IntervenGen: Interventional Data Generation for Robust and Data-Efficient Robot Imitation Learning

Ryan Hoque[1,2], Ajay Mandlekar*[2], Caelan Garrett*[2], Ken Goldberg[1], Dieter Fox[2]

***Abstract*— Imitation learning is a promising paradigm for training robot control policies, but these policies can suffer from distribution shift, where the conditions at evaluation time differ from those in the training data. A popular approach for increasing policy robustness to distribution shift is interactive imitation learning (i.e., DAgger and variants), where a human operator provides corrective interventions during policy rollouts. However, collecting a sufficient amount of interventions to cover the distribution of policy mistakes can be burdensome for human operators. We propose IntervenGen (I-Gen), a novel data augmentation system for robot control that autonomously produces a large set of corrective interventions with rich coverage of the state space from a small number of human interventions. We apply I-Gen to 4 simulated environments and 1 physical environment with object pose estimation error and show that it can increase policy robustness by up to 39× with only 10 human interventions. Videos and more results are available at `https://sites.google.com/view/intervengen2024`.**

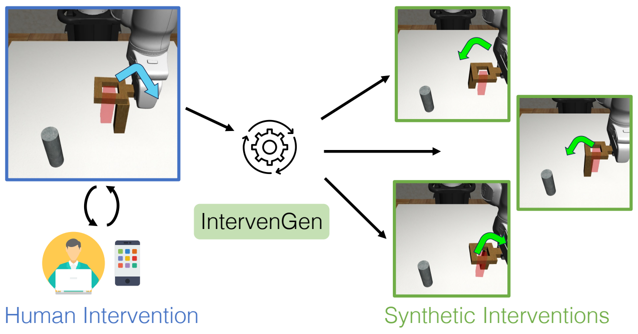

Fig. 1: **Overview.** IntervenGen automatically generates corrective interventional data from a small number of human interventions, with coverage across both diverse scene configurations and policy mistake distributions. Here, the robot mistakenly believes the peg is at the position in red and requires demonstration of recovery behavior toward the true peg position.

I. Introduction

Imitation Learning (IL) from human demonstrations is a promising paradigm for training robot policies. One approach is to collect a set of offline task demonstrations via human teleoperation [1,2] and employ behavior cloning (BC) [3] to train robot policies via supervised learning, where the labels are robot actions. There have been recent efforts to scale this approach by collecting thousands of demonstrations using hundreds of human operator hours and training high-capacity neural networks on the large-scale data [4–8].

However, IL policies can suffer from distribution shift, where the conditions at evaluation time differ from those in the training data [9]. As an example, consider a policy that makes decisions based on object pose observations. A common source of distribution shift in the real world is object pose estimation error, which can occur due to a wide range of factors such as sensor noise, occlusion, network delay, and model misspecification. This can cause inaccuracy in the robot's belief of where critical objects are located in the environment, leading the robot to visit states outside the training distribution that result in poor policy performance.

One approach to addressing distribution shift is to collect a large set of demonstrations under diverse conditions and hope that agents trained on this data can generalize. However, human teleoperation data is notoriously difficult to collect due to the human time, effort, and financial cost required [4–8].

An alternative approach is *Interactive IL*, including DAgger [9] and its variants [10–12], in which humans can intervene during robot execution and demonstrate *recovery behaviors* to help the robot return to the support of the training distribution. Subsequent training on these corrections can increase policy robustness and performance both theoretically and in practice [9]. However, interactive IL imposes even more burden on the human supervisors than behavior cloning, as the human must continuously monitor robot task execution and intervene when they see fit, typically over multiple rounds of interleaved data collection and policy training. Moreover, a significant amount of recovery data may be required to adequately cover the distribution of mistakes the policy may make.

We raise the following question: do we actually need to have a human operator intervene every single time a policy makes a mistake? MimicGen [13], a recently proposed data generation system, raises an intriguing possibility: a large dataset of synthetically generated demonstrations derived from a small set of human demonstrations (typically 100× smaller or more) can produce performant robot policies. The system's key insight is that similar object-centric manipulation behaviors can be applied in new contexts by appropriately transforming demonstrated behavior to the new object frame. Inspired by this insight, we propose a data generation system for *interventional* data (see Fig. 1). With a small set of corrective interventions from a human operator, the system autonomously generates data with significantly higher coverage of the distribution of potential policy mistakes. Such a system has a broad range of applications such as improving policy success rates on a task of interest, making policies robust to errors in perception, and more broadly, acting as a domain randomization [14] procedure to aid in sim-to-real transfer of IL policies without requiring additional data collection from a human supervisor. In this work, we focus

[1]UC Berkeley, [2]NVIDIA, *Equal contribution.

on improving policy robustness to errors in perception.

