficient robotic manipulation,” *arXiv preprint arXiv:2012.07975*, 2020.
- [51] S. Sinha, A. Mandlekar, and A. Garg, “S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics,” in *Conference on Robot Learning*. PMLR, 2022, pp. 907–917.
- [52] S. Pitis, E. Creager, and A. Garg, “Counterfactual data augmentation using locally factored dynamics,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 3976–3990, 2020.
- [53] S. Pitis, E. Creager, A. Mandlekar, and A. Garg, “Mocoda: Model-based counterfactual data augmentation,” *arXiv preprint arXiv:2210.11287*, 2022.
- [54] Z. Mandi, H. Bharadhwaj, V. Moens, S. Song, A. Rajeswaran, and V. Kumar, “Cacti: A framework for scalable multi-task multi-scene visual imitation learning,” *arXiv preprint arXiv:2212.05711*, 2022.
- [55] T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, J. Singh, C. Tan, J. Peralta, B. Ichter *et al.*, “Scaling robot learning with semantically imagined experience,” *arXiv preprint arXiv:2302.11550*, 2023.
- [56] Z. Chen, S. Kiani, A. Gupta, and V. Kumar, “Genaug: Retargeting behaviors to unseen situations via generative augmentation,” *arXiv preprint arXiv:2302.06671*, 2023.
- [57] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar, “Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking,” *arXiv preprint arXiv:2309.01918*, 2023.
- [58] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox, “Rvt: Robotic view transformer for 3d object manipulation,” *arXiv:2306.14896*, 2023.
- [59] M. Shridhar, L. Manuelli, and D. Fox, “Perceiver-actor: A multi-task transformer for robotic manipulation,” in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.
- [60] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki, “Act3d: Infinite resolution action detection transformer for robotic manipulation,” *arXiv preprint arXiv:2306.17817*, 2023.

- [61] S. Belkhale, Y. Cui, and D. Sadigh, “Hydra: Hybrid robot actions for imitation learning,” in *Conference on Robot Learning*. PMLR, 2023, pp. 2113–2133.
- [62] L. X. Shi, A. Sharma, T. Z. Zhao, and C. Finn, “Waypoint-based imitation learning for robotic manipulation,” in *7th Annual Conference on Robot Learning*, 2023. [Online]. Available: <https://openreview.net/forum?id=X0cm1Th1VI>
- [63] P. Parashar, V. Jain, X. Zhang, J. Vakil, S. Powers, Y. Bisk, and C. Paxton, “Slap: Spatial-language attention policies,” in *7th Annual Conference on Robot Learning*, 2023.
- [64] Z. Xian, N. Gkanatsios, T. Gervet, T.-W. Ke, and K. Fragkiadaki, “Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation,” in *7th Annual Conference on Robot Learning*, 2023.
- [65] S. Cheng, C. Garrett, A. Mandlekar, and D. Xu, “NOD-TAMP: Multi-step manipulation planning with neural object descriptors,” in *CoRL 2023 Workshop on Learning Effective Abstractions for Planning (LEAP)*, 2023. [Online]. Available: <https://openreview.net/forum?id=DK7TbAS0Wz>
- [66] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez, “Learning compositional models of robot skills for task and motion planning,” *The International Journal of Robotics Research*, vol. 40, no. 6-7, pp. 866–894, 2021.
- [67] 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*, vol. 4, pp. 265–293, 2021.
- [68] B. Wen, W. Lian, K. Bekris, and S. Schaal, “You only demonstrate once: Category-level manipulation from single visual demonstration,” in *Robotics: Science and Systems (RSS)*, 2022.
- [69] N. Di Palo and E. Johns, “Learning multi-stage tasks with one demonstration via self-replay,” in *Conference on Robot Learning*. PMLR, 2022, pp. 1180–1189.
- [70] E. Johns, “Coarse-to-fine imitation learning: Robot manipulation from a single demonstration,” in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 4613–4619.
- [71] V. Vosylius and E. Johns, “Where to start? transferring simple skills to complex environments,” *arXiv preprint arXiv:2212.06111*, 2022.
- [72] A. Chenu, O. Serris, O. Sigaud, and N. Perrin-Gilbert, “Leveraging sequentiality in reinforcement learning from a single demonstration,” *arXiv preprint arXiv:2211.04786*, 2022.
- [73] E. Valassakis, G. Papagiannis, N. Di Palo, and E. Johns, “Demonstrate once, imitate immediately (dome): Learning visual servoing for one-shot imitation learning,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 8614–8621.
- [74] J. Liang, B. Wen, K. Bekris, and A. Boularias, “Learning sensorimotor primitives of sequential manipulation tasks from visual demonstrations,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8591–8597.
- [75] M. Stolle and D. Precup, “Learning options in reinforcement learning,” in *Abstraction, Reformulation, and Approximation: 5th International Symposium, SARA 2002 Kananaskis, Alberta, Canada August 2–4, 2002 Proceedings 5*. Springer, 2002, pp. 212–223.
- [76] R. Alami, T. Simeon, and J.-P. Laumond, “A geometrical approach to planning manipulation tasks. the case of discrete placements and grasps,” in *The fifth international symposium on Robotics research*. MIT Press, 1990, pp. 453–463.
- [77] Y. Koga, K. Kondo, J. Kuffner, and J.-C. Latombe, “Planning motions with intentions,” in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, 1994, pp. 395–408.
- [78] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, “robosuite: A modular simulation framework and benchmark for robot learning,” in *arXiv preprint arXiv:2009.12293*, 2020.
- [79] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.
- [80] P. Beeson and B. Ames, “{TRAC-IK}: An open-source library for improved solving of generic inverse kinematics,” 11 2015.