We make the following contributions:

1) IntervenGen (I-Gen), a system for automatically generating interventional data across diverse scene configurations and broad mistake distributions from a small number of human interventions.
2) An application of I-Gen to improving policy robustness against 2 sources of object pose estimation error (sensor noise and geometry error) in 5 high-precision 6-DOF manipulation tasks. I-Gen increases policy robustness by up to 39× with only 10 human interventions.
3) Experiments demonstrating the utility of I-Gen over alternate uses of a human data budget of equivalent or even greater size. A policy trained on synthetic I-Gen data from 10 source human interventions can outperform one trained on even 100 human interventions by 24%, with 12% of the data collection time and effort.
4) An experiment that shows that policies trained in simulation with I-Gen are amenable to real-world deployment and retain robustness to erroneous state estimation.

II. Related Work

Data Collection for Robot Learning. Many prior works address the need for large-scale data in robotics. Some use self-supervised data collection [15,16], but the data can have low signal-to-noise ratio due to the trial-and-error process. Other works collect large datasets using experts that operate on privileged information available in simulation [17–19]. Still, designing such experts can require significant engineering. One popular approach is to collect demonstrations by having human operators teleoperate robot arms [1,2,4,5]; however, this can require hundreds of hours of human operator time. Some systems also allow for collecting interventions to help correct policy mistakes [11,20,21]. In this work, we make effective use of a handful of interventional corrections provided by a single human operator to autonomously generate large-scale interventional data, substantially reducing the operator burden.

Interactive Imitation Learning. Interactive IL allows demonstrators to provide corrective supervision in situations where policies require assistance. Some approaches require an expert to relabel states encountered by the agent with actions that the expert would have taken [9,22], but it can be difficult for human supervisors to relabel robot actions in hindsight [23]. An alternative is to cede control of the system to a human supervisor for short corrective trajectories (termed *interventions*) in states where the robot policy needs assistance. Interventional data collection can either be human-gated [10,20], where the human monitors the policy and decides when to provide interventions, or robot-gated [12,24,25], where the robot decides when the human should provide interventions. However, these approaches require collecting a sufficient number of human interventions for the robot to learn robust recovery. In this work, we develop a novel data generation mechanism based on replay-based imitation [13,26,27] in order to alleviate this burden.

MimicGen. MimicGen [13] is a recently proposed system for automatically generating task demonstrations via trajectory adaptation with respect to known object poses. I-Gen employs a similar mechanism for synthesizing trajectories but has several key differences. Unlike MimicGen, I-Gen (1) generates interventional data rather than full demonstrations, (2) relaxes the assumption of precise object pose knowledge, which is critical to MimicGen's success, (3) integrates closed-loop policy execution that allows the robot to visit novel states during the data generation process, and (4) allows variation in not just object poses but also robot belief states about these object poses.

III. Preliminaries

Problem Statement. We model the task environment as a Partially Observable Markov Decision Process (POMDP) with state space S, observation space O, and action space A. The robot does not have access to the transition dynamics or reward function but has a dataset of samples $D = \{(o,a)\}_{i=1}^{N}$ from an expert human policy $\pi_H : O \to A$. We assume that while the human observes observation o, the robot's observation is corrupted by some function z, yielding $z(o) = o' \in O$ (e.g., due to sensor noise or network delay). In this work we train policies on demonstration datasets D using supervised learning with the objective $\arg\min_\theta \mathbb{E}_{(o,a)\sim D}[-\log \pi_\theta(a|o)]$.

Assumptions. I-Gen has assumptions similar to MimicGen [13]. **(Assumption 1)** the action space consists of delta-pose commands in Cartesian end effector space; **(Assumption 2)** the task is a known sequence of object-centric subtasks; **(Assumption 3)** object poses can be observed at the beginning of each subtask during data collection (but not deployment). **(Assumption 4)** We also assume that demonstrated recovery behavior can be explained by some component of the robot's observations $\{o'_1, o'_2, \dots\}$ during a human intervention despite corruption by z. Without this assumption, it would not be possible for the robot to learn a policy that maps o' to $\pi_H(o)$. This information can be provided, for instance, in additional observation modalities such as force-torque sensing or tactile sensing that provide a coarse signal about an object's pose. Some settings may not require any additional information: for example, a fully closed gripper can inform the robot that it must recover from a missed grasp.

MimicGen Data Generation System. MimicGen [13] takes a small set of source human demonstrations D_{src} and uses it to automatically generate a large dataset D in a target environment. It first divides each source trajectory $\tau \in D_{src}$ into object-centric manipulation segments $\{\tau_i\}_{i=1}^{M}$, each of which is defined as an object-centric subtask (Assumption 2 above). Each segment is a sequence of end effector poses. Then, to generate a demonstration in a new scene, it uses the pose of the object corresponding to the current subtask and transforms the poses in a source human segment τ_i (with an SE(3) transform) such that the relative poses between the end effector and the object frame are preserved between the source demonstration and the new scene. It also adds an interpolation segment between the robot's current configuration and the start of the transformed segment. Then, the sequence of

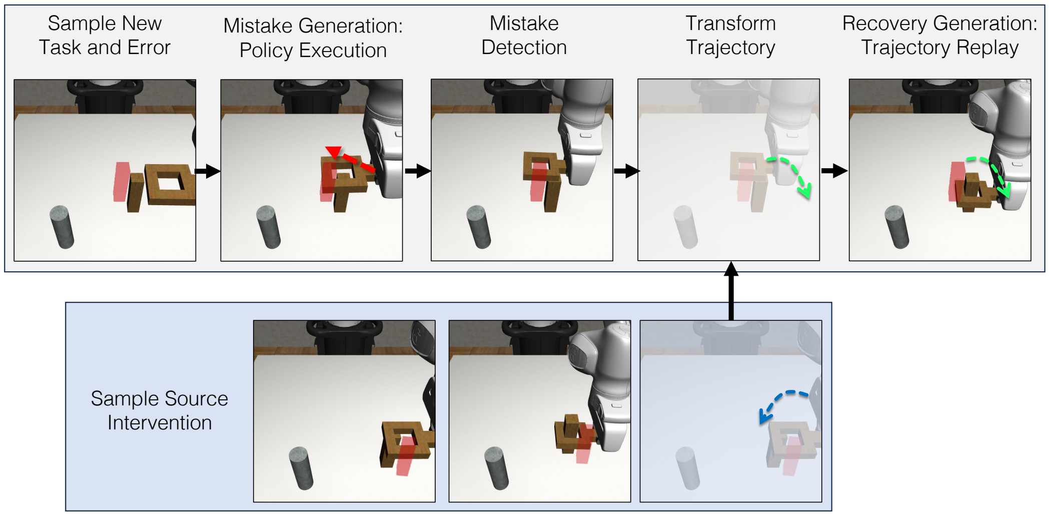
Sample New Task and Error
Mistake Generation: Policy Execution
Mistake Detection
Transform Trajectory
Recovery Generation: Trajectory Replay
Sample Source Intervention

Fig. 2: **I-Gen Data Generation Example.** We provide an example of how I-Gen generates a new intervention. First, a new task instance is sampled with a new configuration (square peg location) and observation corruption (incorrect peg location highlighted in red). We execute the robot policy to generate mistake behavior for the new task instance. When a mistake is detected, we sample a human intervention segment from the source dataset and transform it to adapt to the current scene. Finally, we executed the transformed recovery segment in the environment.

poses in the interpolation segment and transformed segment are executed by the robot end effector controller open-loop until the current subtask is complete, at which point the process repeats for the next subtask. We use a data generation mechanism similar to MimicGen to generate intervention trajectory segments in Section IV-C.

IV. InterventGen

On each of one or more iterations, the IntervenGen system: (1) trains a policy π_θ on the current dataset; (2) rolls out π_θ for interventional data collection with human teleoperation (Sec. IV-A); (3) synthesizes new interventions with closed-loop policy execution and open-loop trajectory replay (Sec. IV-B and Sec. IV-C); (4) returns the new synthetic dataset (Sec. IV-D). Full pseudocode can be found on the website.

A. Interventional Data Collection

We consider human-gated interventions [10], in which the human monitors the robot policy execution and intermittently takes control to correct policy mistakes. As in DAgger [9], this enables the human to demonstrate corrective recovery behavior from mistakes made by the robot policy that otherwise would not be visited in full human task demonstrations (due to distribution shift). The base robot policy π_θ executed during interventional data collection can come from anywhere, but is typically initialized from behavior cloning on an initial set of offline task demonstrations D [9,11,12]. Each collected trajectory can be coarsely divided into robot-generated "mistake" segments and human-generated "recovery" segments.

B. Mistake Generation: Closed-Loop Policy Execution

We aim to use the collected human interventions to automatically synthesize interventions for new scene configurations. Recall that in prior work, MimicGen generates data by executing a sequence of object-centric trajectories in an open-loop manner. In contrast, an appealing property of our interventional IL setting is access to the robot policy π_θ that is executed during interventional data collection with the human operator.