- [81] J. J. Kuffner and S. M. LaValle, “Rrt-connect: An efficient approach to single-query path planning,” in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 2. IEEE, 2000, pp. 995–1001.
- [82] O. Khatib, “A unified approach for motion and force control of robot manipulators: The operational space formulation,” *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.
- [83] B. Wen, W. Yang, J. Kautz, and S. Birchfield, “FoundationPose: Unified 6d pose estimation and tracking of novel objects,” in *CVPR*, 2024.
- [84] E. Valassakis, N. Di Palo, and E. Johns, “Coarse-to-fine for sim-to-real: Sub-millimetre precision across wide task spaces,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5989–5996.
- [85] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [86] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [87] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, “Deep spatial autoencoders for visuomotor learning,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 512–519.
- [88] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, “On the continuity of rotation representations in neural networks,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5745–5753.
- [89] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” *arXiv preprint arXiv:2303.04137*, 2023.
- [90] E. Coumans, “Bullet physics simulation,” in *ACM SIGGRAPH 2015 Courses*. ACM, 2015, p. 7.
- [91] 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*, vol. 30, 2020, pp. 440–448.
- [92] 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*, 2023, pp. 4015–4026.
- [93] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu *et al.*, “Grounding dino: Marrying dino with grounded pre-training for open-set object detection,” *arXiv preprint arXiv:2303.05499*, 2023.
- [94] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *KDD*, 1996.
- [95] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3d surface construction algorithm,” in *Seminal graphics: pioneering efforts that shaped the field*, 1998, pp. 347–353.

# Appendix

## A Overview

The Appendix contains the following content.

- **FAQ** (Appendix [B](#)): answers to some common questions
- **Limitations** (Appendix [C](#)): more thorough list and discussion of SkillGen limitations
- **Analysis on Challenging Data Generation Scenarios** (Appendix [D](#)): more results and discussion on challenging data generation scenarios addressed by SkillGen
- **Dataset Scaling Law Analysis** (Appendix [E](#)): full set of results for generating larger datasets with SkillGen
- **Data Generation Success Rates** (Appendix [F](#)): data generation success rates for SkillGen datasets
- **Data Generation Details** (Appendix [G](#)): more details on how SkillGen generates data
- **Policy Learning Details** (Appendix [H](#)): more details on how policies were trained from SkillGen datasets
- **Planning Details** (Appendix [I](#)): more details on the planners used in this work
- **Tasks and Task Variants** (Appendix [J](#)): detailed descriptions of tasks and task variants used to evaluate SkillGen
- **Sim-to-Real Experiment** (Appendix [K](#)): details on sim-to-real experiments
- **Results with Conventional Teleoperation Source Demonstrations** (Appendix [L](#)): SkillGen performance on conventional teleoperation source demos
- **Ablations** (Appendix [M](#)): ablations of certain data generation and policy learning components
- **Robot Transfer** (Appendix [N](#)): SkillGen applied to generate data and train policies across robot arms
- **Algorithm Pseudocode** (Appendix [O](#)): pseudocode for SkillGen data generation and policy deployment
- **Comparison with HITL-TAMP [13]** (Appendix [P](#)): more discussion on how SkillGen compares with HITL-TAMP
- **Discussion on HSP-Reg Results** (Appendix [Q](#)): more discussion on the gap between HSP-Reg and other methods and additional promising results
- **Skill Segments and Annotations** (Appendix [R](#)): more commentary on skill segments and how they can be annotated in the source data