We use this robot policy during the *data generation* process as well to broaden the distribution of visited mistake states. Unlike MimicGen, we can execute the policy in new scene configurations. This has two benefits: (1) rather than assuming the policy will fail in the same manner as the source trajectory, the generated mistake will reflect the genuine behavior of the policy in the new configuration, and (2) it becomes possible to generate new mistake trajectories for new corruptions of the observed object poses. For example, if sensor noise corrupts the object pose during interventional data collection, a new noise corruption can be applied during the data generation process. This allows data diversity in both object poses and the robot's erroneous beliefs about where the objects are (see Fig. 2). However, the use of policy execution during data generation requires that we know when to stop. In our experiments, we use contact detection to terminate policy execution upon contact. A more flexible option could be to use a learned classifier or robot-gated intervention criteria such as ThriftyDAgger [12].

C. Recovery Generation: Open-Loop Trajectory Replay

In each episode of synthetic data generation, once we have completed policy execution and entered a new mistake state, we generate a recovery trajectory. We select a random source trajectory, segment out the human recovery portion of the trajectory, and adapt the trajectory to the current environment state. This adaptation consists of (1) transforming the source trajectory to the current object pose, (2) linearly interpolating in end-effector space to the beginning of the transformed trajectory, and (3) executing the transformed trajectory open-loop (see Fig. 2). Note that each object-centric subtask in a single task instance can have zero, one, or multiple instances of alternating between mistake and recovery.

D. Output Filtering and Dataset Aggregation

It is possible that the executed trajectory may not complete the task successfully. For instance, the recovery trajectory may be unable to recover from the new mistake state reached by the robot. Consequently, we only keep the generated demonstration if it successfully completes the task. We also filter out the segment of the synthetic demonstration that corresponds to the human recovery segment; such filtering is used by common algorithms such as DAgger [9] and HG-DAgger [10] and can prevent the imitation of mistakes. Each filtered episode of synthetic data is aggregated into the base dataset D (used to train the base policy π_θ), and the policy is retrained on the new dataset after data generation. If desired, the entire process of data collection, data generation, and policy training can be iterated.

E. Inter-Subtask Recovery and Offline Mode

The I-Gen framework accommodates additional modules not considered in the main set of experiments that greatly increase its range of applications, including (1) policy recovery from more severe failure modes that revert to earlier subtasks and (2) "offline" I-Gen, which allows humans to demonstrate mistakes intentionally [28]. We include experiments for these modules on the supplemental website.

V. Experiment Setup

We consider 4 tasks in the MuJoCo [29] robosuite simulation environment [30] (Fig. 3) and 1 physical experiment. Each task involves 6DOF contact-rich manipulation via continuous control. The tasks vary in object geometry, object pose, observation error, and number of subtasks.

Nut Insertion: The robot must place a square nut (held in-hand) onto a square peg. The peg position is sampled in a 10 cm x 10 cm region at the start of each episode.

2-Piece Assembly: The robot must place an object into a square receptacle with a narrow affordance region. The receptacle position is sampled in a 10 cm x 10 cm region at the start of each episode.

Coffee: The robot must place and release a coffee pod into a coffee machine pod holder with a narrow affordance region. The coffee machine position is sampled in a 10 cm x 10 cm region at the start of each episode.

Nut-and-Peg Assembly [30,31]: A multi-stage task consisting of (1) grasping a nut with a varying initial position and orientation and (2) placing it on a peg in a fixed target location. The nut is placed in a 0.5 cm x 11.5 cm region with a random top-down rotation at the start of each episode.

Physical Block Grasp: A Franka robot arm must reach a block and grasp it. The initial block position is sampled in a 20 cm x 30 cm region at the start of each episode.

Sources of Observation Error. The observation space consists of robot proprioception (6DOF end effector pose and gripper finger width) and object poses. In most environments, the source of observation error is *sensor noise* during object pose estimation: at test time, a random 2D offset is uniformly sampled at the beginning of the episode and then fixed for the remainder of the episode. It is applied to the ground truth position of the peg (with noise up to ± 4 cm in each dimension, and at least 2 cm in one dimension), receptacle (± 4 cm in each dimension, with at least 1 cm in one dimension), coffee machine (radial noise between 2 cm and 4 cm), and block (± 1 cm in x and ± 7 cm in y, with at least 2.5 cm in y) respectively. In the Nut-and-Peg Assembly environment, the source of observation error is *object geometry*: for an identical observed nut pose, the nut handle may exist on either one of two sides of the nut. This setting corresponds to object model misspecification during pose registration.

A. Experimental Setup

Data Collection. For interventional data collection, we use the remote teleoperation system proposed by Mandlekar et al. [11]. The action space consists of 6DOF pose deltas and a binary gripper open/close command (except for Block Grasp, which uses 3DOF position control with fixed rotation). For the base policy π_θ used in each task, we (1) collect 10 full human task demonstrations in each environment *without* observation corruption (i.e., ground truth poses), (2) synthesize 1000 demonstrations with MimicGen [13], and (3) train an off-the-shelf BC-RNN policy with default hyperparameters using the robomimic framework [31], with the exception of an increased learning rate of 0.001 [13].

Data Generation. We then deploy π_θ in the test environment *with* observation corruption (i.e., object pose error) and collect 10 human-gated interventions. These interventions are expanded to 1000 synthetic interventions with I-Gen and aggregated with the 1000 synthetic demonstrations used to train the base policy. Finally, we train a new BC-RNN policy on the aggregated dataset. We report policy performance as the success rate over 50 trials for the highest performing checkpoint during training (where training takes 2000 epochs with evaluation every 50 epochs), as in [13,31].

Observability. In order for demonstrated recovery behavior to be learnable (Section III), I-Gen and all baselines can access additional observation information in Nut Insertion, Two-Piece Assembly, Coffee, and Block Grasp upon contact between (1) the nut and peg, (2) object and receptacle, (3) pod and pod holder, and (4) gripper and cube, respectively. We study both the idealized case of full observability (i.e., ground truth pose) upon contact in Section VI and partially improved observability (e.g., position of contact) in Section VI-A. These are intended to be surrogates for sensor modalities such as force-torque sensing that can help inform the robot about the object pose when its belief is wrong. For Nut-and-Peg Assembly, we do not add additional information, as a closed gripper state is sufficient for the policy to map a missed grasp to learned recovery.

Physical Experiment Setup. We wish to evaluate whether or not policies trained on simulation data from I-Gen can retain their robustness to erroneous state estimation when they are deployed directly in the real world. To do this, we train a policy for the Block Grasp task in simulation and deploy it zero-shot on a physical robot. We use a Franka Research 3 robot arm and gripper and a red cube with a side

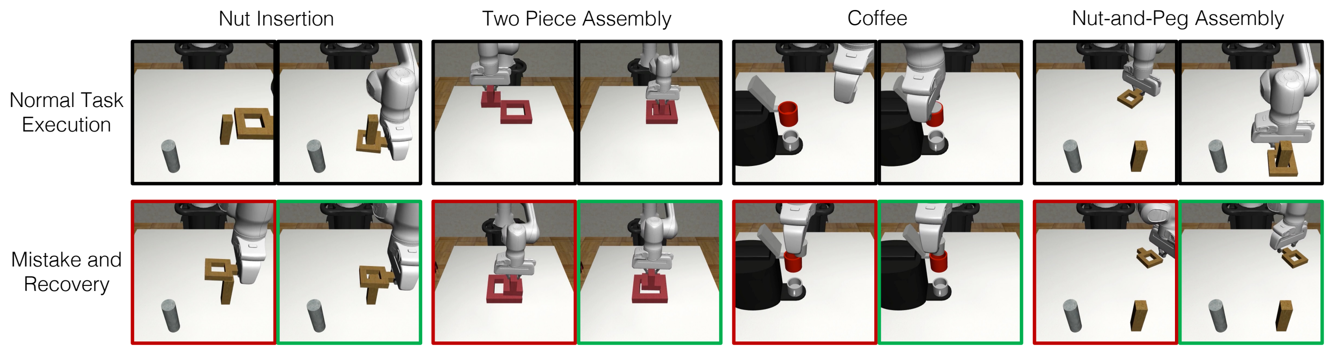

Fig. 3: **Tasks.** We evaluate I-Gen in several contact-rich, high-precision tasks. The top row shows normal task execution while the bottom row shows typical mistakes encountered by the agent when using inaccurate object poses (or object geometry for Nut-and-Peg Assembly) and associated recovery behaviors.

length of 5 cm. We use an Intel RealSense D415 depth camera and Iterative Closest Point (ICP) for cube pose estimation. The deployed policies output continuous control delta-pose actions at 20 Hz and do not require any real-world data or fine-tuning. See Figure 4 for images of the transfer process.

B. Baselines

We implement and evaluate the following baselines. Each baseline corresponds to a *different dataset* used to train the agent (all agents are trained with BC-RNN [31]):

Base: Deploy the base policy in the test environment without any additional data or fine-tuning.

Source Interventions (Source Int): Deploy the base policy π_θ, collect 10 human interventions when the policy makes mistakes, and add them to the base dataset.

Weighted Source Interventions (Weighted Src Int) [11]: Same as Source Interventions, but weight the intervention data higher so that it is sampled as frequently as the base data despite its smaller quantity.

Source Demonstrations (Source Demo): Collect 10 full human task demonstrations in the test environment.

MimicGen Demonstrations (MG Demo) [13]: Same as Source Demonstrations, but use (regular) MimicGen to generate 1000 synthetic demonstrations from the initial 10.

Policy Execution Ablation (I-Gen - Policy): Augment 10 source interventions to 1000 with I-Gen, but use open-loop replay instead of policy execution to generate mistakes.

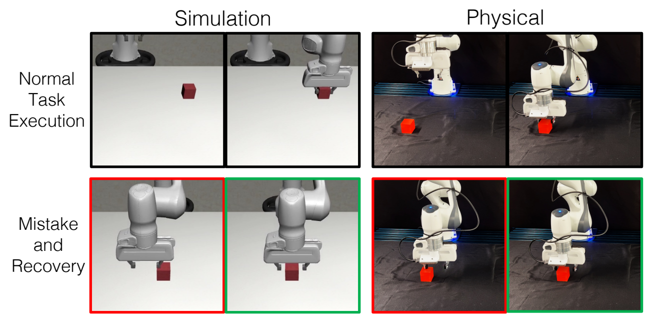

Fig. 4: **Sim-to-Real.** We evaluate sim-to-real transfer for a block grasping task with a Franka Panda robot. Similar to Figure 3 we show normal task execution, typical mistakes due to inaccurate object poses, and associated recovery for the simulation and real world environments. The results show that I-Gen can facilitate sim-to-real transfer of learned control policies, and that these policies retain robustness to erroneous perception.

VI. Experiments

In this section, we summarize the key takeaways from the comparisons presented in Tables I, II, and IV.

I-Gen vastly improves policy robustness under pose estimation error. In Table I, we observe that I-Gen improves policy performance by 3.5×, 10.7×, and 39× over the base policy in Nut Insertion, 2-Piece Assembly, and Coffee respectively, despite only collecting 10 human interventions.

I-Gen significantly improves upon naïve uses of an equivalent amount of full human demonstration data. I-Gen consistently outperforms human demonstrations collected at test time (Source Demo, Table I) by 56%-68%. Even if these demonstrations are expanded by 100× with MimicGen (MG Demo), I-Gen still outperforms by 34%-62%. Since the human's observability does not match the robot's, the human can teleoperate toward the true object poses. Thus, the robot does not observe any recovery behavior in the offline data.

I-Gen significantly improves upon naïve uses of an equivalent amount of interventional human data. Source Int in Table I underperforms I-Gen by 58%-70%. While helpful, with only 10 human interventions, the data is insufficient to learn robust recovery under pose error. This remains the case even if the intervention data is weighted higher, in which case the agent overfits to the 10 interventions and underperforms I-Gen by 48%-74%. With the same budget of interventional human data, I-Gen can generate much richer coverage of the distribution of mistakes under the base policy.

I-Gen significantly improves upon naïve uses of MimicGen. We observe a significant 34%-62% improvement over MimicGen on full task demonstrations (MG Demo, Table I). We also observe that the policy execution component (Section IV-B) boosts performance by 12%-38% respectively over the ablation, indicating that expanding the mistake distribution is valuable. While the ablation dataset covers variation in the object pose, it does not cover variation in the error; only the 10 mistake segments in the source dataset are available. This shows that the novel components we introduced in I-Gen are crucial for high performance.

I-Gen is useful across different environments. While 2-Piece Assembly and Coffee have narrower tolerance regions than Nut Insertion that lower success rates across the board (16%-20% for the base policy, 30%-48% for other baselines, and 18%-28% for I-Gen), the relative performance of I-Gen remains consistent across environments: I-Gen outperforms all baselines by 12%-76% in Nut Insertion, 18%-64% in 2-Pc Assembly, and 38%-78% in Coffee.

I-Gen is useful across different sources of observation error. Results for the Nut-and-Peg Assembly task with object geometry error are in Table II. We evaluate each policy with 50 evaluations of each of the two possible geometries. Base and Source Int attain perfect performance on the original geometry but struggle with the alternate geometry (0%-6% performance). MG Demo has the opposite issue: since it consists of test-time demonstrations with the alternate geometry, it can attain perfect performance on the alternate but 0% on the original. A mixture of full demonstrations on both geometries (Base + MG Demo) attains an even 60% and 64%; since it does not observe recovery behavior it must guess between the two object geometries and has difficulty performing much higher than the 50% expected value of random chance. Finally, I-Gen maintains 92% performance on the original geometry but also learns to recover when missing its grasp due to the alternate geometry (88%), leading to a 28%-40% improvement in the average case over baselines. See the website for videos.

I-Gen facilitates sim-to-real transfer of learned control policies, and these policies retain robustness to erroneous state estimation. In Table IV we observe that state-based policies for the Block Grasp task deployed zero-shot on the physical system perform similarly to simulation. By improving robustness to incorrect pose estimation, I-Gen facilitates sim-to-real transfer for state-based policies, which are easier to transfer across visual domain gaps than image-based policies but rely on accurate perception. I-Gen outperforms baselines by 14%-94% in simulation and 30%-90% in real world trials, suggesting learned recovery behaviors can transfer to real. The policy is also robust to physical perturbations, dynamic object pose changes, and visual distractors; see the website for videos.

A. Analysis

In this section, we present further analysis on various aspects of I-Gen.

How is agent performance affected as observability decreases? For Nut Insertion, we replace true pose information upon contact with the mean position of the first contact between the nut and peg; for 2-Piece Assembly, we provide the unit vector in the direction of the true pose at the first point of contact. Table III in comparison with Table I shows that, as expected, a degradation in observability results in a degradation in agent performance. However, I-Gen performance falls by only 4%-8%, indicating partial observability can be sufficient to ground recovery behavior. An important direction for future work is investigating raw real-world perception signals such as force-torque sensing.

How does performance vary across training seeds? I-Gen in the (full observability) Nut Assembly task attains 98%, 100%, and 98% for 3 training seeds, indicating stability across runs (more evidence on supplemental website).

Dataset	Nut Insertion	2-Pc Assembly	Coffee
Base	22%	6%	2%
Source Int	40%	6%	10%
Weighted Src Int [11]	50%	16%	6%
Source Demo	42%	12%	12%
MG Demo [13]	64%	16%	18%
I-Gen - Policy (Ours)	86%	52%	42%
I-Gen (Ours)	**98%**	**70%**	**80%**

TABLE I: Results in three simulation domains with noisy pose estimation and full observability upon contact. I-Gen outperforms baselines across environments.

Dataset	Geometry 1	Geometry 2	Mixture
Base	**100%**	0%	50%
Source Int	**100%**	6%	53%
MG Demo [13]	0%	**100%**	50%
Base + MG Demo	64%	60%	62%
I-Gen	92%	88%	**90%**

TABLE II: Results in the Nut-and-Peg Assembly experiment. While baselines typically overfit to one geometry or struggle with disambiguating the two, I-Gen attains high performance on the mixture of geometries.

Dataset	Nut Insertion	2-Pc Assembly
Base	26%	6%
Source Int	40%	6%
MG Demo [13]	46%	22%
I-Gen - Policy	68%	42%
I-Gen	**90%**	**66%**

TABLE III: Additional evaluation in two domains with partially improved (rather than full) observability upon contact.

Dataset	Simulation	Real
Base	6%	0%
Source Int	26%	10%
MG Demo [13]	42%	50%
I-Gen - Policy	86%	60%
I-Gen	**100%**	**90%**

TABLE IV: Sim-to-real results for the block grasping task in simulation (50 trials) and zero-shot evaluation of these policies in the real world (10 trials).

How does synthetic IntervenGen data compare to an equal amount of human data? In 2-Piece Assembly, 100 I-Gen interventions (from 10 human interventions) attain 24% while 100 human interventions attain 46%. Both improve upon 10 human interventions, which only attains 6% (Table I). However, 1000 I-Gen interventions from 10 human interventions attains 70%, outperforming 100 human interventions. I-Gen from 10 interventions also takes significantly less human time and effort to collect than 100 human interventions (3.6 minutes instead of 29.9 minutes).

How does performance scale with the amount of synthetically generated interventions? With the same 10 human source interventions in 2-Piece Assembly, an agent trained on 200 synthetic I-Gen interventions attains 34%, 1000 interventions attains 70% (Table I), and 5000 interventions attains 88%. This suggests performance scales with dataset size, at the cost of additional data generation time.

VII. Conclusion

We present IntervenGen (I-Gen), a data generation system for corrective interventions that cover a large distribution of policy mistakes given a small number of source human interventions. Results suggest that training on synthetic data generated by I-Gen compares favorably to collecting more human demonstrations and interventions in terms of both policy performance and human effort.

Although I-Gen improves on MimicGen and reduces its reliance on accurate pose estimation, I-Gen shares some of its limitations. Specifically, we consider only quasi-static tasks with rigid body objects, and we assume valid interventions can be synthesized by transforming source trajectory data.

Future work involves deploying I-Gen with force-torque sensing to improve behavioral adaptation for contact-rich and high-precision tasks. I-Gen can also be used to rapidly adapt policy behavior toward individual human preferences without extensive data collection. Finally, I-Gen can also be applied to facilitating sim-to-real transfer of IL policies by acting as a domain randomization [14] procedure. While RL algorithms can autonomously learn adaptations to dynamical domain randomization, IL typically requires generating new human behavior for these variations. I-Gen may dramatically reduce the data requirements and enable policies to deal with such variations with only a handful of corrective behaviors.

Acknowledgements

This work was made possible with the support of the NVIDIA Seattle Robotics Lab. We especially thank Ankur Handa for valuable discussions, Ravinder Singh for IT help, and Sandeep Desai for robot hardware support.

References

[1] 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 (CoRL)*, 2018.

[2] 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," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[3] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Neural Information Processing Systems (NeurIPS)*, 1988.

[4] 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*, 2021.

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, *et al.*, "Rt-1: Robotics transformer for real-world control at scale," in *Robotics: Science and Systems (RSS)*, 2023.

[6] 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," in *Robotics: Science and Systems (RSS)*, 2022.

[7] 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," in *Conference on Robot Learning (CoRL)*, 2022.

[8] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. K. Baruch, T. Armstrong, and P. R. Florence, "Interactive language: Talking to robots in real time," in *IEEE Robotics and Automation Letters*, 2023.

[9] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[10] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "Hg-dagger: Interactive imitation learning with human experts," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8077–8083, 2018.

[11] A. Mandlekar, D. Xu, R. Martin-Martin, Y. Zhu, L. Fei-Fei, and S. Savarese, "Human-in-the-loop imitation learning using remote teleoperation," *ArXiv preprint arXiv:2012.06733*, 2020.

[12] R. Hoque, A. Balakrishna, E. Novoseller, A. Wilcox, D. S. Brown, and K. Goldberg, "ThriftyDAgger: Budget-aware novelty and risk gating for interactive imitation learning," in *Conference on Robot Learning (CoRL)*, 2021.

[13] 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 *Conference on Robot Learning (CoRL)*, 2023.

[14] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[15] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman, "Scaling up multi-task robotic reinforcement learning," in *Conference on Robot Learning*, 2021.

[16] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, "Robonet: Large-scale multi-robot learning," in *Conference on Robot Learning*, 2019.

[17] 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," in *International Conference on Machine Learning (ICML)*, 2023.

[18] M. Dalal, A. Mandlekar, C. Garrett, A. Handa, R. Salakhutdinov, and D. Fox, "Imitating task and motion planning with visuomotor transformers," in *Conference on Robot Learning*, 2023.

[19] M. J. McDonald and D. Hadfield-Menell, "Guided imitation of task and motion planning," in *Conference on Robot Learning*, 2022.

[20] J. Luo, O. Sushkov, R. Pevceviciute, W. Lian, C. Su, M. Vecerik, N. Ye, S. Schaal, and J. Scholz, "Robust multi-modal policies for industrial assembly via reinforcement learning and demonstrations: A large-scale study," in *Robotics: Science and Systems*, 2021.

[21] H. Liu, S. Nasiriany, L. Zhang, Z. Bao, and Y. Zhu, "Robot learning on the job: Human-in-the-loop autonomy and learning during deployment," in *Robotics: Science and Systems*, 2022.

[22] S. Chernova and M. Veloso, "Interactive policy learning through confidence-based autonomy," *Journal of Artificial Intelligence Research*, vol. 34, pp. 1–25, 2009.

[23] M. Laskey, C. Chuck, J. Lee, J. Mahler, S. Krishnan, K. Jamieson, A. Dragan, and K. Goldberg, "Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations," in *International Conference on Robotics and Automation (ICRA)*, 2017.

[24] R. Hoque, A. Balakrishna, C. Putterman, M. Luo, D. S. Brown, D. Seita, B. Thananjeyan, E. Novoseller, and K. Goldberg, "LazyDAgger: Reducing context switching in interactive imitation learning," in *IEEE Conference on Automation Science and Engineering (CASE)*, 2021.

[25] R. Hoque, L. Y. Chen, S. Sharma, K. Dharmarajan, B. Thananjeyan, P. Abbeel, and K. Goldberg, "Fleet-dagger: Interactive robot fleet learning with scalable human supervision," in *Conference on Robot Learning (CoRL)*, 2022.

[26] B. Wen, W. Lian, K. E. Bekris, and S. Schaal, "You only demonstrate once: Category-level manipulation from single visual demonstration," in *Robotics: Science and Systems (RSS)*, 2022.

[27] E. Johns, "Coarse-to-fine imitation learning: Robot manipulation from a single demonstration," *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[28] D. Brandfonbrener, S. Tu, A. Singh, S. Welker, C. Boodoo, N. Matni, and J. Varley, "Visual backtracking teleoperation: A data collection protocol for offline image-based reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[29] 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 (IROS)*, 2012.

[30] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu, "Robosuite: A modular simulation framework and benchmark for robot learning," in *arXiv preprint arXiv:2009.12293*, 2020.

[31] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martin-Martin, "What matters in learning from offline human demonstrations for robot manipulation," in *Conference on Robot Learning (CoRL)*, 2021.